



InduSoft Web Studio™ v7.0

User Guide and Technical Reference



InduSoft® is a registered trademark of InduSoft, Inc.
InduSoft Web Studio™, EmbeddedView™, and CEView™ are trademarks of InduSoft, Inc.

Windows, Windows XP, Windows XP Embedded, Windows Embedded Standard 7, Windows Embedded Compact, Windows 2003 Server, Windows 2008 Server, Windows Vista, Windows 7, Windows CE, and Internet Explorer are registered trademarks of Microsoft Corp. in the United States and other countries.

Other brand or product names are trademarks or registered trademarks of their respective owners.

Copyright © 2010 InduSoft, Inc. All rights reserved worldwide.
This document shall not be reproduced or copied in any manner without expressed written authorization from InduSoft.

The information contained within this document is subject to change without notice.
InduSoft, Inc. does not guarantee the accuracy of the information.

Contents

INTRODUCTION.....	12
CONVENTIONS USED IN THIS DOCUMENTATION.....	14
ABOUT THIS APPLICATION.....	15
Product Overview.....	16
Product Features.....	17
How the Software Works.....	19
Internal Structure and Data Flow.....	19
Executing/Switching Modules.....	20
Executing/Switching the Background Task.....	24
INSTALLATION.....	26
System requirements.....	27
Installing the Software.....	29
Starting the Software.....	32
Uninstalling the Software.....	33
LICENSING.....	34
Protection Types.....	35
License Settings.....	36
Execution Modes.....	37
Product Versions.....	39
Installing a New Hardkey License.....	40
Upgrading the Current Hardkey License.....	41
Installing a Softkey License.....	43
Invalid Licenses.....	45
Installing or Upgrading a CEView License (Locally).....	46
Installing or Upgrading a CEView License (Remotely).....	49
THE DEVELOPMENT ENVIRONMENT.....	51
Application button.....	52
Recent Projects.....	52
New.....	52
Open Project.....	54
Open.....	55
Save.....	55
Save As.....	55
Save All.....	55
Save All as HTML.....	55
Save as HTML.....	55
Save Screen Group as HMTL.....	55
Print.....	56
Print Preview.....	56
Print Setup.....	56
Close.....	56
Close All.....	57
Exit.....	57
Quick Access Toolbar.....	58
Ribbon.....	59
Home tab.....	59
View tab.....	59
Insert tab.....	60
Project tab.....	60
Graphics tab.....	60
Format tab.....	61
Help tab.....	61

Project Explorer.....	62
Global tab.....	62
Graphics tab.....	63
Tasks tab.....	63
Comm tab.....	64
Screen/Worksheet Editor.....	65
Database Spy.....	66
Output (LogWin).....	67
Title Bar.....	69
Status Bar.....	70
CREATING A NEW PROJECT.....	71
Creating a new project.....	72
About target platforms and product types.....	73
Changing the product type of an existing project.....	74
Configuring additional project settings.....	75
Information tab.....	75
Options tab.....	76
Viewer tab.....	81
Communication tab.....	84
Web tab.....	85
Preferences tab.....	88
Enabling mobile access to your project.....	90
Configuring your project's default email settings.....	94
Configuring your project's default FTP settings.....	95
Starting Modules on the Target System.....	96
Running a Project Under Windows Services.....	98
TAGS AND THE PROJECT DATABASE.....	104
About Tags and the Project Database.....	105
Project Tags Folder.....	105
Classes Folder.....	107
Shared Database Folder.....	109
System Tags Folder.....	109
Designing a Tag.....	110
Naming the Tag.....	110
Choosing the Tag Type.....	110
Choosing the Tag Data Type.....	112
Choosing the Tag Scope.....	113
Creating Database Tags.....	114
Adding Tags to the Datasheet.....	114
Creating Tags "On-the-Fly".....	114
Editing Tags.....	115
Creating Classes.....	117
Setting Tag Properties.....	118
Understanding Tag Properties and Parameters.....	118
Using Tag Properties: Alarms.....	119
Using Tag Properties: History.....	121
List of Tag Properties.....	121
Using Tags in Your Project.....	126
Deleting a tag from the project database.....	127
Using the Tags Toolbar.....	128
Global Replace.....	128
Replace.....	128
Remove unused tags.....	129
Reset tags database.....	130
Tag Name text box.....	130
Object Finder.....	131
Cross Reference.....	131
Properties.....	132
Importing an External Database.....	133
Using the Import Wizard.....	133
Importing from.....	135
Integrating the project database with a TwinCAT PLC.....	147

SCREENS AND GRAPHICS.....	148
Working with Screens.....	149
Screens folder.....	149
Screen Group Folder.....	152
Web Pages Folder.....	153
Mobile Access.....	154
Layout.....	158
Using Screen Objects and Animations.....	159
Editing.....	159
Shapes.....	161
Active Objects.....	166
Libraries.....	184
Animations.....	202
Formatting Screen Objects.....	212
Move to Front and Move to Back.....	212
Move Backward and Move Forward.....	213
Group.....	213
Align.....	213
Rotate.....	215
Size.....	216
Fill Color.....	217
Line Color.....	217
Fonts.....	217
ALARMS, EVENTS, AND TRENDS.....	218
Alarms.....	219
Alarm Worksheet Header.....	220
Alarm Worksheet Body.....	223
Saving your alarm history to an external database.....	224
Format of the alarm history.....	225
Events.....	227
Enabling the event logger.....	227
Saving your event log to an external database.....	228
Format of the event log.....	229
Alarm/Event Control object.....	232
Trends.....	238
Converting Trend History Files from Binary to Text.....	239
Converting Trend History Files from Text to Binary.....	240
Creating Batch History.....	241
Setting the Trend Database.....	243
Trend Control object.....	244
About the trend control runtime interface.....	244
Object Properties: Trend Control.....	246
Using the Data Source Text File.....	259
Using the Data Source Database.....	261
Grid object.....	266
Columns dialog.....	267
Data dialog.....	269
Advanced dialog.....	270
BACKGROUND TASKS.....	273
Alarms.....	274
Trends.....	276
Recipes.....	278
Reports.....	280
ODBC.....	282
Math.....	284
Script.....	285
Startup Script worksheet.....	286
Scheduler.....	287
Database/ERP worksheet.....	289
COMMUNICATION WITH OTHER DEVICES.....	293

Drivers.....	294
Main Driver Sheet.....	299
Standard Driver Sheets.....	301
OPC DA 2.05.....	303
OPC UA.....	306
OPC .Net.....	311
OPC XML/DA.....	316
TCP/IP.....	320
DDE.....	322
PROJECT SECURITY.....	324
About security modes.....	325
About security access levels.....	326
Using the security system configuration wizard.....	328
Configuring server settings for security modes.....	331
Extending the LDAP schema to allow saving of security rights.....	334
Creating and configuring groups.....	342
Creating and configuring users.....	346
Managing an existing security system.....	347
Backing up the security system configuration.....	349
Logging on/off.....	351
Blocking or unblocking a user.....	352
Password Protection of Project Files.....	353
AUTOMATIC TRANSLATION.....	355
Adding a language to the Translation Table.....	356
Setting the project's language at startup.....	358
Setting the project's language during runtime.....	359
Disabling translation of selected screen objects.....	360
TESTING AND DEBUGGING.....	361
Debugging from the Output Window.....	362
Debugging from the Database Spy.....	364
Using the LogWin Module.....	365
Using Remote Tools.....	366
Using Remote Database Spy.....	366
Using Remote LogWin.....	366
DEPLOYING AS A WEB APPLICATION.....	368
Introduction to Thin Clients.....	369
Building a Simple Application.....	371
The Underlying Technology.....	379
ISSymbol Control Layer.....	380
Examples of Client/Server Architecture.....	382
Configuring the Data Server.....	386
Configuring a web server to host your project pages.....	389
Installing the web tunneling gateway.....	391
Configuring the Thin Client.....	393
Implementing Security.....	397
Port Usage.....	400
Exercise: Viewing Your Project on the Web.....	401
DOWNLOADING TO A REMOTE DEVICE.....	403
Configuring the Target System.....	404
Configuring the Development Station.....	406
Automatically Running a Project.....	408
DATABASE INTERFACE.....	409
SQL Relational Databases.....	410
Linking the Database Through a Remote DB Provider.....	412
Studio Database Gateway.....	413
Database Configuration.....	418
Configuring a Default Database for All Task History.....	422

Database Troubleshooting.....	423
Appendices.....	426
Using ODBC Databases.....	426
Using Microsoft SQL Server.....	427
Using ORACLE Databases.....	428
Using Microsoft Access Databases.....	430
Using SQL Server CE.....	431
Using Sybase.....	433
Using Microsoft Excel.....	433
Using MySQL.....	436
TROUBLESHOOTING.....	438
General Troubleshooting.....	439
Frequently Asked Questions.....	441
Getting help.....	448
Technical Reference.....	448
Communication Drivers.....	448
License Agreement.....	448
Home Page.....	448
Release Notes.....	448
System Information.....	449
Support Information.....	449
About.....	450
APPENDIX: BUILT-IN SCRIPTING LANGUAGE.....	451
Logic and arithmetic operators.....	452
How to read function descriptions.....	454
Log Message functions.....	456
Trace.....	456
Arithmetic functions.....	457
Abs.....	457
Div.....	457
Format.....	458
GetBit.....	460
Mod.....	461
Pow.....	461
ResetBit.....	462
Round.....	462
SetBit.....	463
Sqrt.....	463
Swap16.....	464
Swap32.....	464
Trunc.....	465
Statistical functions.....	466
Avg.....	466
Max.....	466
Min.....	467
Rand.....	468
Logarithmic functions.....	469
Exp.....	469
Log.....	469
Log10.....	469
Logical functions.....	471
False.....	471
If.....	471
Toggle.....	472
True.....	472

String functions.....	474
Asc2Str.....	474
CharToValue.....	474
CharToValueW.....	475
ClassMembersToStrVector.....	475
NCopy.....	476
Num.....	477
Str.....	477
Str2Asc.....	478
StrCompare.....	478
StrCompareNoCase.....	479
StrFromInt.....	479
StrFromReal.....	480
StrFromTime.....	481
StrGetElement.....	481
StrLeft.....	482
StrLen.....	482
StrLower.....	483
StrRChr.....	483
StrRight.....	483
StrSetElement.....	484
StrStr.....	484
StrStrPos.....	485
StrTrim.....	485
StrTrimAll.....	486
StrUpper.....	486
ValueToChar.....	487
ValueWToChar.....	487
Date & Time functions.....	489
ClockGetDate.....	489
ClockGetDayOfWeek.....	489
ClockGetTime.....	490
DateTime2Clock.....	490
GetClock.....	491
Hour2Clock.....	491
SetSystemDate.....	492
SetSystemTime.....	492
Trigonometric functions.....	493
ACos.....	493
ASin.....	493
ATan.....	493
Cos.....	494
Cot.....	494
Pi.....	495
Sin.....	495
Tan.....	495
Screen functions.....	497
Close.....	497
Open.....	497
OpenPrevious.....	499
ShowInplaceInput.....	500
ShowMessageBox.....	501

Security functions.....	502
BlockUser.....	502
CheckESign.....	502
CreateUser.....	503
ExportSecuritySystem.....	504
GetSecuritySystemStatus.....	504
GetUserFullName.....	505
GetUserNames.....	506
GetUserPwdAging.....	507
GetUserState.....	507
ImportSecuritySystem.....	508
RemoveUser.....	509
SetPassword.....	509
UnblockUser.....	510
Module Activity functions.....	512
AppActivate.....	512
AppIsRunning.....	513
AppPostMessage.....	514
AppSendKeys.....	515
CleanReadQueue.....	515
CloseSplashWindow.....	515
DisableMath.....	516
EnableMath.....	516
EndTask.....	517
ExitWindows.....	517
IsScreenOpen.....	518
IsTaskRunning.....	518
IsViewerInFocus.....	519
KeyPad.....	519
LogOff.....	520
LogOn.....	521
Math.....	521
PostKey.....	522
Recipe.....	522
Report.....	523
RunGlobalProcedureAsync.....	524
RunGlobalProcedureAsyncGetStatus.....	525
RunGlobalProcedureOnFalse.....	526
RunGlobalProcedureOnServer.....	527
RunGlobalProcedureOnTrigger.....	528
RunGlobalProcedureOnTrue.....	529
RunVBScript.....	529
SecureViewerReload.....	530
SendKeyObject.....	530
SetAppPath.....	532
SetViewerInFocus.....	532
SetViewerPos.....	533
ShutDown.....	533
StartTask.....	534
ViewerPostMessage.....	534
WinExec.....	535
WinExecIsRunning.....	537

File functions.....	538
DeleteOlderFiles.....	538
DirCreate.....	538
DirDelete.....	539
DirLength.....	539
DirRename.....	540
FileCopy.....	540
FileDelete.....	541
FileLength.....	542
FileRename.....	542
FileWrite.....	542
FindFile.....	543
FindPath.....	544
GetFileAttributes.....	545
GetFileTime.....	545
GetHstInfo.....	546
GetLine.....	546
HST2TXT.....	548
HST2TXTIsRunning.....	549
LookupContains.....	550
LookupGet.....	550
LookupLoad.....	551
PDFCreate.....	551
Print.....	552
RDFileN.....	553
WebGetFile.....	554
Graphic functions.....	555
AutoFormat.....	555
GetScrInfo.....	555
PrintSetup.....	556
PrintWindow.....	556
ResetDecimalPointsTable.....	558
RGBColor.....	558
RGBComponent.....	558
SaveScreenShot.....	559
SetDecimalPoints.....	560
SetDisplayUnit.....	561
SetTagDisplayUnit.....	562
Translation functions.....	563
Ext.....	563
SetLanguage.....	563
SetTranslationFile.....	564
Multimedia functions.....	565
Play.....	565

System Info functions.....	566
DbVersion.....	566
GetAppHorizontalResolution.....	566
GetAppPath.....	566
GetAppVerticalResolution.....	567
GetComputerIP.....	567
GetComputerName.....	567
GetCursorX.....	568
GetCursorY.....	568
GetDisplayHorizontalResolution.....	568
GetDisplayVerticalResolution.....	569
GetFreeMemoryCE.....	569
GetHardKeyModel.....	570
GetHardKeySN.....	570
GetIPAll.....	571
GetMemoryCE.....	571
GetNetMACID.....	572
GetOS.....	572
GetPrivateProfileString.....	573
GetProductPath.....	573
GetRegValue.....	574
GetRegValueType.....	574
GetServerHostName.....	575
GetTickCount.....	575
InfoAppAlrDir.....	576
InfoAppHSTDir.....	576
InfoDiskFree.....	576
InfoResources.....	577
IsActiveXReg.....	577
IsAppChangedOnServer.....	578
NoInputTime.....	578
ProductVersion.....	579
RegSaveCE.....	579
ReloadAppFromServer.....	580
SaveAlarmFile.....	580
SetAppAlarmPath.....	581
SetAppHSTPath.....	581
SetDateFormat.....	582
SetKeyboardLanguage.....	582
SetRegValue.....	583
SetWebConfig.....	584
SNMPGet.....	585
SNMPSet.....	586
WritePrivateProfileString.....	587
Tags Database functions.....	589
ExecuteAlarmAck.....	589
ForceTagChange.....	589
GetTagValue.....	590
SetTagValue.....	591
Loop functions.....	592
For ... Next.....	592

ODBC functions.....	593
ODBCBeginTrans.....	593
ODBCBindCol.....	593
ODBCCanAppend.....	594
ODBCCanTransact.....	595
ODBCCanUpdate.....	595
ODBCClose.....	595
ODBCCommitTrans.....	596
ODBCDelete.....	596
ODBCExecuteSQL.....	597
ODBCInsert.....	597
ODBCIsBOF.....	598
ODBCIsDeleted.....	598
ODBCIsEOF.....	599
ODBCIsFieldNULL.....	599
ODBCIsFieldNullable.....	600
ODBCMove.....	600
ODBCMoveFirst.....	601
ODBCMoveLast.....	601
ODBCMoveNext.....	602
ODBCMovePrev.....	602
ODBCOpen.....	602
ODBCQuery.....	603
ODBCRollBack.....	604
ODBCSetFieldNULL.....	604
ODBCSetFilter.....	605
ODBCSetSort.....	605
ODBCUnbindCol.....	606
ODBCUpdate.....	606
Email functions.....	608
CnfEmail.....	608
GetStatusSendEmailExt.....	610
SendEmail.....	610
SendEmailExt.....	611
Dial-up functions.....	613
DialError.....	613
DialGetClientIP.....	617
DialGetServerIP.....	618
DialStatus.....	620
DialUp.....	621
DialUpToCE.....	622
FindAllDevices.....	624
FindModem.....	624
HangUp.....	625
PhoneDialUp.....	625
PhoneDisableListen.....	626
PhoneEnableListen.....	626
PhoneHangUp.....	627
PhoneStatus.....	628
ActiveX and .NET Control functions.....	629
XGet.....	629
XRun.....	629
XSet.....	630
Event Logger functions.....	631
SendEvent.....	631
FTP functions.....	632
CnfFTP.....	632
FTPGet.....	633
FTPPut.....	634
FTPStatus.....	635

Database/ERP functions.....	636
DBCursorClose.....	636
DBCursorColumnCount.....	636
DBCursorColumnInfo.....	637
DBCursorCurrentRow.....	638
DBCursorGetValue.....	638
DBCursorMoveTo.....	639
DBCursorNext.....	640
DBCursorOpen.....	640
DBCursorOpenSQL.....	642
DBCursorPrevious.....	643
DBCursorRowCount.....	643
DBDelete.....	644
DBExecute.....	645
DBInsert.....	646
DBSelect.....	647
DBUpdate.....	648
SyncAlarm.....	649
SyncAlarmStatus.....	650
SyncEvent.....	651
SyncEventStatus.....	651
SyncTrend.....	652
SyncTrendStatus.....	653
APPENDIX: VBSCRIPT.....	654
Overview.....	655
VBScript Interfaces in the Software.....	656
Global Procedures.....	657
Graphic Module.....	659
Background Task.....	664
Language Reference.....	667
Operators.....	667
Constants.....	667
Objects and Collections.....	670
Properties.....	670
Statements.....	671
Methods.....	672
Functions.....	673
Keywords.....	673
Errors.....	674
Tips & Tricks.....	677
VBScript Editor IntelliSense.....	677
VBScript Compared to VBA.....	678
Screen Events.....	680
MsgBox and InputBox Functions.....	680
VBScript Procedures.....	680
Creating Constants.....	681
Declaring Variables.....	682
Scope and Lifetime of Variables.....	682
Boolean Tags and Boolean Variables.....	682
Writing Real Values to Integer Tags.....	683
Precedence of VBScript Operators.....	683
Logical Operator NOT.....	684
Using Conditional Statements.....	685
Looping Through Code.....	686
Support for ActiveX Controls.....	688
Windows Embedded Support.....	688

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 chapters:

- This chapter: Describes the purpose, content, and organization of the *User Guide and Technical Reference*. In addition, this chapter contains the following information:
 - Explains the formatting, mouse, and Windows conventions used
 - Lists other publications providing information about IWS
 - Explains how to contact a technical support representative
- **About This Software:** Provides a high-level overview of the product's uses, features, and functions.
- **Installing and Uninstalling:** Provides step-by-step instructions for installing, licensing, starting, and uninstalling InduSoft Web Studio and CEView.
- **Navigating the Development Environment:** Describes the IWS interface (or *development environment*), and explains some basic skills and techniques you must understand before creating a new project.
- **How the Software Works:** Describes the internal structure of IWS, including how data flows through the runtime modules and how these modules are executed.
- **Creating and Configuring a New Project:** Provides step-by-step instructions for creating and configuring a new project.
- **Working with Tags:** 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.
- **Creating Screens and Graphics:** Explains how to use the different IWS development tools to create your project screens and graphics.
- **Configuring Background Tasks:** Explains how to create and configure the different IWS *task* worksheets for your projects.
- **Event Settings:** Describes the logging and event-retrieval features.
- **Communicating 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.
- **Configuring a Security System:** Explains how to set-up and manage a security system for your projects.

- **Testing and Debugging Your Project:** 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.
- **Configuring a Web Solution:** Explains how configure and run your projects on the Web.
- **Downloading to a Remote Device:** Explains how to download, monitor, and debug projects from a remote runtime workstation.
- **Using the Translation Tool:** Explains how to use the Translation Tool to translate the text in your projects from one language to another.
- **Database Interface:** Explains how to connect IWS to compatible databases.
- **Scripting Languages:** Describes IWS's built-in scripting language, as well as the support for VBScript in IWS.
- **Troubleshooting and FAQs:** Provides instructions for verifying projects, describes some common development errors, and explains what to do if you need to contact a support representative.

Related documentation

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

- *Getting Started Manual:* 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 an 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** (or *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 or function is *enabled* (selected). No dot indicates the option or function is *disabled* (not selected).
- **Check boxes** are white squares in which a check *Idlebar* appears or disappears when you click on it with the cursor. Typically, a check indicates the option or function is *enabled* (selected). No check indicates the option or function is *disabled* (not selected).
- **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 Application

InduSoft Web Studio (or simply IWS or Studio) is a powerful, fully integrated software program that enables you to design and build feature-rich *HMI* (Human-Machine Interface) or *SCADA* (Supervisory Control and Data Acquisition) projects for:

- Data acquisition
- Local supervisory stations
- Remote supervisory stations
- Data concentrators on distributed processes
- Data communications with corporate systems

Product Overview

InduSoft Web Studio projects run on microcomputers connected in real-time to machines or processors through programmable controllers, remote I/O devices, and other data-acquisition equipment.

These projects consist of animated operator-interface screens, configurable PLC (programmable logic controller) drivers and other controllable I/O devices, a project tags database, and optional modules such as alarm monitors, logic, trend charts, recipes, schedulers, and a security system. IWS projects interface with industrial I/O systems and other Windows applications in the runtime environment using the following protocols:

- ODBC (*Open Database Connectivity*)
- DDE (*Dynamic Data Exchange*)
- NetDDE (*Network Dynamic Data Exchange*)
- OPC (*Open Connectivity*)
- TCP/IP (*Transmission Control Protocol/Internet Protocol*)

After developing a project, you can run it on your development workstation or download the project to a runtime workstation (using a serial or TCP/IP connection) and run it using InduSoft Web Studio or CEView runtime software. The workstation processes scan data from connected devices according to parameters defined in the project and then react to, display, store and upload the data.

The product consists of two parts:

- The development system software runs on a desktop, laptop, or industrial PC running a currently supported Microsoft Windows desktop or server operating system.
- The runtime system software runs on an operator interface workstation running a currently supported Microsoft Windows desktop operating system or Windows Embedded.

 **Note:** The runtime client for Windows Embedded operating systems (CEView) is often pre-loaded on the HMI. If necessary, you can update the CEView version of the development system software by downloading the current version to the HMI.

Product Features

The InduSoft Web Studio product provides the following features:

- Integrated Windows [development environment](#) with toolbars, dialogs, and menus:
 - Drop-down (pop-up) menus, which you activate by right-clicking on any area of the development environment (Options vary according to context.)
 - Customizable fly-over [toolbars](#)
 - Tasks, objects, and controls organized in a tree-view [explorer](#)
- Full-featured objects and animations (the ability to modify object properties, execute commands, or inset values to tags used to build screens on the fly at runtime):
 - Configurable objects such as buttons, rectangles, ellipse, polygons, lines, and text
 - [Object animations](#) such as bar graphs, color, resizing, position, hide/unhide, rotation, command, hyperlink, and text input/output
 - On-line and historical [alarm list](#) displays
 - On-line and historical [trending](#)
 - [Alignment and distribution tools](#)
 - Background [bitmap](#) layer creation and editing
 - Graphics importation
 - [ActiveX object](#) containers
- On-line remote management and configuration
- Microsoft DNA architecture compliance, with full OPC and XML support
- Web interface enabled, which exports project screens to a "thin" client through the Internet/intranet and by exchanging data on-line through the TCP/IP protocol
- [Symbols library](#) with more than 100 pre-made objects, such as pushbuttons, meters, sliders, switches, text and numeric displays, LED-style indicators, pipes, bumps, icons, vehicles, valves, frames, motors, gauges, and common controls
- Debugging tools:
 - [Database Spy window](#) to monitor/force tag values and execute functions
 - [LogWin module](#) to record OPC, DDE, and TCP/IP transactions, modules activation, trace tags, and so forth
 - Cross-referencing to locate tags throughout the project
 - On-line system and network diagnostics
- Powerful and flexible tag database (Boolean, Integer, Real, and String tags), array tags, classes, and indirect tag-pointers
- Open architecture with API exchanges and tag values with external software
- [Translation editor](#), which enables you to translate a project into several different languages, and switch between them while the runtime system is online
- [TCP/IP](#) client and server modules to exchange tag values and configure redundancy systems
- More than 200 [direct communication drivers](#) for different devices (such as PLC) from several manufacturers; such as Allen-Bradley, Siemens, GE-Fanuc, as well as standard protocols such as MODBUS RTU/ASCII, DeviceNet, Profibus, Interbus, and so forth
- Full integration with PC-based control packages (imports tags database) such as ISaGRAF, SteepleChase, Think&Do, OpenControl, FP Control and ASAP.
- OPC Server and OPC Client with integrated [OPC Browser](#)
- Screen and object password-protected runtime [security](#) (256 levels)
- Logical expressions and a [scripting language](#) with more than 200 functions
- [Recipe](#) and [Report](#) (ASCII, UNICODE, and RTF formats) builders integrated into the product
- Event [scheduler](#) based on date, time, or data condition (100ms resolution)

- Multi-layer project, which means modular worksheets and screens can be merged easily to other projects
- [Dial-Up functions](#) to trigger, monitor, and hang-up a dial-up connection with the RAS Server of remote stations
- Functions to [send e-mail](#) from IWS (or CEView)
- Real-time project documentation
- Screen [resolution converter](#)

 **Note:** IWS provides different product types for each level of project responsibility. However, IWS does not support some features in certain product types. For more information, see [About target platforms and product types](#).

How the Software Works

Understanding the Internal Structure and Data Flow

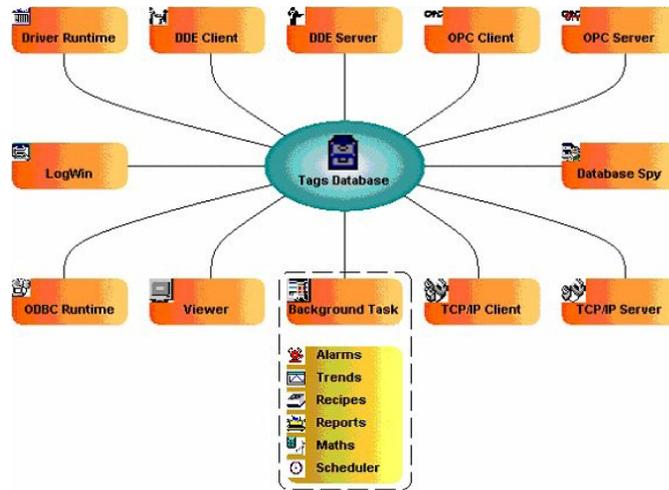
IWS runtime environment runs on an operator interface workstation (running Windows 2K/XP/Vista/CE) and consists of the following modules or *threads* (program elements that can execute independently of other program elements):

- **Background Task** (a supervisory task): 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** (debugging tool):
 - 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** (debugging tool): 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**: Manages TCP/IP communication with a remote TCP/IP Server module (from IWS), according to parameters configured in the *TCP/IP Client* worksheets.
- **TCP/IP Server**: Manages TCP/IP communication messages with a remote TCP/IP Client module (from IWS).
- **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 IWS.

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).

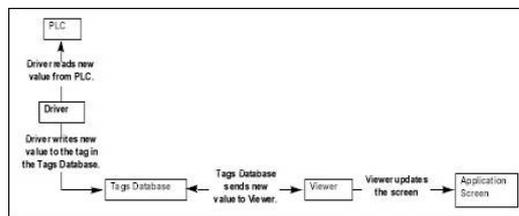
Note: Tags are variables (such as communication points in field equipment, calculation results, alarm points, and so forth) that are used in screens and worksheets. For detailed information about tags, tag values, and tag properties see *Working with Tags*.



Data Flow

Each IWS 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.



Data Flow Example

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.

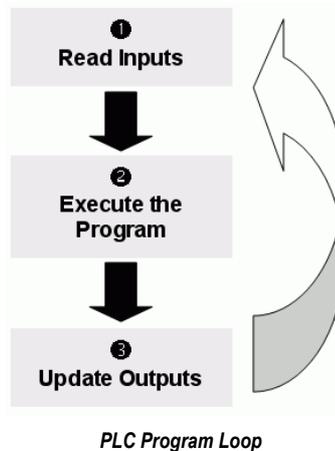
Important: The *Viewer* module will update an object only when (at least) one of the object's tag values change. If you configure an object animation (such as [Text Data Link](#)) with a function that does not require a tag (for example, `NoInputTime()`), the *Viewer* will not update the object because there are no tags associated with that object.

The architecture of IWS 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/Switching Modules

IWS 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 other 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:



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.

IWS only has one process: `Studio Manager.exe`. When you execute a runtime project, the `Studio Manager.exe` 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 the runtime.

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 IWS 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.

IWS 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 IWS 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 IWS.

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.

IWS 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 IWS): 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 IWS): Specifies the maximum amount of time each IWS thread (except the Background Task) can remain inactive.



Caution: We *strongly* recommend that you do not modify 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 IWS 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).

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

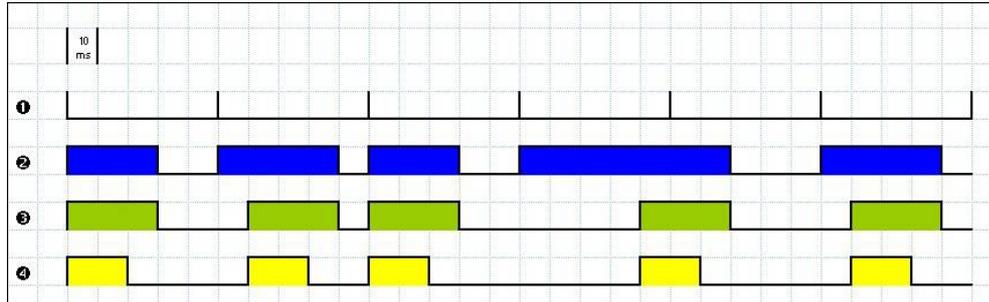
```
[TimeSlice]
UniDDEClient=100
Driver=10
OPCClient=10
OPCServer=10
TCPClient=200
TCPServer=200
Viewer=200
```



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.

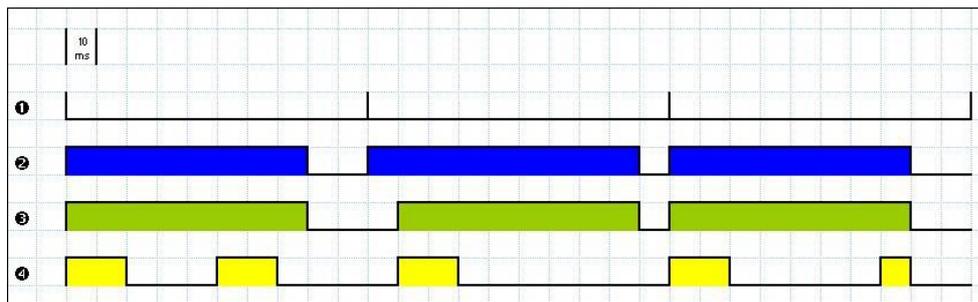
In this example, IWS generates a Period message every 50ms (signal **1**). When IWS generates this message, its thread becomes active and remains active until the specified TimeSlice time period (from IWS) expires. The thread then remains inactive until IWS generates the next Period message (signal **1**).

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 time 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/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.

System requirements

These are the minimum system requirements to install and run the InduSoft Web Studio software.

-  **Note:** The requirements described below are based on typical projects. Depending on your specific project, the requirements may vary:
- "Windows Embedded-compatible devices" includes a wide variety of processors and feature sets, from smartphones to industrial displays. Consult your vendor for the specific hardware requirements to run your project on these devices.
 - Some of the items listed as optional may be mandatory depending on your project. For instance, if you need to exchange data with a PLC via a serial interface, then the computer must have a serial COM port.

Development / Project Server / Project Client

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

- A Windows-compatible computer or Windows Embedded-compatible device with a standard keyboard, mouse, and SVGA-minimum display
- A Windows or Windows Embedded operating system that is currently supported by Microsoft, which at this time includes:
 - Microsoft Windows XP Service Pack 3 or later
 - Microsoft Windows Vista Service Pack 1 or later
 - Microsoft Windows 7, all versions
 - Microsoft Windows Server 2003 Service Pack 2 or later
 - Microsoft Windows Server 2008, all versions
 - Microsoft Windows XP Embedded Service Pack 3
 - Microsoft Windows Embedded Standard 7 (2009)

-  **Tip:** We recommend "Professional" and "Ultimate" editions over "Home" and "Media Center" editions, because they include Internet Information Services (IIS). You will need IIS if you want to deploy your IWS project as a web application.

- Microsoft Internet Explorer 6.0 or later
- Minimum of 500 MB free hard drive space
- Ethernet adapter or wireless networking
- CD-ROM drive (optional, to install the application; it can also be downloaded from our website)
- USB port (optional, to be used with hard key licensing)
- Serial COM ports and adapters (optional, to be used for direct communication with PLCs and other devices)

Any computer that has the full InduSoft Web Studio software installed can also run as a project server and/or a project client. That includes Windows XP Embedded and Windows Embedded Standard devices, but in most cases, if you do not plan to do project development, it is more practical to install EmbeddedView on these devices because it has a smaller footprint and can be installed and managed remotely.

You cannot install the full InduSoft Web Studio software on Windows Embedded Compact devices.

Project Server – Embedded

To run as a project server — that is, to host the project runtime — using EmbeddedView or CEView, you must have:

- A Windows Embedded-compatible device
- A Windows Embedded 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 (2009)
- Microsoft Windows Embedded Compact (formerly known as Windows CE or Windows Mobile), version 5.0 or later
- Minimum of 500 MB free hard drive space
- Ethernet adapter or wireless networking
- USB port (optional, to be used with hard key licensing)
- Serial COM ports and adapters (optional, to be used for direct communication with PLCs and other devices)

Project Client – Embedded

To run as a project client using EmbeddedView or CEView, you must have:

- A Windows Embedded-compatible device with a mouse or touchscreen input
- A Windows Embedded 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 (2009)
 - Microsoft Windows Embedded Compact (formerly known as Windows CE or Windows Mobile), version 5.0 or later
- Ethernet adapter or wireless networking

Project Client – Thin

To run as a project client using either the Secure Viewer program or the browser-based Thin Client, you must have:

- A Windows-compatible computer or Windows Embedded-compatible device with a mouse or touchscreen input
- A Windows desktop, server, or embedded operating system that is currently supported by Microsoft, which at this time includes:
 - Microsoft Windows XP Service Pack 3 or later
 - Microsoft Windows Vista Service Pack 1 or later
 - Microsoft Windows 7, all versions
 - Microsoft Windows Server 2003 Service Pack 2 or later
 - Microsoft Windows Server 2008, all versions
 - Microsoft Windows XP Embedded Service Pack 3
 - Microsoft Windows Embedded Standard 7 (2009)
 - Microsoft Windows Embedded Compact (formerly known as Windows CE or Windows Mobile), version 5.0 or later
- Microsoft Internet Explorer 6.0 or later
- Ethernet adapter or wireless networking

Installing the Software

InduSoft Web Studio provides development tools for all IWS projects, and it can be installed on a PC running Microsoft Windows XP, Windows Vista, or Windows 7 operating system. For more information, see [System Requirements](#).

You can install the development application either from the web download or from the InduSoft Web Studio installation CD. For projects running on Windows Embedded target systems, you can use the development application to download CEView (the runtime engine) to the target system via serial or TCP/IP link.

The IWS installation program creates directories as needed, copies files to your hard drive, and creates the InduSoft Web Studio icon on your Windows desktop.

 **Note:**

- You must have Administrator privileges on your PC in order to install or uninstall the development application.
- You must uninstall an older version of the development application (or move it to a different directory) before installing a new version. Also, you cannot install the same version of the development application in two different paths on the same PC.

The instructions for installing InduSoft Web Studio and CEView are provided in the following two sections.

Installing the Development Application on Your Windows PC

To install the IWS development application from the installation CD:

1. Turn on your PC and be sure that no other programs are running.
2. Insert the installation CD into your PC's CD-ROM drive.

Internet Explorer should run automatically and show the CD's welcome screen. If it does not — for example, if you have the Autorun feature turned off in your Windows settings — then you can manually show the screen by using Windows Explorer to locate and open the file `D:\InduSoft.htm`.

3. In the welcome screen, select the product that you want to install.

Internet Explorer will ask if you want to run or save the installer.

4. Click **Run**.

The product's installation wizard will begin.

5. Follow the wizard's instructions to proceed with the installation.
6. When the installation is finished, select **Yes, I want to restart my computer now** and then click **OK**.

After your PC has restarted, you can run the development application. See [Starting the Software](#) for instructions.

 **Note:** When you install the development application, Microsoft .NET Framework 2.0 and some other utilities are also installed to support the features of IWS. Your PC may have later versions of the .NET Framework already installed, but there is no reason for concern because multiple versions of the .NET Framework should not conflict with each other. You can see which versions are installed on your PC by opening the *Add or Remove Programs* control panel (**Start > Control Panel > Add or Remove Programs**).

For more information about Microsoft .NET Framework, see [Database Appendix A: Using ODBC Databases](#).

Installing CEView on Your Windows Embedded Device

CEView is the runtime engine for IWS projects on Windows Embedded devices. CEView must be installed on your device *before* you send your project to it.

Where the Files Are Located

Given the nature of Windows Embedded devices, each combination of OS version and device processor has its own build of CEView. All of these builds are located in the following directory:

```
[...]\InduSoft Web Studio v7.0\Redist\
```

The build for your specific device is located in the following directory:

```
[...]\InduSoft Web Studio v7.0\Redist\version\processor\
```

...where:

- **version** is the version of the operating system on the device where CEView will be installed:
 - The \WinCE 5.0 folder contains the files for Windows CE and Windows Mobile 5.0 or later; and
 - The \WinEmbedded folder contains the files for Windows XP Embedded and Windows Embedded 7.
- **processor** is the processor used by your Windows CE device. We provide a CEView runtime for every processor that is currently supported by the Windows CE operating system (e.g., Pocket2003-ArmV4, ArmV4i, x86). For more information, consult the manufacturer's documentation for the device.

To install the files on your device, use the [Remote Management](#) tool in the development application.

Installing via TCP/IP (Ethernet)

 **Tip:** We recommend using TCP/IP whenever possible.

To install CEView on a Windows Embedded device via a TCP/IP (Ethernet) connection:

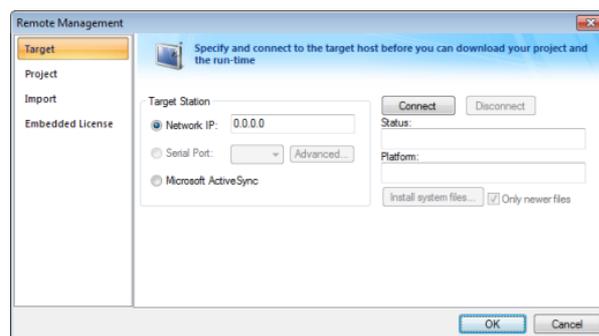
1. Make sure your Windows Embedded device is connected to your network.
2. Turn on the device.

The *Remote Agent* dialog should open automatically. If it does not, then you must manually install the file **CEServer.exe** on the device. The file is located here:

```
[...]\InduSoft Web Studio v7.0\Redist\version\processor\CEServer.exe
```

There are different ways to install the file — for example, you can use Microsoft ActiveSync (for Windows 2000 and Windows XP) or Windows Mobile Device Center (for Windows Vista) to communicate directly with the device or you can map the device as a shared folder on your PC. For more information about copying and executing files, consult the manufacturer's documentation for the device.

3. In the *Remote Agent* dialog, click **Setup** and then configure the communication settings for the device's network connection. In particular, note the IP address of the device.
4. Start the development application on your PC.
5. On the **Home** tab of the ribbon, in the **Remote Management** group, click **Connect**. The *Remote Management* dialog is displayed:



Remote Management dialog

6. In the *Target System* group-box, select **Network IP** and then type the IP address of the device.
7. Click **Connect** to establish a connection between the development application and the device.

If the connection is successful, then the device's specifications will be displayed in the **Platform** text-box.

8. Click **Install System Files** to download the CEView files to the device.
9. When the installation is completed, click **Disconnect**.

For more information about downloading and running finished projects on the Windows Embedded device, please see [Remote Management](#).

Installing via Microsoft ActiveSync

To install CEView on a Windows Embedded device via Microsoft ActiveSync:

1. Make sure that Microsoft ActiveSync (for Windows 2000 and Windows XP) or Windows Mobile Device Center (for Windows Vista) is installed on your PC.
2. Turn on the Windows Embedded device and connect it to your PC. Most devices should be able to connect via USB.
3. Start the development application on your PC.
4. On the **Home** tab of the ribbon, in the **Remote Management** group, click **Connect**. The *Remote Management* dialog is displayed.
5. In the *Target System* group-box, select **Microsoft ActiveSync**.
6. Click **Connect** to establish a connection between the development application and the device.
If the connection is successful, then the device's specifications will be displayed in the **Platform** text-box.
7. Click **Install System Files** to download the CEView files to the device.
8. When the installation is completed, click **Disconnect**.

 **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 IWS. The utility is located at:

```
[...]\InduSoft Web Studio v7.0\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 doesn't matter where on the device the file is located. (For more information about copying and executing files, consult the manufacturer's documentation for the device.) When this is done, try again to use the Remote Management tool to connect to the device.

For more information about downloading and running finished projects on the Windows Embedded device, please see [Remote Management](#).

Starting the Software

To run InduSoft Web Studio:

- Double-click the InduSoft Web Studio v7.0+SP1 icon on the desktop; or
- Choose **Start > All Programs > InduSoft Web Studio v7.0 > InduSoft Web Studio v7.0**.

 **Tip:** You can run the IWS development environment under any video setting. However, we recommend that you configure the video settings to a resolution of 800x600 (or higher) and use more than 256 colors for a more pleasing environment. The project resolution (screen size) is independent of the operating system resolution.

Uninstalling the Software



Caution: Before starting the uninstall procedure, be sure to back-up any program files you may find useful later. Also, be certain that you have a current (or newer) version of the IWS installation CD or diskettes so you can re-install the software later if necessary.

If you find it necessary to remove IWS from your system, follow these instructions:

1. From the Windows task bar, select **Start > Settings > Control Panel** to open the Control Panel.
2. Double-click on the **Add/Remove Programs** icon in the *Control Panel* window.
3. When the *Add/Remove Programs Properties* dialog displays, select InduSoft Web Studio from the list and click **Add/Remove**.
4. When the *Confirm File Deletion* dialog displays, click **Yes**.
The *Uninstall Shield Wizard* and the *Remove Programs from Your Computer* dialogs display.
5. When the *Uninstall successfully completed* message displays and the **OK** button becomes active, click **OK**.
Verify that InduSoft Web Studio is no longer listed in the *Add/Remove Programs Properties* dialog.
6. Click the **Cancel** button or the close button () to close the *Add/Remove Programs Properties* dialog, then close the *Control Panel* window.
7. Open the *Windows Explorer* and browse to IWS program directory.
8. Verify that all of the IWS files and folders were deleted. (You must manually delete any that remain.)



Note: The uninstall tool cannot delete files you created or modified in your IWS projects folder. You must have administrator privileges to uninstall (and install) InduSoft Web Studio.

Licensing

Licensing

Protection Types

InduSoft Web Studio and CEView support the following protection types:

Hardkey

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.

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



Caution: 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.

Softkey

When you install InduSoft Web Studio or CEView, the program generates a unique code called a *Site Code*. You can send this site code to your software vendor, who will then generate a license code called a *Site Key* to match your site code. The site key installs the InduSoft Web Studio or CEView license on your computer or Windows Embedded device.



Note: When you use a softkey, IWS records the license in the computer's (or Windows Embedded device's) non-volatile memory. If this device is damaged, you will lose the license.

License Settings

Both hardkey and softkey licenses set the following parameters:

Version

The overall [version](#) of the IWS software, e.g., 6.1. (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 supported). 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 runtime 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](#).

Importers

The list of third-party applications that can be handled by the [Import Wizard](#).

Thin Clients

The number of [Thin Clients](#) that can connect simultaneously to the server. One connection is included with every license. Contact your vendor to purchase additional connections.

Secure Viewers

The number of [Secure Viewer](#) clients that can connect simultaneously to the server. One connection is included with every license. Contact your vendor to purchase additional connections.

SMA Clients

The number of [Studio Mobile Access](#) (SMA) clients that can connect simultaneously to the server. One connection is included with every license. Contact your vendor to purchase additional connections.

Execution Modes

InduSoft Web Studio and CEView support the following execution modes:

Execution Mode	IWS	CEView
Evaluation Mode	Y	N
Demo Mode	Y	Y
Licensed for Engineering Only	Y	N
Licensed for Runtime Only	Y	Y
Licensed for Engineering + Runtime	Y	N

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, 5 hours a day for 8 days, 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 terminates and IWS automatically converts to and runs in Demo Mode (see following description) until you install a valid license (*Hardkey* or *Softkey*). You cannot reactivate Evaluation Mode, even if you uninstall and then reinstall the product on your computer.

 **Note:** Every version of IWS has an evaluation period that is independent of every other version. For example, if your IWS version 6.1 evaluation period expires and you are running in Demo Mode because you have not installed a license, when you install IWS version 6.1 on the same computer, the newer version will begin its own 40-hour evaluation period and the 6.1 version will continue running in Demo Mode only.

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 workbench options for an unlimited time.

This mode also allows you to execute the runtime tasks and debugging tools (*Database Spy*, *Output* window, and *LogWin* module) for 24 hours continuously. After the 24-hour period these tasks shut down, but you can restart them again and run for another 24 hours, and so on. You can use this license for development and testing only.

Licensed for Runtime Only

Enables you to run all runtime 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 engineering tools, runtime tasks, and debugging tools (*Database Spy*, *Output* window, and *LogWin* module) for an unlimited period of time.

 **Note:** The Remote Management tool (**Connect** on the Home tab of the ribbon) is always available, regardless of 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.

Product Versions

InduSoft Web Studio and CEView should both have the same version number, which uses the syntax **x.y** +**SPww** (for example, InduSoft Web Studio v7.0+SP1 and CEView v7.0+SP1), where:

- **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.

Both InduSoft Web Studio and CEView can execute projects built in previous versions of the product. However, older versions of IWS and CEView cannot execute projects built or modified in newer versions of the product.

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

Installing a New Hardkey License

To install a new hardkey license for InduSoft Web Studio or CEView:

1. Install IWS or CEView according to the instructions provided earlier in this chapter.
2. On the computer where you've installed the software, connect the hardkey to the parallel port (LPT1) or USB interface.



Caution: Be careful when you connect or remove a hardkey on a parallel port. We strongly recommend that you turn off the computer and disconnect it from the power supply before connecting or removing the hardkey.

3. Run the software.

If the software recognizes the hardkey, then it will run normally without any alert messages. However, if it does not, then try the following:

- For IWS, use the Protection Manager utility to make sure the software is set to check for a hardkey. (See [Upgrading the Current Hardkey License](#) on page 41.)
- For 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 on the Windows Embedded device to diagnose the problem. (See [Installing or Upgrading a CEView License \(Locally\)](#) on page 46.)

Please note that not all Windows Embedded devices can recognize the USB hardkey technology used by InduSoft; our internal testing has shown 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.

Upgrading the Current Hardkey License

Note: These instructions only apply to upgrading a hardkey license for InduSoft Web Studio. To upgrade a license for CEView, see [Installing or Upgrading a CEView License \(Locally\)](#) on page 46 or [Installing or Upgrading a CEView License \(Remotely\)](#) on page 49.

To upgrade your current hardkey license for InduSoft Web Studio:

1. Close all IWS development and runtime modules and then exit the application.
2. Make sure the hardkey is connected to the parallel port (LPT1) or USB interface on the computer where you installed InduSoft Web Studio.
3. Choose **Start > All Programs > InduSoft Web Studio v7.0 > Register** to launch the Protection Manager.
4. When the *Protection Manager* dialog displays, select **Hardkey** in the *Protection Type* section and then click **Check**.



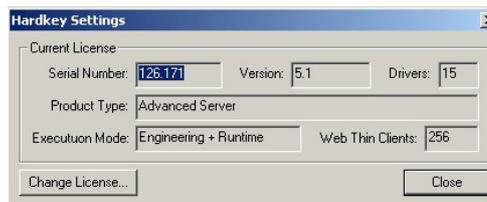
Protection Manager: Select Hardkey

- If you do not have a valid hardkey connected to the computer's parallel port (LPT1) or USB interface, the following error message displays:



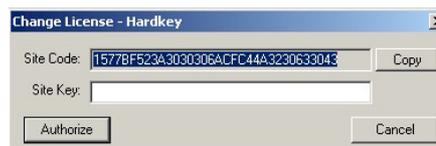
No Hardkey

- If you do have a valid hardkey connected to the computer's parallel port (LPT1) or USB interface, the *Hardkey Settings* dialog displays, which contains the current license settings recorded on the hardkey.



Checking the Hardkey Settings

5. Click the **Change License** button to open the *Change License – Hardkey* dialog:



Change License Dialog

6. Copy the code from the **Site Code** text box and send it to your software vendor.

7. Your software vendor should send back a Site Key to match the Site Code. Type this site key into the **Site Key** field of the *Change License – Hardkey* dialog and then click the **Authorize** button.

You will be prompted to confirm the operation. If the program accepts (validates) your site key, the following message displays:



Register: Successful Completion

Note: If your new Site Key is not valid, an error message displays. If this happens, double-check that you entered the Site Key correctly. If you entered the key correctly and still receive an error message, contact your software vendor for assistance.

You can upgrade any license setting (*ProductType*, *Execution Mode*, or *Number of Thin Clients*) simultaneously supported by the server, or upgrade the software version that is being supported currently. The upgrade cost will depend on your current license settings and the settings of the upgrade license.

Installing a New Softkey License

Note: These instructions only apply to installing a softkey license for InduSoft Web Studio. To install a license for CEView, see [Installing or Upgrading a CEView License \(Locally\)](#) on page 46 or [Installing or Upgrading a CEView License \(Remotely\)](#) on page 49.

Also, you must have Administrator privileges on the computer on which you are installing or modifying a softkey license.

To install a new softkey license for InduSoft Web Studio:

1. Install InduSoft Web Studio according to the instructions provided earlier in this chapter.
2. Launch the Protection Manager by choosing **Start > All Programs > InduSoft Web Studio v7.0 > Register**.
3. Select **Softkey** in the *Protection Type* group, and then click **Check**.



Protection Manager: Softkey

Note: If you already have a hardkey license installed on your computer, then you will be asked to confirm the change of protection type.

The *Softkey Settings* dialog displays:

- If you already have a valid InduSoft Web Studio softkey license installed, then the current license settings display.
- If you have not previously installed a license on your computer, then the **Status** text box displays a "License not found" message.



Checking the Softkey Settings

4. Click the **Change License** button on the *Softkey Settings* dialog.
5. When the *Change License – Softkey* dialog displays, copy the code information from the **Site Code** text box and send it to your software vendor.



Change License: Softkey

Your software vendor will send back a Site Key that matches this Site Code. Type the Site Key into the **Site Key** field of the *Change License – Softkey* dialog and then click the **Authorize** button.

You will be prompted to confirm the operation. If the program accepts (validates) your Site Key, the following message displays:



Successful Site Key Installation

 **Note:** If your new Site Key is not valid, an error message displays. If this happens, double-check that you entered the site key correctly. If you entered the key correctly and still receive an error message, contact your software vendor for assistance.

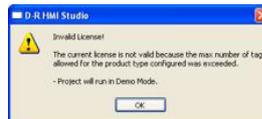
6. Close the Protection Manager and run IWS.

Invalid Licenses

When you try to run InduSoft Web Studio with an invalid license, you will receive a specific warning message that gives you the information you need to resolve the issue. Examples:



Version of software higher than license version



Selected Target System is not supported by the license

Installing or Upgrading a CEView License (Locally)

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

You can register a CEView license on your Windows Embedded device by using the Remote Agent utility on the device itself. This works for both hardkey and softkey licenses.

To install a new (or upgrade an existing) CEView softkey license (locally):

1. Download the *Remote Agent* program (**CEServer.exe**) into the **\Non-Volatile Folder** path of the Windows Embedded device. The **\Non-Volatile Folder** must retain this data after you reboot the Windows Embedded device.

 **Note:** The **\Non-Volatile Folder** path can vary with each Windows Embedded device manufacturer.

After installing InduSoft Web Studio on the Win2K/XP/Vista computer, the *Remote Agent* program file (**CEServer.exe**) is stored in the following path:

```
[...]\InduSoft Web Studio v7.0\Redist\version\processor\Bin\CEServer.exe
```

...where:

- **version** indicates the operating system version.
- **processor** is the Windows Embedded device processor type (for example, **x86**).

 **Note:** In some Windows Embedded devices, the **\Non-Volatile Folder** points to a FlashCard memory that is connected to the device. Also, before downloading the *Remote Agent* program (**CEServer.exe**) to your Windows Embedded device, be sure it is not already loaded in the **\Non-Volatile Folder**.

 **Tip:**

- There are two ways to download the *Remote Agent* program (**CEServer.exe**) to a Windows Embedded device:
 - You can use the Microsoft ActiveSync® utility to download/upload files from a Win2K/XP/Vista station to a Windows Embedded device. You can download ActiveSync from the Microsoft Web site at no charge.
 - You can use the following command syntax to map a shared folder from a Win2K/XP/Vista computer to most Windows Embedded devices:

```
net use [LocalName] [RemoteName] [/user:UserName]
```

After executing this command successfully, open a *Command Prompt* window and use a **COPY** command to copy files to the Windows Embedded device.
- We *strongly* recommend that you configure the Windows Embedded device to execute the *Remote Agent* program automatically when you power on the Windows Embedded device. See the Windows Embedded device manufacturer's documentation for information about how to configure the *Startup* program on the device.

- If the *Remote Agent* program (CEServer.exe) does not start automatically when you power on the Windows Embedded device, you can run it manually from the *\Non-Volatile Folder*.



Remote Agent Dialog

- If you are upgrading a hardkey license, make sure the hardkey is connected to the device.
- From the *Remote Agent* dialog, click the **Setup** button to open the *Setup* dialog:



Setup Dialog

- Click the **License** button to open the *License* dialog:



License Dialog

- Click the **Change License** button to open the *Change License* dialog:



Change License Dialog

- Copy the site code information (provided in the **Site Code** field) and send it to your software vendor.
- Your software vendor will send back a Site Key that matches this site code. Type the Site Key into the **Site Key** field on the *Change License* dialog, and click the **Authorize** button.

If the Site Key is accepted (validated), the following message displays:



Successful Site Key Installation

 **Note:** If the new site key is not validated, an error message displays. If this happens, double-check that you entered the site key correctly. If you typed the key correctly and get an error message, contact your software vendor for further assistance.

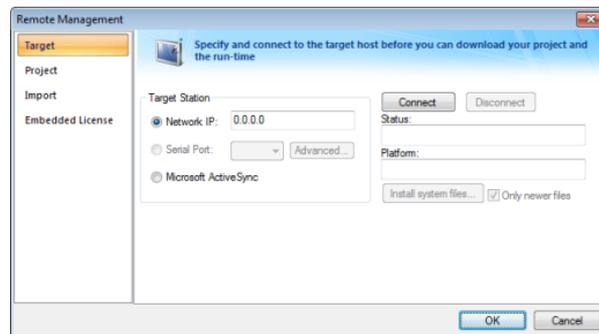
Installing or Upgrading a CEView License (Remotely)

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

You can register a CEView license on your Windows Embedded device by using IWS to send the license to the device. This works for both hardkey and softkey licenses.

To install a new (or upgrade an existing) CEView license (remotely):

1. Perform the four first steps described in [the previous section](#).
2. In the *Setup* dialog, specify the **Device Connection** type by clicking (enabling) the **Serial Port** or **TCP/IP** radio button. (If you enable **Serial Port**, you also must select a port from the combo-box list). Click **OK** to close the dialog.
3. Connect your development workstation to the Windows Embedded device using either a serial or TCP/IP link.
4. Run the IWS development application.
5. On the Home tab of the ribbon, in the Remote Management group, click **Connect**. The *Remote Management* dialog is displayed:



Remote Management dialog

6. Specify a target system by clicking one of the following radio buttons in the *Target System*:
 - **Network IP**, and then type the IP address into the field provided
 - **Serial Port**, and then select a port from the combo-box list provided
 - **Microsoft ActiveSync**

Note: You cannot upgrade a CEView hardkey license via Microsoft ActiveSync.

7. When the **Connect** button becomes active, click the button to connect to the Windows Embedded device on which the *Remote Agent* is running. (If you select **Network IP**, then you must also enter the IP address in the text box provided.)

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

The **Status** field must display the following message: **Connected to CEView version**

8. Select the **Embedded License** tab to see which license settings are currently installed on your Windows Embedded device.
9. From the *License Codes* section, copy the information from the **Site Code** field and send it to your software vendor.
 - Your software vendor will send you a Site Key that corresponds to this Site Code. Type this site key into the **Site Key** field.
 - Click the **Send** button to send the code to the *Remote Agent* running on the Windows Embedded device.

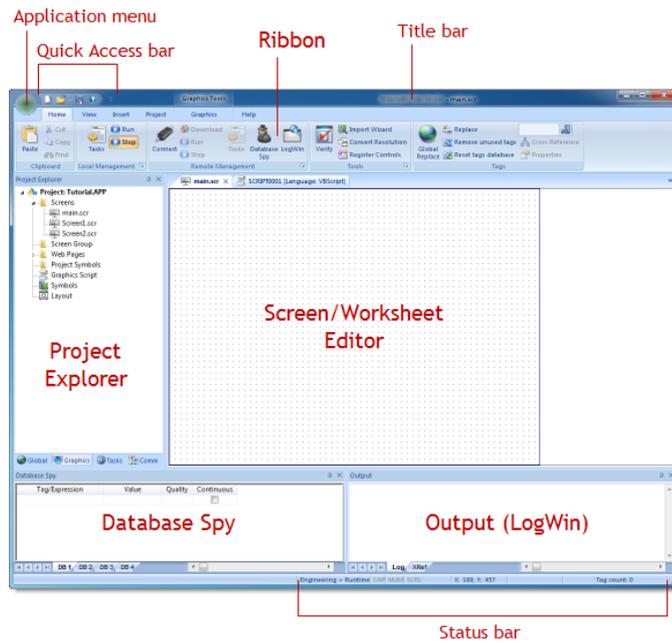
The *Remote Agent* program will attempt to install the new license using the site key sent from the Remote Management tool. If the site key is accepted (validated), then a confirmation message is displayed.

 **Note:** If the new site key is not valid, an error message will display. If this happens, double-check that you typed the Site Key correctly. If you entered the Site Key correctly and still receive an error message, contact your software vendor for further assistance.

 **Caution:** After sending the license to 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.

The Development Environment

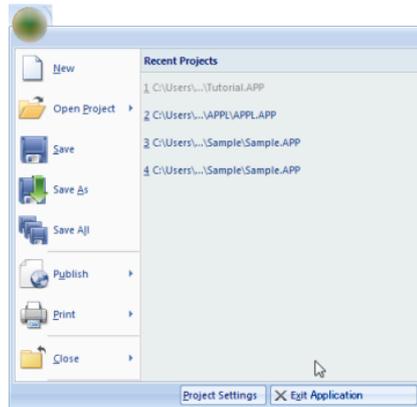
InduSoft Web Studio incorporates a modern, Ribbon-based Windows interface to provide an integrated and user-friendly development environment.



The IWS 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

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:

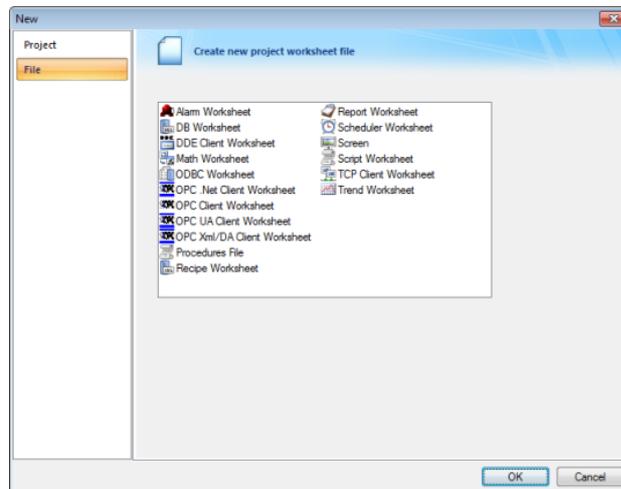
- **File** tab: Select this tab to create new worksheets or screens for an open project.
- **Project** tab: Select this tab to create a new project.

Instructions for creating new files and projects follow.

Creating a New File

To create a new worksheet or screen:

1. Click the **File** tab.



New 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](#).

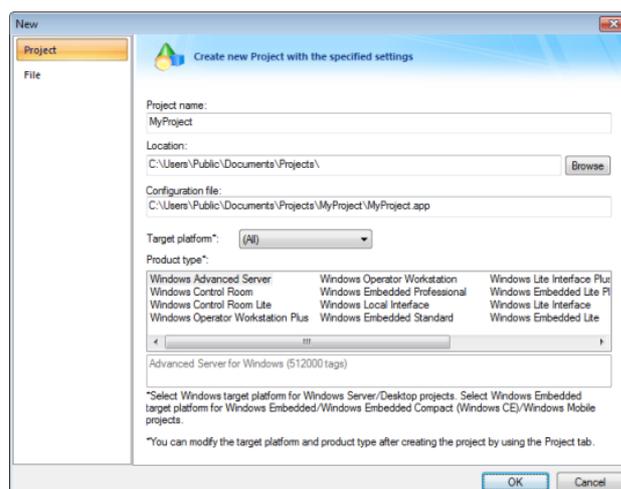
Note: When you add an I/O driver to the project, an associated option allows you to open a new [driver worksheet](#). You also can create new screens or worksheets by right-clicking on the folder in the [Project Explorer](#) and selecting the **Insert** option from the shortcut menu.

Note: Worksheets for [DDE Client](#) and [ODBC](#) are not available for Windows CE projects, because they are not supported by the Windows® CE operating system.

Creating a New Project

To create a new project:

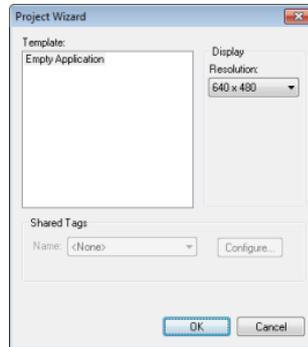
1. Click the **Project** tab.



New Project tab

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

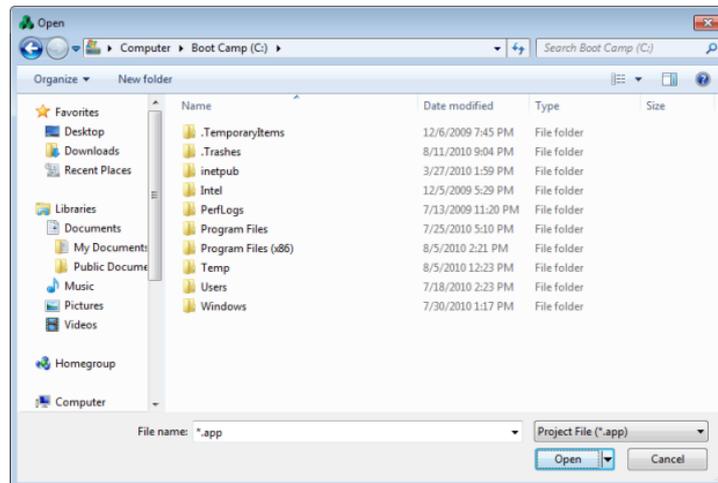
6. In the **Resolution** box, select a screen resolution. If you select Custom, then also type the width and height in pixels.
7. To share tags with another PC-based control application, select the application type from the list and click the **Configure** button. (Each type has its own configuration 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).

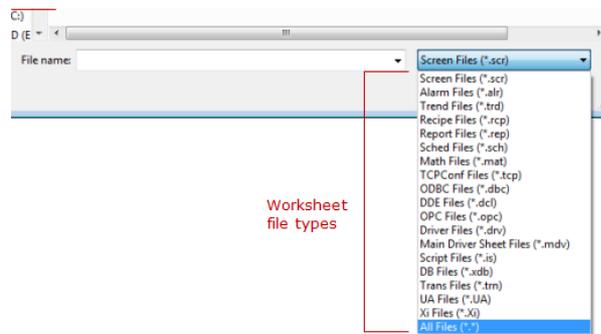


Open dialog

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

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.

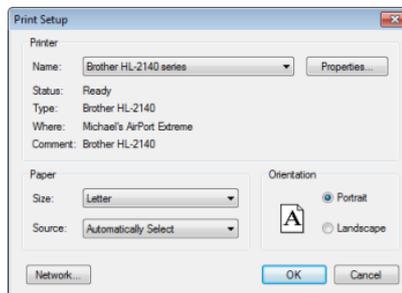
 **Note:** The **Zoom Out** button becomes active after you **Zoom In**.

- 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

 **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**.

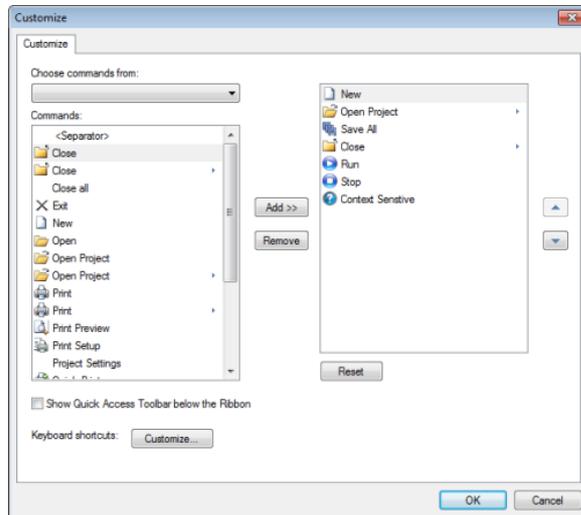
Add a command to the Quick Access Toolbar

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.



Customize Quick Access Toolbar dialog

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.

Ribbon

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.



The Ribbon interface

Home tab

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



Home tab of the ribbon

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](#).
- **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.
- **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.



View tab of the ribbon

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.



Insert tab of the ribbon

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.



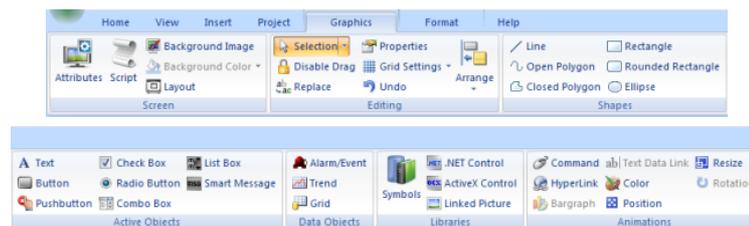
Project tab of the ribbon

The tools are organized into the following groups:

- **Settings:** Configure the general [project settings](#), and also set the project to [run as a Windows service](#).
- **Security System:** Enable and configure the [project security system](#).
- **Web:** Configure the project to accept connections from [thin clients](#) and [mobile devices](#), and also configure outgoing [email](#) and [FTP](#).

Graphics tab

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



Graphics tab of the ribbon

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

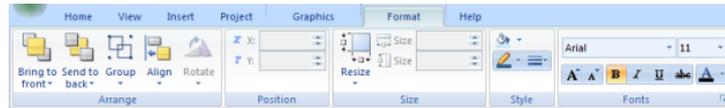
The tools are organized into the following groups:

- **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 picture 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.



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.

Help tab

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



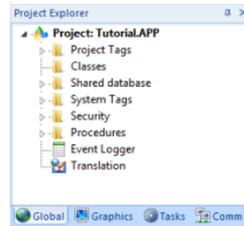
Help tab of the ribbon

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 components that make up your project and presents them in an expandable tree-view.



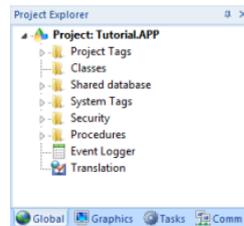
The Project Explorer

Click the Expand icon **▸** or double-click the folder to view the contents of the folder. Click the Collapse icon **▹** to close the folder.

If you right-click on any component in the *Project Explorer*, a shortcut menu is displayed with options for that component.

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 security and UI translation.



Global tab of the Project Explorer

The folders on the **Global** tab are described on the following pages:

- **Project Tags** contains tags you create during project development (such as screen tags or tags that read from/write to field equipment).
- **Classes** contains compound tags, called *class* tags, created to associate a set of values (rather than a single value) with an object.
- **Shared Database** contains tags that were created in a PC-based control program and then imported into the project tags database.

For example you can import SteepleChase tags into your project so that it can read/write data from a SteepleChase PC-based control product.

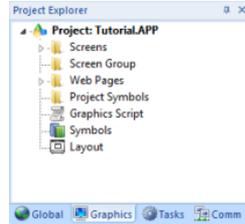
- **System Tags** contains predefined tags with predetermined functions that are used by the project for specific, supervisory tasks (for example, *Date* tags hold the current date in string format).

All system tags are *read-only*, which means you cannot add, edit, or remove these tags from the database.

- **Security** contains all of the group and individual user security accounts configured for the current project.
- **Procedures** contains VBScript functions and sub-routines that can be called by any other script in the project.
- **Event Logger** contains logging and event-retrieval features.
- **Translation** contains the translation worksheet that defines how your project's user interface should be translated into another language.

Graphics tab

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



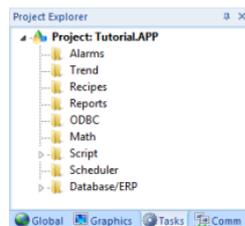
Graphics tab of the Project Explorer

The folders on the *Graphics* tab are described on the following pages:

- **Screens** contains all of the [screens](#) created for the current project.
- **Screen Group** contains the entire [screen groups](#) (individual screens combined into manageable groups) created for the current project.
- **Web Pages** contains all of the [Web pages](#) (i.e., screens saved in HTML format) created for the project.
 - **Mobile Access** allows [configuration of the mini-site](#) that is targeted to cell phones, PDAs, and other mobile devices.
- **Project Symbols** contains all of the [user-defined symbols](#), which can be groups of images and/or text. You can create custom symbols for the project and save them into this folder.
- **Graphics Script** contains [predefined functions that are executed when certain screen actions occur](#), such as when the Thin Client is launched on a remote station.
- **Symbols** contains the [library of common symbols and graphics](#) provided with the project. Double-click the **Library** icon to open the *Symbol Library*.
- **Layout** displays all screens currently open in the Screen Editor and allows you to [visualize how the screens fit together](#) during runtime.

Tasks tab

The **Tasks** tab of the Project Explorer organizes the worksheets that are processed as background tasks during project runtime.



Tasks tab of the Project Explorer

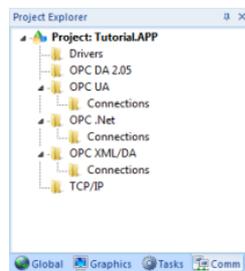
The folders on the Tasks tab are described on the following pages:

- **Alarms** contains the [Alarm worksheets](#) used to configure alarm groups and the tags related to each alarm group in the project. You also use the Alarm task to define the alarm messages generated during project runtime.
- **Trend** contains the [Trend worksheets](#) used to configure history groups that store trend curves for the project. You can use the Trend task to declare which tags must have their values stored on disk, and to create history files for trend graphs. Your project stores the samples in a binary history file (*.hst), and displays both history and on-line samples in a trend graph screen.
- **Recipes** contains the [Recipe worksheets](#) used to configure how data is exchanged between the project database and disk files in ASCII or DBF format, and how values are transferred between files and real-time memory.

- **Reports** contains the [Report worksheets](#) used to configure reports (text type) that are sent to a printer or a disk. Reports tasks allow you to configure text reports with system data, which makes report creation easier and more efficient.
- **ODBC** contains the [ODBC worksheets](#) used to configure how the ODBC interface runs in a network environment and uses standard Windows ODBC configuration. You configure ODBC tasks to exchange data between your project and any database supporting the ODBC interface.
- **Math** contains the [Math worksheets](#) used to configure and implement additional routines to work with different tasks. Your project executes *Math* worksheets as Background Tasks during runtime. You can configure *Math* worksheets to provide free environments for logical routines and mathematical calculations required by the project.
- **Script** contains the [Startup Script](#) and other [Script Groups](#).
- **Scheduler** contains the [Scheduler worksheets](#) used to configure events using defined mathematical expressions, which are executed according to time, date, or other monitored event.
- **Database/ERP** contains the [Database worksheets](#) that communicate 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 establish communication with another device or software using available protocols.



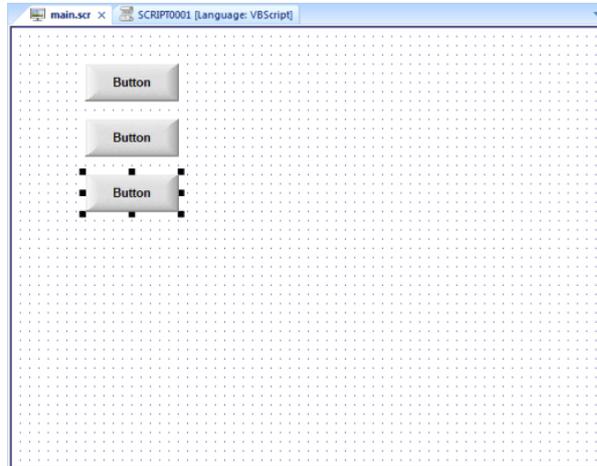
Comm tab of the Project Explorer

The folders on the Comm tab are described on the following pages.

- **Drivers** contains the [Driver worksheets](#) used to configure a communication interface (or interfaces) between the project and remote equipment (such as a PLC or transmitters).
A communication driver is a .DLL file that contains specific information about the remote equipment and implements the communication protocol.
- **OPC DA 2.05** contains the [OPC worksheets](#) used to configure communications between your project (which acts as an OPC client) and any other device that implements the classic OPC standard.
- **OPC UA** contains the [OPC UA worksheets](#) that are used to connect to OPC Servers via the new OPC Unified Architecture protocol.
- **OPC .Net** contains the [OPC .Net worksheets](#) that are used to connect to OPC Servers via the new OPC .NET 3.0 protocol.
- **OPC XML/DA** contains the [OPC XML/DA worksheets](#) that are used to connect to OPC Servers via the new OPC XML-DA protocol.
- **TCP/IP** contains the [TCP/IP worksheets](#) used to configure TCP/IP Client interfaces for other IWS stations.
IWS TCP/IP Client and Server modules enable two or more projects to keep their databases synchronized using the TCP/IP protocol.
- **DDE** contains the [DDE worksheets](#) used to configure a DDE Client for a DDE Server application (such as Microsoft Excel or any other Windows program that supports this interface).
DDE (Dynamic Data Exchange) is a protocol that enables dynamic data exchange between Windows applications. A DDE conversation is an interaction between server and client programs. IWS provides interfaces that run as clients or as servers.

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.



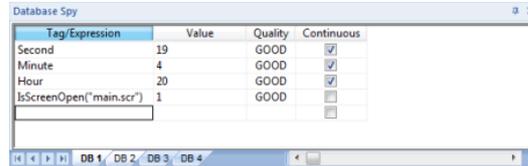
Screen/Worksheet Editor

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.



Tag/Expression	Value	Quality	Continuous
Second	19	GOOD	<input checked="" type="checkbox"/>
Minute	4	GOOD	<input checked="" type="checkbox"/>
Hour	20	GOOD	<input checked="" type="checkbox"/>
IsScreenOpen("main.scr")	1	GOOD	<input type="checkbox"/>

Sample Database Spy window

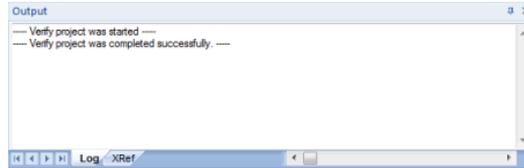
The window contains the following elements:

- 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 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.

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

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 (☒): 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 (⌵): 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.

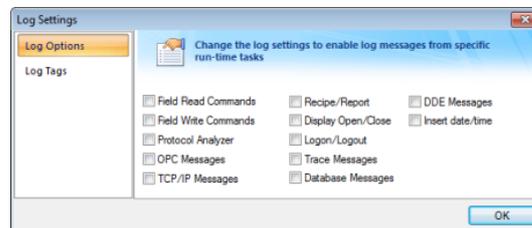
The *Output* window is dockable, which means you can move it to another location in the development environment. Click on the title bar and drag the window to a new location. Release the mouse button to attach or *dock* the window to its new location.

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.

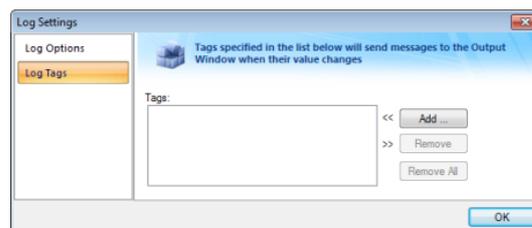


Log Settings — Options Tab

Option	Description
Field Read Commands and Field Write Commands	Show any read and/or write commands that are sent to connected devices.

Option	Description
Protocol Analyzer	Show messages generated by configured device drivers .
OPC Messages	Show messages generated by OPC communications.
TCP/IP Messages	Show messages generated by TCP/IP communications.
Recipe/Report	Show messages generated by the Recipe and Report tasks.
Display Open/Close	<p>Display detailed information whenever a screen is opened or closed:</p> <ul style="list-style-type: none"> Disk Load Time: Time to load the screen file from the disk into memory. Open Time: Time to open the screen, including initializing tags used in the screen and running any "OnOpen" scripts or functions. Total Load Time: Total time to load the screen (includes Disk Load Time and Open Time above). First Draw Time: Time to first drawing of screen objects. First OnWhile Time: Time to first running of any "OnWhile" scripts or functions. Total Open Time: Total time to open the screen (includes First Draw Time and First OnWhile Time above). Close Time: Time to close the screen, including finalizing tags used in the screen and running any "OnClose" scripts or functions. Total Close Time: Total time to close the screen, including the time to close the screen file on the disk. <p>This information can be used to analyze runtime performance on low-end target systems. If a particular step of opening or closing takes an unusually long time, then it can be identified and redesigned.</p>
Logon/Logout	Display a message whenever a user logs on or logs out. (For more information, see Security .)
Trace Messages	Show messages generated by the Trace () function. This function is used to generate customized messages from within your project.
Database Messages	Show messages generated by the ODBC and ADO.NET database interfaces.
DDE Messages	Show messages generated by DDE communications.
Insert date/time	Timestamp each message.

- In the *Log Tags* tab of the dialog, click **Add** to browse for [project tags](#).



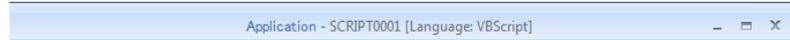
Log Settings — Tags Tab

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

- 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:

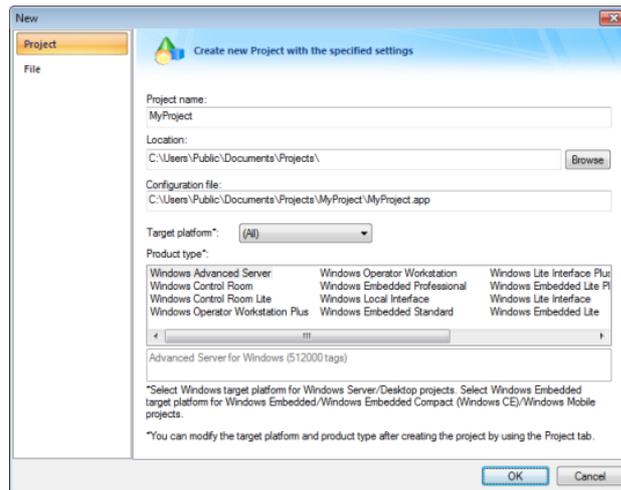
Field	Description
Execution Mode	The current execution mode of the application.
CAP	Indicates whether the keyboard Caps Lock is on (black) or off (grey).
NUM	Indicates whether the keyboard Num Lock is on (black) or off (grey).
SCRL	Indicates whether the keyboard Scroll Lock is on (black) or off (grey).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

Creating a New Project

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 the Application menu.
The *New* dialog is displayed.
2. Click the **Project** tab.



Project tab of the New dialog

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\My Documents\InduSoft Web Studio v7.0 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. In the **Target platform** list, select the platform of the computer or embedded device that will host your project runtime.
Selecting a platform will restrict the list of available product types to only those types that work on the selected platform. It is not absolutely necessary to creating a new project — if you leave the selection as **(All)**, then all product types are shown — but it helps you to make the decision.
6. 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](#).

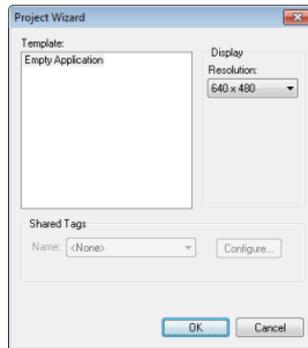
 **Tip:** You can change the product type later, after you have created the project, by using the **Target System** command on the **Project** tab of the ribbon.

 **Note:** 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 v7.0 > 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 you are not licensed to run.

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.

- Click **OK**.
The *New* dialog is closed, and the *Project Wizard* dialog is displayed.



Project Wizard dialog

- 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 can change the resolution of individual project screens later, after you have created the project, by editing the screen attributes. Also, you can 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.

- 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 runtime.

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 or later
- Microsoft Windows Vista Service Pack 1 or later
- Microsoft Windows 7, all versions
- Microsoft Windows Server 2003 Service Pack 2 or later
- Microsoft Windows Server 2008, all versions
- Microsoft Windows XP Embedded Service Pack 3
- Microsoft Windows Embedded Standard 7 (2009)

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 [Installation](#).

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 (2009)
- Microsoft Windows Embedded Compact (formerly known as Windows CE or Windows Mobile), version 5.0 or later

The device must have either EmbeddedView (for Windows XP Embedded and Windows Embedded Standard) or CEView (for Windows Embedded Compact) installed. The device must also have an appropriate runtime license key. For more information, see [Installation](#).

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 UA Client task and worksheets.
- 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.

 **Note:** 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.

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 may create in the project, including tags shared or imported from other systems. Windows can support far more tags than Windows Embedded.

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.

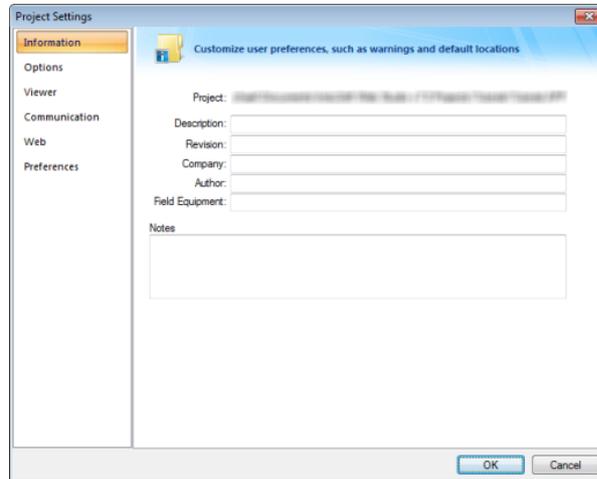
1. On the **Project** tab of the ribbon, in the **Settings** group, click **Target System**.
A list of available product types is displayed.
2. In the list, select the new product type of the project.
The product type determines how many tags your project will support, among other things. For more information, see [About target platforms and product types](#).

The project is converted to the selected product type.

 **Note:** If you changed from a Windows target platform to a Windows Embedded target platform, some features will no longer be available.

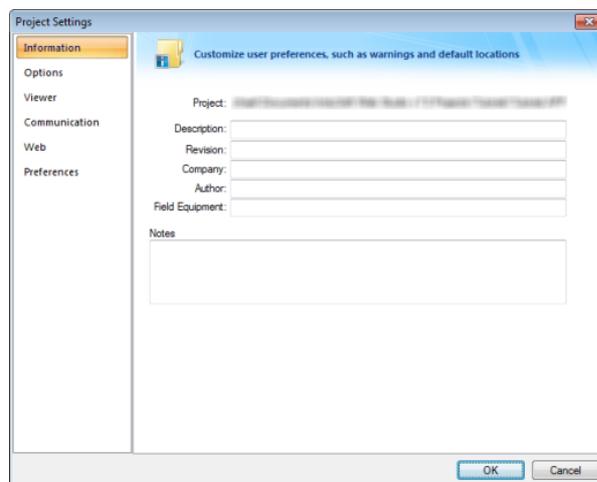
Configuring additional project settings

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



Project Settings dialog

Information tab



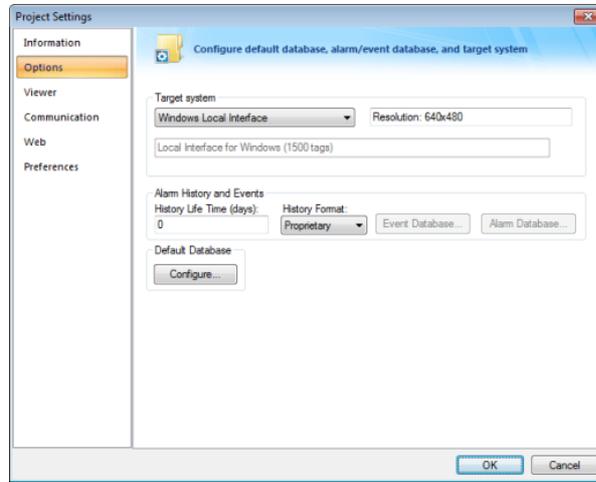
Project Settings: Identification tab

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

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.



Caution: If you specify a **Target System** level that does not match the actual license level on the target system, then your project may 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:

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

For more information, see [Saving your alarm history / event log to an external database](#).

- **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.
- **Shared Tags** pane: Select a third-party software from the combo-box. Click the Configure button to configure the settings for importing tags from one of the following data sources into the [Shared Database](#) folder:
 - **<None>:** Does not share tags with any external software

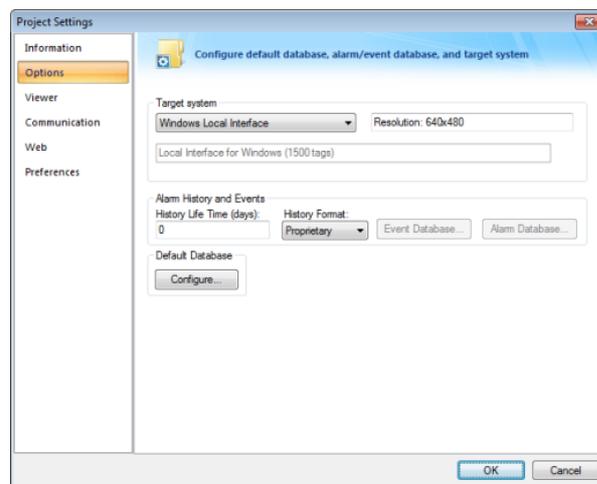
- **First ISaGRAF:** Import tags from a First ISaGRAF project into the Shared Tags folder of the current project, and, if enabled, configure the communication interface with the ISAGR driver automatically.
- **ISaGRAF:** Import tags from a ISaGRAF project into the Shared Tags folder of the current project, and, if enabled, configure the communication interface with the ISAGR driver automatically.
- **Think & Do:** Import tags from a Think & Do project into the Shared Tags folder of the current project, and, if enabled, configure the communication interface with the TND driver automatically.
- **SteepleChase:** Import tags from a SteepleChase project into the Shared Tags folder of the current project, and configure the communication interface with the VLC driver automatically.
- **SixNet:** Import tags from a SixNet project into the Shared Tags folder of the current project, and configure the communication interface with the SNET driver automatically.
- **OpenControl:** Import tags from an OpenControl project into the Shared Tags folder of the current project, and configure the communication interface with the OC driver automatically.
- **Straton:** Import tags from a Straton project into the Shared Tags folder of the current project, and configure the communication interface with the STRAT driver automatically.

 **Note:** PC-based control has its own, customized interface that requires you to provide information about the PC-based control application in order to share tags with the IWS project.

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.

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Options**. The *Project Settings* dialog is displayed.



Project Settings: 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](#).

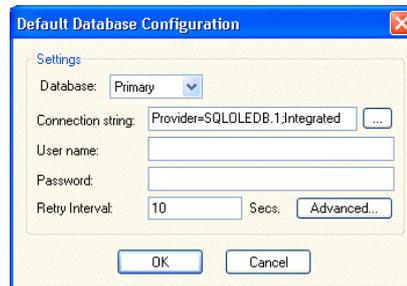
- To configure a separate database for either your event log or your alarm history, click **Event Database** or **Alarm Database**, respectively. In either case, a *Database Configuration* dialog is displayed. Use the dialog to configure the database connection. For more information, see [Database Configuration](#).
- 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:

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

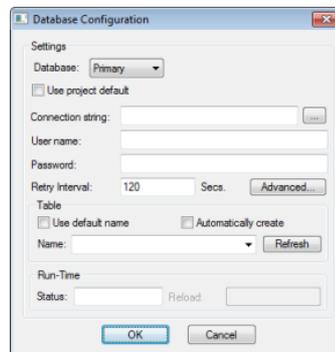


Default Database Configuration dialog

Please refer to [Database Configuration dialog](#) for help completing the fields in this window.

DATABASE CONFIGURATION

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



Database Configuration dialog

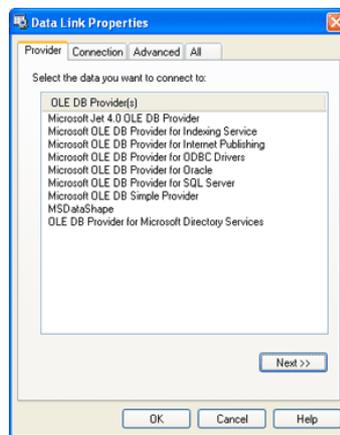
- Database** combo-box: Allows you to select either Primary or Secondary. With Primary, all settings displayed in the Database Configuration window apply to the Primary Database interface. Otherwise, they apply to the Secondary Database interface. You can configure the Secondary database in the following 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.

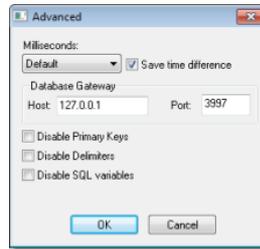


Data Link Properties dialog

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

- **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.
- **Save time difference** checkbox: When this option is checked (default), IWS saves the Time Zone configured in the computer where the project is running in each register of the database. This option must be enabled to avoid problems with daylight savings time.
- **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.

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:

Value	Description
0	Disconnected from the database. The database is not available; your configuration is incorrect or it is an illegal operation.
1	The database is connected successfully.
2	The database is being synchronized.

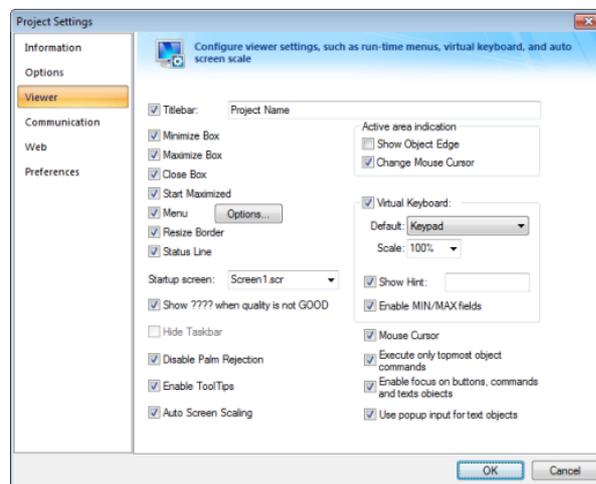
- **Reload** (output): Specify a reload tag if you are using curly brackets in any of the configuration fields. When you want to reconnect to the database using the updated values on your tags, set the tag on this field to 1. IWS will update the configuration when trying to perform an action in the database,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

See also:

[Configuring a Default Database for All Task History.](#)

Viewer tab

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



Project Settings: Runtime Desktop 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.



Runtime Menu Options dialog

- **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 a 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.
- **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.*
- **Active area indication** pane: 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 panels — 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.

- **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:

Value 1: 0 Value 2: 0

Focus on text input (left), no focus on text input (right)

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:



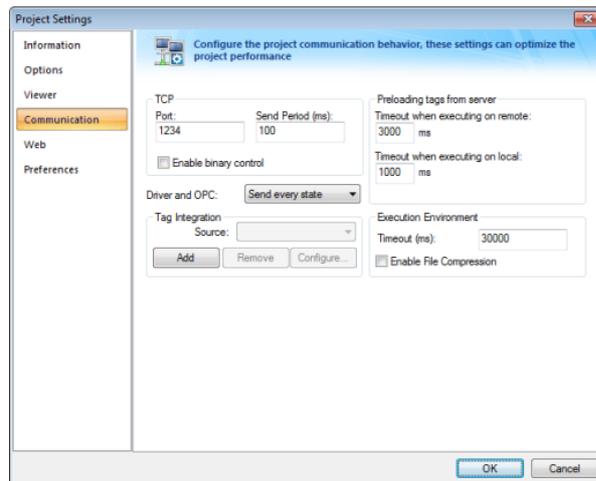
Popup for text input

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.

Communication tab

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



Project Settings: Communication tab

- **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.
 - **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.
- **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 modules running in the current computer. 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.

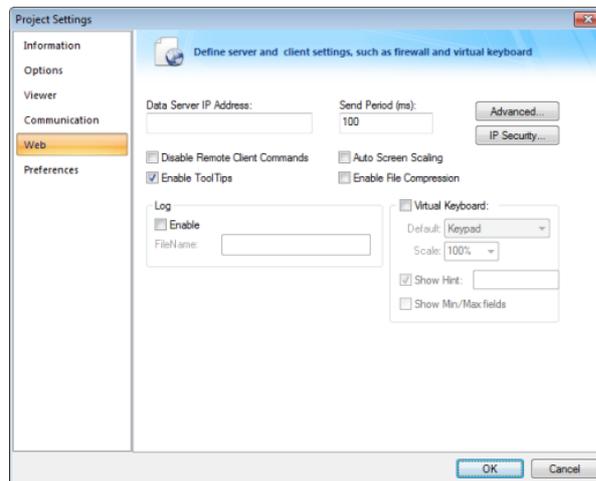
Note: This setting is required for all projects and cannot be left blank. Make sure that your network configuration will allow connections to this port on whichever station will be the project server.

- **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.

- **Remote Management** area: Configure the communication settings for the [Remote Management operation](#), 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.
- **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.

Web tab

Select the Web tab on the *Project Settings* dialog:



Project Settings: Web tab

Note: If you change any of these settings, then you should [verify your project](#) before resending it to the target device.

Configure the parameters on this tab as follows:

- **Data Server IP** field: Type 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 of IWS is running. If this field is left blank, then the Thin Client will assume that the Web server (i.e., the address entered into the browser) is also the data server.

Tip: You can use the IP address **127.0.0.1** (localhost) to access the TCP/IP server on the local computer, regardless of its actual address on the network. This option is useful for local tests; however, you are not able to access the data server from remote computers using this configuration.

- **Send Period** field: Type a value to specify the send period (in milliseconds) used to exchange data between the server and the Thin Client stations. It means that the Thin Client will send a package with the new tag values to the server every *n* millisecond(s).

The **Send Period** of the server is configured in the project settings (**Communication** on the Project tab of the ribbon). The default value is **1000** (milliseconds). You can set a lower value in this field to increase

the update rate between the server and the Thin Clients. Doing so may result in higher traffic in the network (the network will be accessed more frequently) if the tags are changing continuously (faster than 1 second).

- **Disable Remote Client Commands** checkbox: Click to enable this box, to prevent a remote client from issuing commands from your Thin Client to your server. When this option is enabled, the Thin Client can read data from the server, but cannot send data (tag values, set-points) to the data server. In this case, the Thin Client station becomes a *Read Only* station.
- **Enable ToolTips** checkbox: Click to enable this box, to display the *ToolTips* configured on the objects of the screens when viewing them on the Thin Client (web browser).
- **Auto Screen Scaling** checkbox: Click to enable this box, to automatically scale screens displayed in a web browser window. Using this option, the screen fits to the size of the web browser window, regardless of its resolution.

 **Note:** The **Auto Screen Scaling** option is not valid for Web browsers running under Windows CE operating systems.

- **Enable File Compression** checkbox: Click to enable this box to compress the files stored on the Web sub-folder of your project folder. This option is useful for reducing download time, particularly if you have a slow connection between your server and the Thin Client.
- **Log (Enable checkbox and FileName text field):** Click to enable the checkbox, and type a file name into the text field to generate a log file on the Thin Client station. You can use this log file for debugging purposes.
- **Virtual Keyboard:** When this option is checked, the **Virtual Keyboard** is enabled for your project running Thin Clients. The keyboard allows the user to enter data during runtime on touchscreen panels — 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.

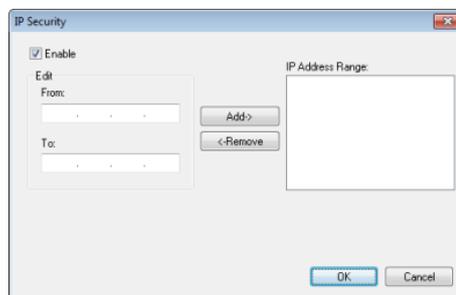
You can establish a default configuration for the virtual keyboard:

- **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 objects, the hint is configured in the *Object Properties dialog*. Otherwise, enter a string or string Tag in the **Show Hint** field to serve as a default hint.
- **Show 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.

IP Security Settings

IP Security button: Click this button to open the IP Security dialog.



IP Security dialog

Use the parameters on this dialog to specify the range of IP addresses for the computers that are allowed to access your project as Thin Clients. This option is useful when you can control the IP Address of the Thin Client computers allowed to connect to the Web server.

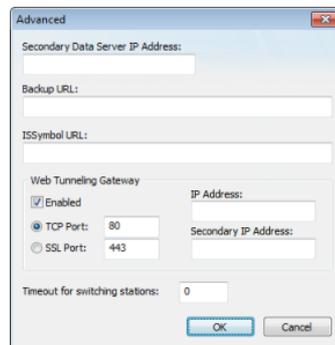
Click the **Enable** checkbox, and when the **Edit** pane parameters become active, type IP addresses in the **From** and **To** fields to specify the IP address range. Use the **Add** and **Remove** buttons to move the IP addresses into the **IP Address Range** list. When a Thin Client attempts to connect to the server, it checks for an IP address for the Thin Client station that is within any range configured in the IP Security dialog. If one is not found, the server refuses the connection request from the Thin Client station.

Note: By default, IP security applies only to Thin Clients connecting to the Data Server. You can also implement IP security for [database synchronization between projects running on different stations](#). To do this, insert the following parameter into your project file (*project_name.app*):

```
[TCP]
UseWebIPSecurity=1
```

Advanced Settings

Advanced button: Click this button to open a dialog where you can edit the Advanced Web settings. For most cases, these settings do not need to be modified. However, depending on the architecture used in your project, you have the flexibility to configure advanced settings.



Advanced dialog

- **Secondary Data Server IP Address** field: Type the IP address (or host name) of the secondary data server station. The data server station is the computer or device where the TCP/IP server module of IWS is running. This field must be filled when you are using redundant data servers from the Thin Clients. If the primary data server fails, the Thin Client will attempt to connect to the secondary data server automatically.
- **BackUp URL** field: Type the URL where the Web files are stored (files from the web sub-folder of your project folder). This URL is used to download the files from the secondary Web server when the primary Web server is not available.
- **ISSymbol URL** field: When the Thin Client connects to the server, it attempts to load the [ISSymbol control](#). If ISSymbol is not registered in the local computer (Thin Client), the browser will attempt to download it from the URL specified in this field. The default URL is a web site where InduSoft keeps the most updated version of ISSymbol available for download. You may need to configure a different location, especially when the Thin Client computer is not connected to the internet. The **ISSymbolVM.cab** (stored in the \BIN sub-folder of IWS) must be available in the URL configured in this field.

Tip: When the Thin Client stations do not have access to the internet, it is recommended that the **ISSymbolVM.cab** file be made available at the Web server station, and that the URL be configured for it in this field.

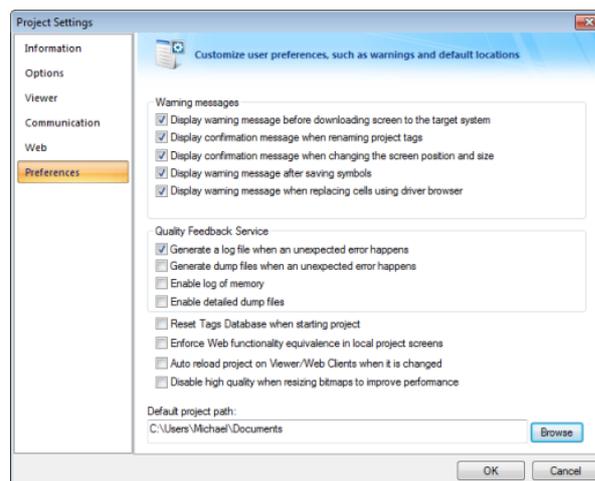
Note: Web browsers running under the Windows CE operating system are not able to download ISSymbol control (**ISSymbolCE.ocx**) automatically from a remote location. **ISSymbolCE.ocx** must be manually registered in the Windows CE device to be used as a Thin Client.

- **Web Tunneling Gateway** checkbox: Check this option to enable the Web Tunneling Gateway. Depending on the [architecture](#) of your project, you may need to use the Web Tunneling Gateway to route the Thin Client computers to the data server.
- **TCP Port**: Select this option when using HTTP with the Web server of Microsoft IIS. You can specify the TCP port used by your HTTP Web server (80 is the default TCP port for HTTP protocol).
- **SSL Port**: Select this option when using SSL (Secure Socket Layer) with the Microsoft IIS Web server. You can specify the TCP port used by your HTTPS Web server (443 is the default TCP port for HTTPS protocol).
- **IP Address**: IP address of the Web server computer where the Web Tunneling Gateway is running. This must be the IP address of the Web server accessible from the Thin Client station(s).
- **Secondary IP Address**: IP address of the Web server computer where the secondary Web Tunneling Gateway is running. This must be the IP address of the secondary Web server accessible from the Thin Client station(s). This field must be configured when you are using redundant Web servers.

 **Tip:** Most of the Web settings can be modified dynamically by the [SetWebConfig](#) function. It is especially useful when you want to create a project just once, and make sure that the Web settings are automatically configured when you run your project on different stations with different IP Addresses.

Preferences tab

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



Project Settings: Preferences tab

Warning Messages

- **Display warning message before downloading screen to the target system** option: When this option is selected and IWS is connected to a remote station ([Remote Management](#) dialog), you are prompted to download the updated screen to the remote station immediately after saving a screen on the screen editor. If this option is not checked, 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 checked, 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](#).

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 \Web\Dump sub-folder of the running project.

- **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 (\Web\Dump*.dmp) with useful information about the conditions of the error. This is a binary file that can only be read by the software vendor.

 **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, then 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,

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

Other Preferences

- **Reset Tags Database when starting project** option: When this option is selected, the project tags are reset automatically whenever you run the project (**Run** on the Home tab of the ribbon). See [Reset Tags Database](#) for additional details about this feature.
- **Enforce Web functionality equivalence in local project screens** option: When this option is selected, the development software will automatically warn you when you try to select functions or features that are incompatible with the remote runtime modules (e.g., Thin Client and Secure Viewer).

 **Note:** This option is unchecked by default in order to maintain compatibility with previous versions of InduSoft Web Studio.

- **Auto reload project on Viewer/Web Clients when it is changed** option: When this option is selected, remote stations (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, then they will automatically download the new version from the server.

Mobile Access

Studio Mobile Access (SMA) enables your project to send Alarms and Process information to cell phones, PDAs, and other mobile devices.

SMA is somewhat different from the traditional [Thin Client](#) solution, however. When you create your project for an operator interface, you do not want to worry about details like creating additional screens that would fit on a cell phone. SMA takes care of these details and provides an easy-to-use interface for getting [alarm notifications](#) and [tag values](#) on almost any mobile device.

How It Works

When you enable the Mobile Access feature in your IWS project and then run your project, IWS creates a Collaboration Data Object (CDO) on the server and periodically refreshes it with alarm notifications and whatever tag values you choose to make available. (CDO is a Microsoft .NET technology that is used to share data between programs. It was previously known as Active Messaging.)

Once the SMA data object is in place, an ASP-powered Web application parses the data and builds lightweight pages for mobile browsers. 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 Web application and log on.

The connection between your project and the SMA data object goes both ways, so you can also acknowledge alarms and write new tag values through the Web application. These actions are recorded by the SMA data object and then passed back to your project.

If you're an experienced ASP developer, then you can modify the default Web application or build your own to access the SMA data object in new and creative ways. Doing so, however, is beyond the scope of this documentation. Please contact Customer Support for help.

Licensing

One SMA Client is included with every IWS runtime license. That means the SMA Web application will allow only one user to connect at a time. If you want the Web application 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

To enable Mobile Access and configure the data to be served:

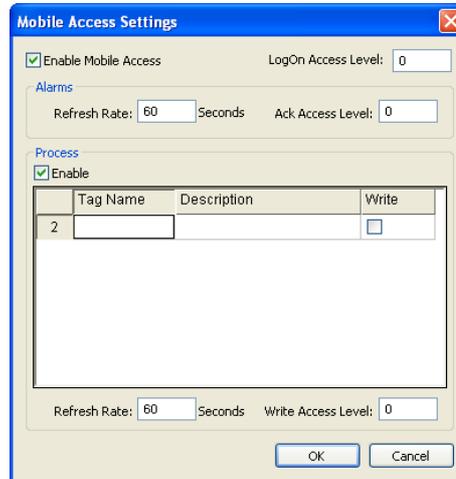
1. In *Graphics* tab of the [Project Explorer](#), open the **Web Pages** folder.



Opening the Web Pages folder

2. Double-click the **Mobile Access** icon.

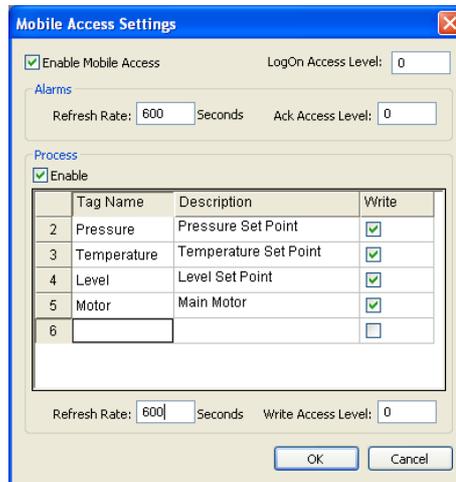
The *Mobile Access Settings* dialog is displayed.



Mobile Access Settings dialog

3. Select **Enable Mobile Access**.
4. In the **LogOn Access Level** text-box, type the user security level needed to log on to the Web application. For more information about security levels, see [Security](#).
5. The Web application 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.
6. To have the Web application show tag values, select **Enable** in the **Process** group-box.
7. 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 Web application 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 Web application.
8. 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.
9. 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.
10. Click **OK** where you are done.

The following screenshot show Mobile Access enabled with a selection of tags:



Example of Mobile Access settings

Installing and Configuring IIS

Studio Mobile Access (SMA) uses Collaboration Data Objects (CDO) and Active Server Pages (ASP) to build the Web application pages for mobile browsers. The mobile browser does not need to support Java®, Flash™, or any other advanced features because the pages are built entirely on the server-side and then sent to the browser as simple HTML. The Web server, however, must support CDO and ASP, and that typically means it must be Microsoft IIS running on Windows. For more information about installing and configuring IIS, see "[Configuring a Web server to host your project pages](#)" as well as the Microsoft IIS documentation.

Accessing the Web Application

Once you've enabled Mobile Access, 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
`http://server_address/SMA/LogOn.asp`
- If the IIS home directory is set to the \Web\SMA sub-folder in your project folder, then the URL is
`http://server_address/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 SMA 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 Web application.

This menu is also displayed in the Alarms and Process pages described below.

Alarms

Alarms

[Home](#) [Process](#) [Log Off](#)

State	Message	Type	Time
	Hi Pressure	HiHi	14:59
	Low Temperature	Lo	14:44
	Hi Level	HiHi	14:59

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

Process

[Home](#) [Alarms](#) [Log Off](#)

Description	Value
Pressure Set Point	27
Temperature Set Point	90
Level Set Point	50
Main Motor	On

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
```

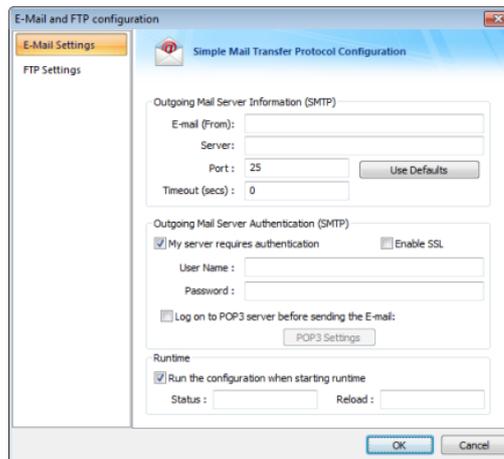
Please note that as long as a user is logged on, he counts against the number of SMA Clients in the runtime license. If too many users stay logged on for extended periods, then you may run out of available connections.

Configuring 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.

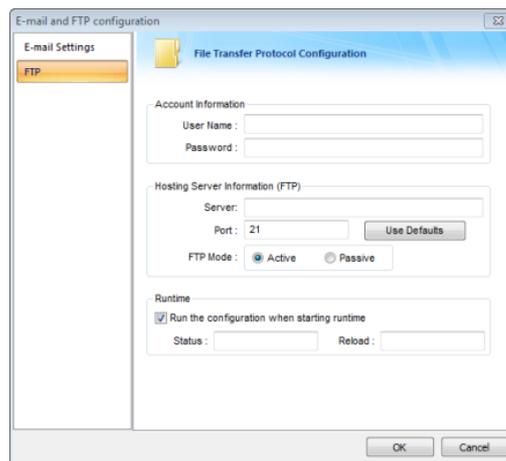
6. In the **Status** box, type the name of a tag (Integer type) that will receive status codes when the project sends email.
7. In the **Reload** box, type a tag/expression. When the value of this tag/expression changes, the project will reload these default email settings.
8. Click **OK** to save your 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.

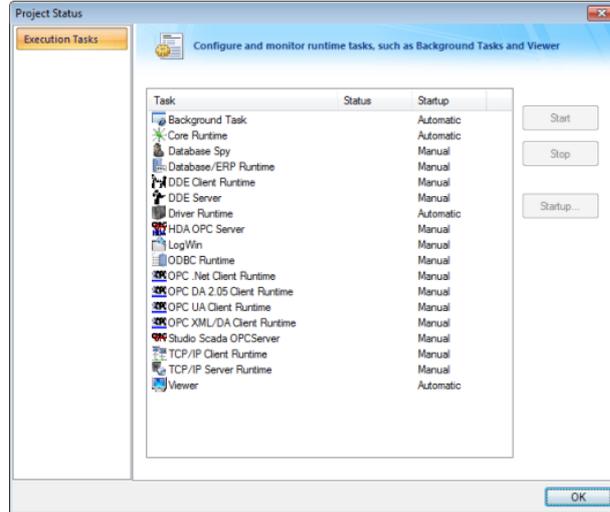


3. In the **User Name** and **Password** boxes, type your credentials for the FTP server.
4. In the **Server** and **Port** boxes, type the server address and port number.
The default port for FTP is 21, but it depends on your server and network configuration. Please consult your server administrator.
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.

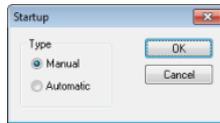


Execution Tasks dialog – Execution Tasks

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

Task	Available for Windows Embedded
Background Task	Yes
Database Client Runtime	Yes
Database Spy	No
DDE Client Runtime	No
DDE Server	No
HDA OPC Server	No
Driver Runtime	Yes
LogWin	No
ODBC Runtime	No
OPC Client Runtime	Yes
Studio Scada OPC Server	Yes
TCP/IP Client Runtime	Yes
TCP/IP Server	Yes
Viewer	Yes

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.

Running a Project Under Windows Services

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 boots, 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.

Creating and Configuring the Windows Service

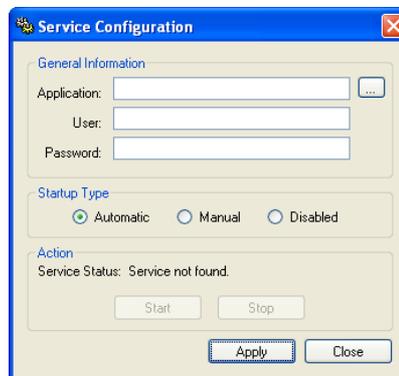
Note: All of the procedures described below were tested using Microsoft Windows XP Service Pack 3.

Also, to perform these actions, you must be logged on as a user with administrative privileges and you should know how to use the *Computer Management* console. (To access the console, right-click the **My Computer** icon and choose **Manage** from 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 IWS itself, or you can use the command-line utility that is installed with IWS.

Service Configuration Tool

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



Service Configuration dialog

Project box

The location of the project file (*project_name.app*) that the service will load and run when it is started. This must be a complete file path.

User box

The Windows user account under which the service will run. This is an optional setting; if it's 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 box

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

Startup Type pane

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

- **Automatic:** The service will start automatically when the computer boots.
- **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 an administrative user 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 > Administrative Tools > Services**) under the name "InduSoft Web Studio". You can use that console to quickly stop and restart the service, if you don't want to run the IWS development environment.

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 launching the IWS development environment. 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, and enter the command with the desired options.

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 (`project_name.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 switch 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 switch is optional; if it's not used, then the default behavior for a new service is **manual**.

-user username

Specifies the Windows user account under which the service will run. (This is the same as the **User** box in the *Service Configuration* dialog.) This is an optional switch; if it's 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 is an optional parameter; it's not needed if no user is specified or if the specified user doesn't 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 is an optional switch; 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 is an optional switch.

-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\My Documents\InduSoft Web Studio v7.0 Projects\project_name\project_name.app
Startup Mode	Automatic
User	IWS
Password	IWS
Service Name	"InduSoft Web Studio"
Service Description	"Starts a IWS project"

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

So, to create the service with the desired options:

1. Make sure you're logged in as a user with administrative privileges.
2. Open a command prompt (**Start > All Programs > Accessories > Command Prompt**).
3. Navigate to the Bin folder:

```
cd Bin
```

4. Enter the command:

```
StdSvcInst -create -app "C:\Users\username\My Documents\InduSoft Web Studio v7.0  
Projects\project_name\project_name.app" -startup auto -user IWS -password IWS -name  
"InduSoft Web Studio" -descr "Starts a 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** switch:

1. Make sure you're logged in as a user with administrative privileges.
2. Stop the service if it is running.

3. Open a command prompt.
4. Navigate to the Bin directory.
5. Enter the command — for example, to set NTDemo as the project file:

```
stdSvcInst -app "C:\Program Files (x86)\InduSoft Web Studio v7.0\Demos\NTDemo\NTDemo.app"
```

Example: Deleting the Service

To delete the service:

1. Make sure you're logged in as a user with administrative privileges.
2. Stop the service if it is running.
3. Open a command prompt.
4. Navigate to the Bin directory.
5. Enter the command:

```
stdSvcInst -delete
```

Configuring User Privileges

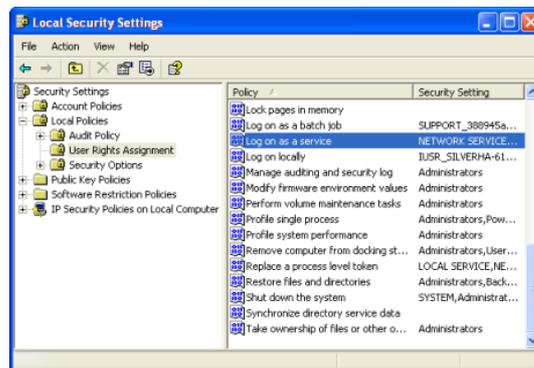
The service will run under the privileges of the user account specified in the **User** field of the *Service Configuration* tool (or by the **-user** switch of the command-line utility). If IWS needs some system resource to which that account doesn't have privileges, it will fail. Therefore, you must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

Note: The following actions can be performed only by a user with full 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:

1. Open the *Local Security Settings* console (**Start > Control Panel > Administrative Tools > Local Security Policy**).
2. In the console window, select the folder "Security Settings\Local Policies\User Rights Assignment".
3. In the list of available policies, double-click **Log on as a service**.



Selecting the "Log on as a service" policy

The *Log on as a service* properties page 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 Program Directory

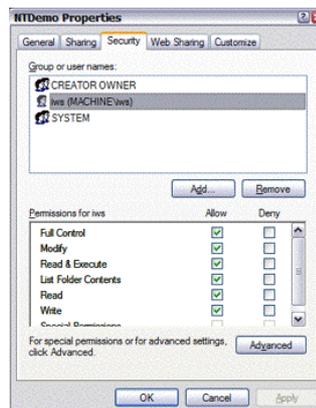
For your IWS project to run properly, the specified user account must have full control over the program directory and all of the files in it. To give the account those privileges:

1. Open *Windows Explorer* (**Start > All Programs > Accessories > Windows Explorer**).
2. Choose **Tools > Folder Options**.
The *Folder Options* dialog is displayed.
3. Click the **View** tab.
4. Make sure the option **Use simple file sharing** is *not* selected.



Make sure simple file sharing is disabled

5. Click **OK** to close the *Folder Options* dialog.
6. Still in *Windows Explorer*, locate and select your IWS program directory, i.e., the folder that contains the file `project_name.APP`.
In this example, the folder is `NTDemo`.
7. Right-click the folder and choose **Properties** from the shortcut menu.
The directory's property sheet is displayed.
8. Click the **Security** tab.



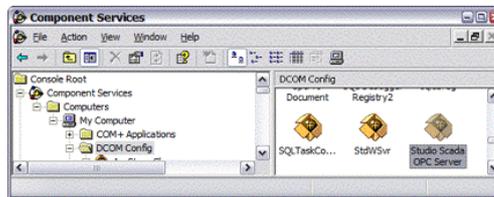
Properties dialog for the program directory

9. Click **Add** and then add the user account you specified when you created the service.
In this example, the user is `iws`.
10. Select this user and then set its "Full Control" permission to **Allow**.
11. Click **OK** to apply your changes and close the dialog.
12. Close *Windows Explorer*.

Allowing the User Account to Run the OPC Client/Server Module

As mentioned previously, normal users have too little privileges to properly run the OPC Client/Server module. Therefore, you must configure your user account to have those privileges:

1. Open the *Component Services* console (**Start > Control Panel > Administrative Tools > Component Services**).
2. In the console window, select the folder "Console Root\Component Services\Computers\My Computer\DCOM Config".



Selecting the OPC Server module in the Component Services console

3. In the DCOM Config pane, right-click the **Studio Scada OPC Server** icon and then choose **Properties** from the shortcut menu.

The *Studio SCADA OPC Server* property sheet is displayed.

4. Click the **Identity** tab.
5. Click **This user** and then complete the fields with the same user and password that you specified when you created the service.

In this example, user is *Studio* and password is also *Studio*.

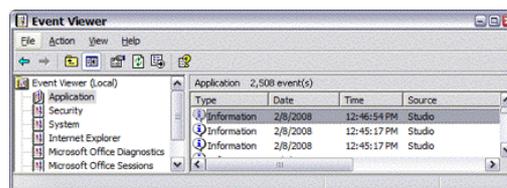


Entering the user name and password

6. Click **OK** to apply your changes and close the property sheet.
7. Close the *Component Services* console.

Troubleshooting

When IWS runs under Windows services, 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 > Administrative Tools > Event Viewer**).



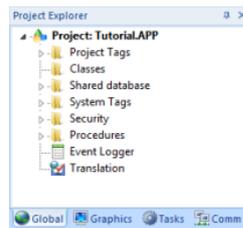
IWS error messages in the Event Viewer

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

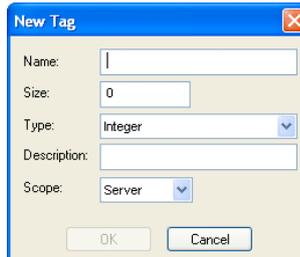
The *Project Tags* folder contains all tags created and customized by the user. You can create project tags for displays, to read from and write to field equipment, for control, to perform mathematical calculations, and so forth.

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

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:
 - **Local**: Select if you do not want to share the information in this tag over the Internet.
 - **Server**: Select if you want to share the information in this tag over the Internet.

These options have no effect 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 dialogues:

- *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 contains five columns (see the following sample dialog).

Application Tags					
	Name	Size	Type	Description	Scope
1	startup	0	Boolean		Server
2	SysOS	0	String		Server
3	SysProdVer	0	String		Server
4	SysCompName	0	String		Server
5	SysIPAddress	0	String		Server
6	testabc	0	Integer		Server
7	taga	0	Integer		Server
8	tagmin	0	Integer		Server
9	tagmax	0	Integer		Server
10	SimulAnalog	0	Real		Server
11	SimulDigital	0	Real		Server

Project Tags dialog

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:

```
[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.

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 `tk` tag is an *array*, you use the following syntax:

```
ArrayTagName[ArrayIndex].MemberName
```

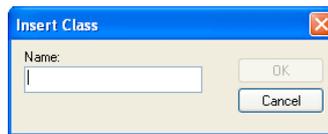
For example: `tk[1].LEV` or `tk[n].TMP`

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

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: Class1			
	Name	Type	Description
*		Integer	

Class Datasheet

Note: The maximum number of members for each class is 512 as long as it does not exceed the maximum number of tags supported by the license (Product Type) selected for the project. When a tag is created from a class type, each member from the class counts as one tag for licensing restrictions, because each member has an independent value.

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; or
- 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.

 **Note:** Use the @ character at the beginning of a tag name to indicate that the tag will be used as an [indirect tag](#) in the project.

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

Array Tag Example	Description
Tank[1], Tank[2], Tank[500]	Simple arrays, where the array indexes (1, 2, and 500) are numerical constants. For example, tank numbers.
Tank[tk]	A simple array, where the array index (tk) is a tag. For example, a tag representing the tank number.
Tank[tk+1]	A complex array, where the array index (tk+1) is an expression. For example, the value of tk (tank number) plus 1.

-  **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 make a tag into a compound tag by assigning it a **Class**. A Class is a template consisting of two or more tag definitions, each with its own data type. You can use Classes in projects that have items (e.g., tanks of liquid) with multiple attributes (e.g., fill level, temperature, pressure) to be monitored or controlled.

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](#)

Changing How Boolean Tags Receive Numeric Values

By default, if any numeric value other than 0 (i.e., $\neq 0$) is written to a Boolean tag, then the tag automatically assumes a value of 1. You can change this behavior, if necessary, by editing the `project_name.app` file to change the following setting:

```
[Options]  
BooleanTrueAboveZero=value
```

If `BooleanTrueAboveZero` is set to the default 0, then the project will behave as described above. If `BooleanTrueAboveZero` is set to 1, then the project will behave as follows:

- When you write any numeric value less than or equal to 0 (i.e., ≤ 0) to a Boolean tag, the tag assumes a value of 0 (false).
- When you write any numeric value greater than 0 (i.e., > 0) to a Boolean tag, the tag assumes a value of 1 (true).

 **Caution:** This is a global runtime setting. If you only want to change how certain tags are handled, then you should not change this setting.

Choosing the Tag Scope

IWS allows you to decide whether a tag "lives" on the project server or on each local station:

- **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:

Application Tags					
	Name	Size	Type	Description	Scope
1		0	Integer		Server
*			Integer		Server
*			Integer		Server
*			Integer		Server
*			Integer		Server
*			Integer		Server

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.

Application Tags					
	Name	Size	Type	Description	Scope
1	startup	0	Boolean		Server
2	SysOS	0	String		Server
3	SysProdVer	0	String		Server
4	SysCompName	0	String		Server
5	SysIPAddress	0	String		Server
6	testabc	0	Integer		Server
7	taga	0	Integer		Server
8	tagmin	0	Integer		Server
9	tagmax	0	Integer		Server
10	SimulAnalog	0	Real		Server
11		Server

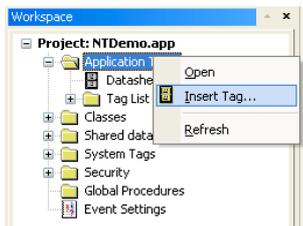
Example Project Tags Datasheet

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:

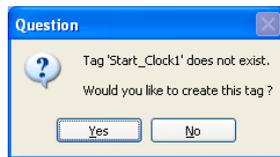
- On the Insert tab of the ribbon, in the Global group, click **Tag**;

- 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 Tags Datasheet

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 type the new information into the datasheet.
4. When you are finished editing, save your changes to the tags database.

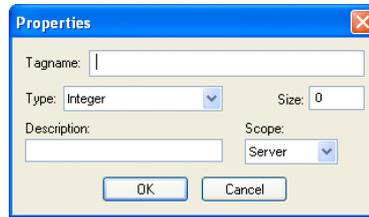
Tip: You can sort the data in the *Project Tags* sheet and/or insert/remove additional columns to/from the sheet by right-clicking on it and choosing the applicable option from the shortcut menu.

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 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.



Properties dialog

The *Properties* dialog contains fields and combo-boxes that correspond in name and function to the columns on the *Project Tags* datasheet.

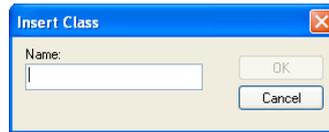
3. Make your changes in the *Properties* dialog as follows:
 - To change the current **Type** or **Scope** properties, click the arrow button and select the new information from the list.
 - To change the **Size** or **Description**, highlight the existing text and type the new information into the text 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.

Creating Classes

To create a new class tag:

1. From the **Global** tab, right-click the Classes folder and then select **Insert Class** from the resulting shortcut menu.
2. When the *Insert Class* dialog displays, type a name into the **Name** text box using the [design guidelines](#) and [tag name syntax](#).



Insert Class dialog

3. Click **OK** to close the *Insert Class* dialog. The *Class: worksheet* is displayed automatically.

Configure the columns in this worksheet as follows:

- **Name** field: Type a class member name.
- **Type** drop-down list: Click the arrow to select the class member's **data type** (Boolean, Integer, Real, or String) from the list.
- **Description** field (optional): Type a description of the class member (for documentation purposes only).

	Name	Type	Description
1	temperature	Real	Tank temperature
2	pressure	Real	Tank pressure
3	level	Real	Tank level
4			
5			

Sample CTank worksheet

4. Click in the next blank line and provide the information for the next class member you want to include in this class. Or, if you are finished adding members, you can close the *Class* worksheet.

You can expand the Classes folder and subfolders to see the data structure:



Expanded Classes folder

5. Next, use the instructions provided in "[Adding Tags to the Datasheet](#)" to create and associate a tag with the new class.

Note that when you click the arrow button to view the Type list, your new class name (**CTank**) is included (see line 5 in the following figure). Select the class name from this list.

	Name	Array Size	Type	Description	Web Data
1	valve_fill_state	3	Boolean	Fill valve state (open/close)	Server
2	valve_empty_state	3	Boolean	Empty valve state (open/close)	Server
3	valve_fill_command	3	Boolean	Fill valve command (open/close)	Server
4	valve_empty_command	3	Boolean	Empty valve command (open/close)	Server
5	Tank	3	Class: CTank	Tank Data	Server

Creating the Tank Class Tag

6. When you are done, save your work and close the worksheet.

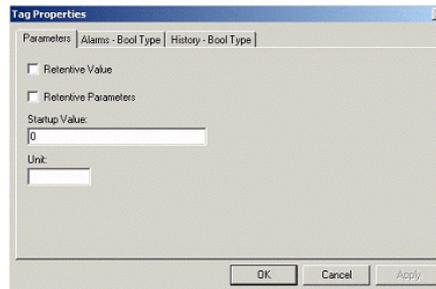
Setting Tag Properties

Understanding Tag Properties and 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, similar to the following:



Tag Properties dialog for Boolean tag

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 121.

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](#)

Using 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:

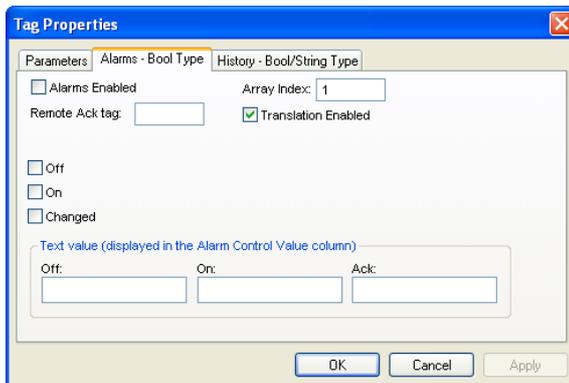
- **Alarms Enabled** checkbox: Click (check) to enable checking according to configuration.
- **Remote Ack tag** field: Type a tag to acknowledge this alarm.
- **Dead Band Value** field: Type a value of the filter for alarms normalization. For example: if the TEMP1 tag is configured with an alarm with Hi Limit = 90 and Dead Band = 5, then IWS generates an alarm when $TEMP1 \geq 90$. The return to normal occurs when $TEMP1 \leq 85$.
- **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.



Alarms-Bool Type Tab

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 acknowledgement, 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.

Using 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

Tag properties (also known as "tag fields") are metadata attached to each tag in the database. Most of these properties can be set using the *Tag Properties* dialog, which you can open by clicking the **Tag Properties** button on the Tag Properties toolbar.

To access a tag property during runtime, use the following syntax (without spaces) anywhere that you would normally specify a tag:

`tag_name->property_name`

You can access the following tag properties during runtime:

Tag Property	Description	R or R/ W	Data Type	Available on Data Type...				Retain
				Bool	Int	Real	Str	
Name	The name of the tag, as configured in the Project Tags database.	R	String, up to 32 chars	Y	Y	Y	Y	n/a
MemberName	The name of the class member, in a properly configured Class . NOTE: The syntax must be: <code>Class.Member->MemberName</code> Example: <code>Tank.Lv1->MemberName = Lv1</code>	R	String, up to 32 chars	Y	Y	Y	Y	n/a

Tag Property	Description	R or R/ W	Data Type	Available on Data Type...				Retain
				Bool	Int	Real	Str	
Size	Array Size. If the tag is not an array tag, it returns the value 0	R	Integer	Y	Y	Y	Y	n/a
Index	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] -> Index Example: Tag[1] -> Index = 1	R	Integer	Y	Y	Y	Y	n/a
Description	The description of the tag, configured in the Tags datasheet.	R	String	Y	Y	Y	Y	Y
Quality	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.	R	Integer	Y	Y	Y	Y	N
TimeStamp	Time and date when the value of the tag last changed.	R	String	Y	Y	Y	Y	N
Blocked	This property can have two values: <ul style="list-style-type: none">0: The tag is blocked and all runtime tasks will ignore it. It is effectively removed from the project database.1: The tag is unblocked and all runtime tasks can access it normally. 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.	R/W	Boolean	Y	Y	Y	Y	N
Unit	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, ps i.	R/W	String, up to 9 chars	Y	Y	Y	Y	Y
Max	The maximum value that can be written to the tag during runtime.	R/W	Real	N	Y	Y	N	Y
Min	The minimum value that can be written to the tag during runtime	R/W	Real	N	Y	Y	N	Y
B0 ... B31	Value (0 or 1) of any of the 32 bits (b0, b1, b2, ... b31) of an Integer tag. (B0: LSB B31: MSB)	R/W	Boolean	N	Y	N	N	N
DisplayValue	A converted Tag value that is only displayed on-screen: DisplayValue = (Value / UnitDiv) + UnitAdd This is used when the actual Tag values have one Engineering Unit (see Unit above) but need to be displayed on-screen in another Engineering Unit (see DisplayUnit below). For example, Celsius degrees and Fahrenheit degrees. If user input changes DisplayValue during runtime, then the conversion is reversed before the change is actually written to the Tag:	R/W	Real	N	Y	Y	N	n/a

Tag Property	Description	R or R/ W	Data Type	Available on Data Type...				Retain
				Bool	Int	Real	Str	
	Value = (DisplayValue - UnitAdd) * UnitDiv							
DisplayUnit	A brief description (up to 9 characters) of the Engineering Unit for DisplayValue . NOTE: This property can only be set by using the SetDisplayUnit and SetTagDisplayUnit functions.	R	String, up to 9 chars	N	Y	Y	N	N
UnitDiv	Number by which the Tag value is divided to get DisplayValue . To perform no division, UnitDiv should be 1. NOTE: This property can only be set by using the SetDisplayUnit and SetTagDisplayUnit functions.	R	Real	N	Y	Y	N	N
UnitAdd	Number added to the Tag value to get DisplayValue . To perform no addition, UnitAdd should be 0. NOTE: This property can only be set by using the SetDisplayUnit and SetTagDisplayUnit functions.	R	Real	N	Y	Y	N	N
DisplayMax	The maximum value that can be input to DisplayValue during runtime: DisplayMax = (Max / UnitDiv) + UnitAdd If DisplayMax is changed during runtime, then Max is also changed as follows: Max = (DisplayMax - UnitAdd) * UnitDiv	R/W	Real	N	Y	Y	N	N
DisplayMin	The minimum value that can be input to DisplayValue during runtime: DisplayMin = (Min / UnitDiv) + UnitAdd If DisplayMin is changed during runtime, then Min is also changed as follows: Min = (DisplayMin - UnitAdd) * UnitDiv	R/W	Real	N	Y	Y	N	N
HiHiLimit	Limit value for the HiHi alarm.	R/W	Real	N	Y	Y	N	Y
HiLimit	Limit value for the Hi alarm.	R/W	Real	N	Y	Y	N	Y
LoLimit	Limit value for the Lo alarm.	R/W	Real	N	Y	Y	N	Y
LoLoLimit	Limit value for the LoLo alarm.	R/W	Real	N	Y	Y	N	Y
RateLimit	Limit value for the Rate alarm.	R/W	Real	N	Y	Y	N	Y
DevSetpoint	Setpoint value for Deviation alarms.	R/W	Real	N	Y	Y	N	n/a
DevPLimit	Limit value for the Deviation+ alarm.	R/W	Real	N	Y	Y	N	Y
DevMLimit	Limit value for the Deviation- alarm.	R/W	Real	N	Y	Y	N	Y
HiHi	If 0, the HiHi alarm is not active. If 1, the HiHi alarm is active.	R	Boolean	Y	Y	Y	N	n/a
Hi	If 0, the Hi alarm is not active. If 1, the Hi alarm is active.	R	Boolean	Y	Y	Y	N	n/a
Lo	If 0, the Lo alarm is not active. If 1, the Lo alarm is active.	R	Boolean	Y	Y	Y	N	n/a
LoLo	If 0, the LoLo alarm is not active. If 1, the LoLo alarm is active.	R	Boolean	Y	Y	Y	N	n/a
Rate	If 0, the Rate alarm is not active. If 1, the Rate alarm is active.	R	Boolean	Y	Y	Y	N	n/a
DevP	If 0, the Deviation+ alarm is not active. If 1, the DevP alarm is active.	R	Boolean	N	Y	Y	N	n/a

Tag Property	Description	R or R/ W	Data Type	Available on Data Type...				Retain
				Bool	Int	Real	Str	
DevM	If 0, the Deviation- alarm is not active. If 1, the DevM alarm is active.	R	Boolean	N	Y	Y	N	n/a
AlrStatus	<p>Integer value with the status of the current active alarms associated to the tag. Each bit of this integer value indicates a specific status:</p> <ul style="list-style-type: none"> • Bit 0 (LSB): HiHi Alarm active • Bit 1: Hi Alarm active • Bit 2: Lo Alarm active • Bit 3: LoLo Alarm active • Bit 4: Rate Alarm active • Bit 5: Deviation+ Alarm active • Bit 6: Deviation- Alarm active <p>Examples: If Tag>AlrStatus returns the value 2, it means that "Hi" alarm is active. If it returns the value 3, it means that the "HiHi" and the "Hi" alarm are active simultaneously.</p> <p>If this property returns the value 0, it means that there are no active alarms associated to this tag.</p> <p>For Boolean tags, only the values 1 (bit 1), 4 (bit 2) or 16 (bit 4) can be returned.</p>	R	Integer	Y	Y	Y	N	N
Ack	<p>This property can have two values:</p> <ul style="list-style-type: none"> • 0: There are no alarms associated with this tag that require acknowledgment. • 1: There is at least one alarm associated with this tag that requires acknowledgment. <p>This works as a global acknowledge for the tag and goes to 0 only when all alarms for the tag have been acknowledged.</p>	R	Boolean	Y	Y	Y	N	N
UnAck	<p>This property can have two values:</p> <ul style="list-style-type: none"> • 0: There is at least one alarm associated with this tag that requires acknowledgment. • 1: There are no alarms associated with this tag that require acknowledgment. <p>If you manually set this value to 1, then the active alarms (if any) are acknowledged. The value of this property is always the opposite of the Ack property.</p>	R/W	Boolean	Y	Y	Y	N	N
AlrAckValue	<p>Text associated with the Acknowledged state of a Boolean tag. This text is displayed in the Value column of an Alarm/Event Control.</p> <p>You can also edit this text in the Tag Properties dialog (Alarms – Bool Type).</p>	R/W	String, up to 32 chars	Y	N	N	N	Y
AlrOffValue	<p>Text associated with the Normalized state of a Boolean tag. This text is displayed in the Value column of an Alarm/Event Control.</p> <p>You can also edit this text in the Tag Properties dialog (Alarms – Bool Type).</p>	R/W	String, up to 32 chars	Y	N	N	N	Y
AlrOnValue	<p>Text associated with the Active state of a Boolean tag. This text is displayed in the Value column of an Alarm/Event Control.</p>	R/W	String, up to 32 chars	Y	N	N	N	Y

Tag Property	Description	R or R/ W	Data Type	Available on Data Type...				Retain
				Bool	Int	Real	Str	
	You can also edit this text in the Tag Properties dialog (Alarms – Bool Type).							
AlrDisable	<p>This property can have two values:</p> <ul style="list-style-type: none"> 0: The alarms associated with this tag are enabled. This means that when an alarm condition occurs, the alarm will become active. 1: The alarms associated to this tag are disabled. This means that even if an alarm condition occurs, the alarm will not become active. 	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 runtime (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 can 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.

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:



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's no longer in use by deleting its line in the *Project Tags* 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.

1. [Stop the project](#) if it is running.
2. Open the [Project Tags datasheet](#).
3. Right-click anywhere in the datasheet and then select **Disable Sort** from the shortcut menu.
Tags cannot be deleted when the datasheet is sorted in any way.
The tags revert to their default order — that is, the order in which they were created.
4. In the datasheet, find the line for the tag you want to delete.
5. Right-click the line and then select **Delete Line** from the shortcut menu. (If the option is disabled, then you still need to disable sorting as described above.)
An alert dialog is displayed asking you to confirm the action.
6. Click **Yes**.
The line is deleted from the datasheet.
7. Save and close the datasheet.

Using the Tags Toolbar

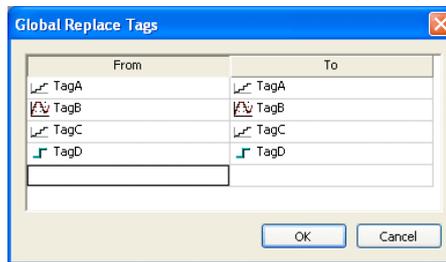
The *Tags* toolbar provides a text box and several tools (shortcuts) that enable you to create, locate, and access different tags, functions, and tag properties.



Tags tools

Global Replace Tool

When clicking on the **Global Replace** tool from the Tag Properties Toolbar, the following window displays:



Global Replace dialog

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

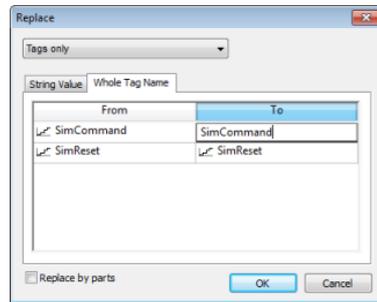
To replace all occurrences of a tag in the current document, do one of the following:

- 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.

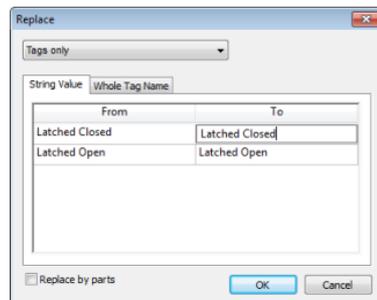
You can replace one or more tags by clicking the **Whole Tag Name** tab. Current tags used are displayed. The original tag names are shown in the **From** column on the left, and you can enter your new tag names in the **To** column on the right.



Whole Tag Name tab

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

When you are done, click **OK**.

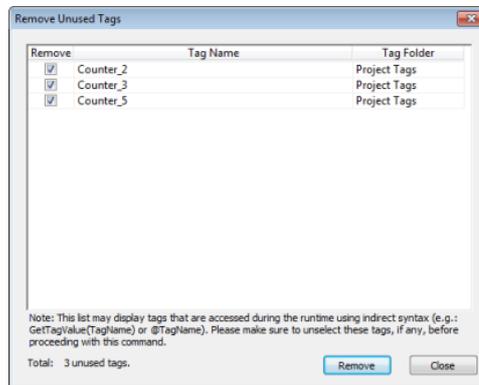
Removing unused tags from the project database

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. If it finds unused tags, then it lists them in the *Remove Unused Tags* dialog.



Unused tags listed in Remove Unused Tags dialog

- Determine which tags you want to remove, if any.
 - If you want to remove all of the listed tags, click **Remove**.
 - If you want to keep some of the listed tags, clear the **Remove** options for these tags and then click **Remove**.

Note: The listed tags may include some that are accessed during runtime using indirect syntax (e.g., `GetTagValue(TagName)` or `@TagName`, where the value of `TagName` is the name of an unused tag).

- 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.

- 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.

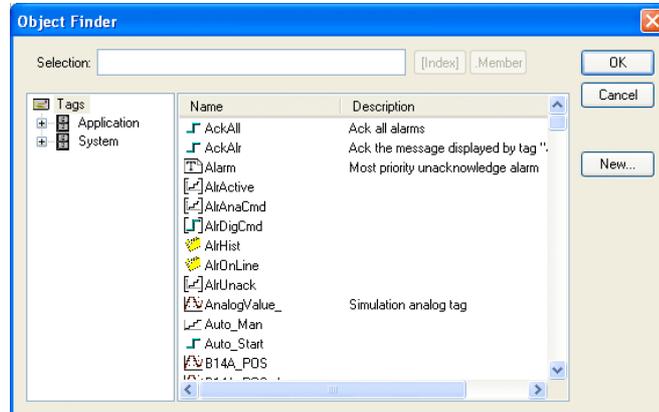
Tip: If you want to reset the project tags automatically whenever you run the project (**Run** on the Home tab of the ribbon), you can check the option **Reset Tags Database** when starting project on [the Preferences tab of the Project Settings dialog](#).

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.

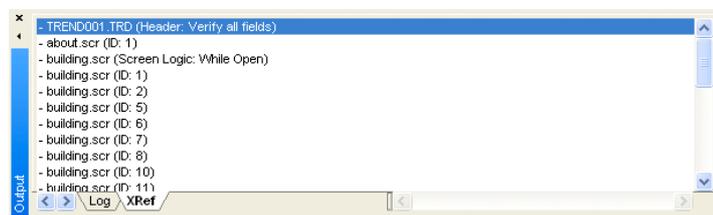


To select an existing tag/function, double-click on the tag/function name, and then click **OK** to close the box. The selected name displays in the **Tagname** text box.

- 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)

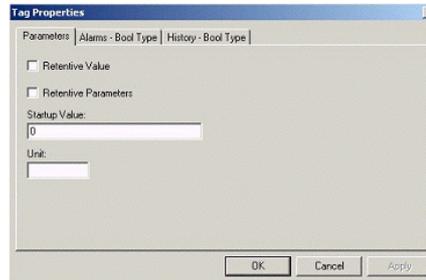
Cross Reference Tool

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:



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.

Importing an External Database

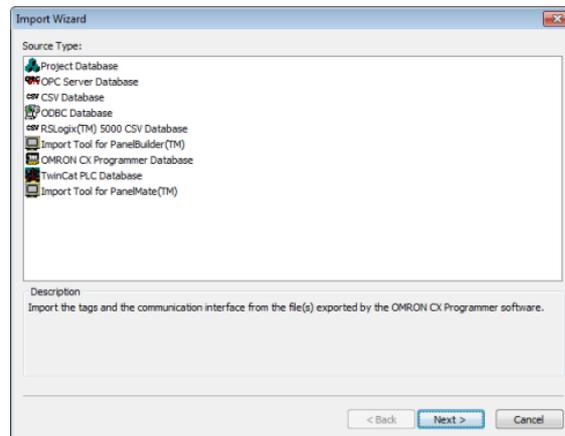
Import Wizard

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
- OMRON™ CX Programmer Database
- TwinCAT™ PLC Database
- PanelMate Plus™ Database

Step1: Select the Source Type



Import Wizard

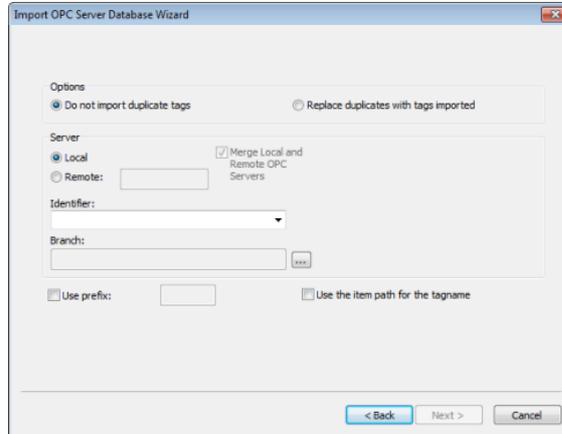
Click the data Source Type, which is where the tags are being imported from. Click **Next**.

Continue to the appropriate section for the instructions you need to complete the import database procedure:

- [Importing from Other project databases](#)
- [Importing from OPC Server Databases](#)
- [Importing from CSV Databases](#)
- [Importing from ODBC Databases](#)
- [Importing from PanelBuilder32 Databases](#)
- [Importing from RSLogix 5000 CSV Databases](#)
- [Importing from OMRON CX Programmer Databases](#)

- [Importing from TwinCAT PLC Databases](#)
- [Importing from PanelMate Plus Databases](#)

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



Import OPC Server dialog

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).

Data Source Types

Continue to the appropriate section for the instructions you need to complete the import database procedure:

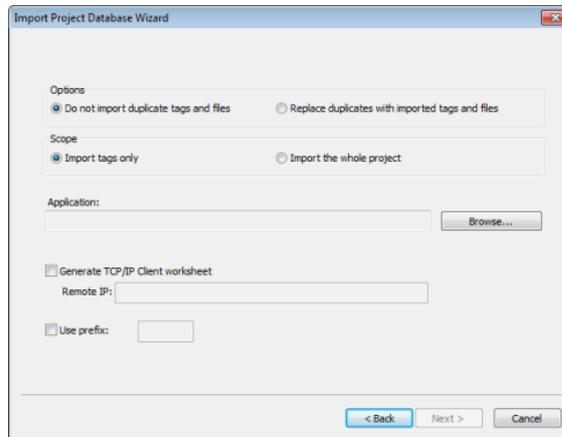
- [Importing from Other project databases](#)
- [Importing from OPC Server Databases](#)
- [Importing from CSV Databases](#)
- [Importing from ODBC Databases](#)
- [Importing from PanelBuilder Database Wizard](#)
- [Importing from RSLogix 5000 CSV Database Wizard](#)
- [Importing from OMRON CX Programmer Databases](#)
- [Importing from TwinCAT PLC Databases](#)
- [Importing from PanelMate Wizard](#)

Importing from...

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.

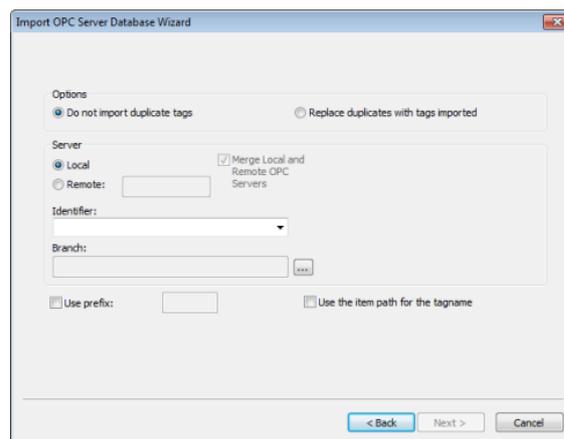
 **Note:** When you select the option to **Import the whole project**, the following worksheets will always be imported, regardless of existing worksheets with the same number in the current project: ODBC, Math, Script, Scheduler, Drivers, OPC, TCP/IP and DDE. If there are worksheets with the same number 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

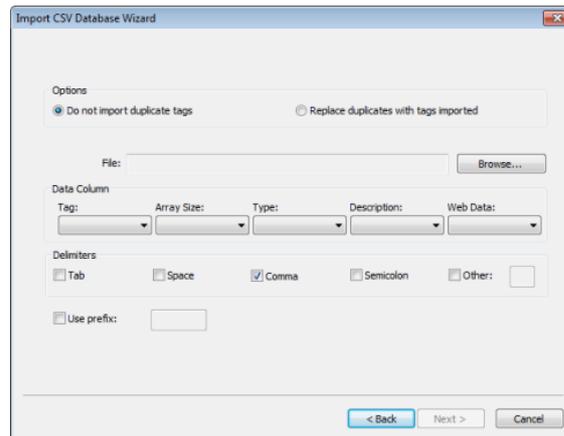
- **Local/Remote:** Provide the following options:
 - **Local:** Select this option to import tags from an OPC Server installed in the local computer.
 - **Remote:** Select this option to import tags from an OPC Server installed in a remote computer. Type the IP Address (or the host name) of the remote 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.

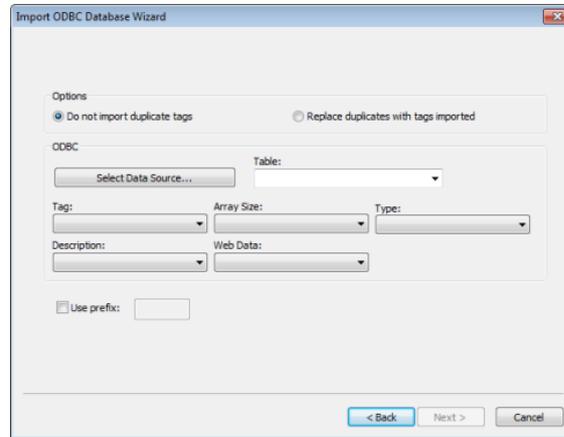
For properties that are not included in the text file, select the option **Not used**. IWS will insert defaults or leave the field blank, according to the following list:

- **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

- **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.

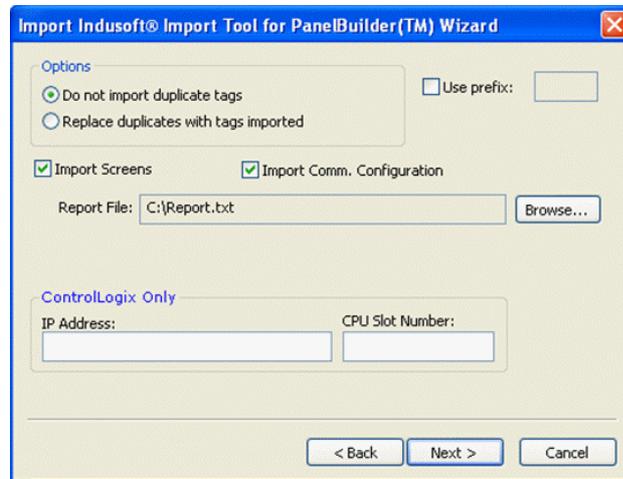
 **Note:** See Steps 1, 2 and 3 of **Import Wizard** for the settings and fields that are common for all Source Types.

IMPORTING FROM 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.

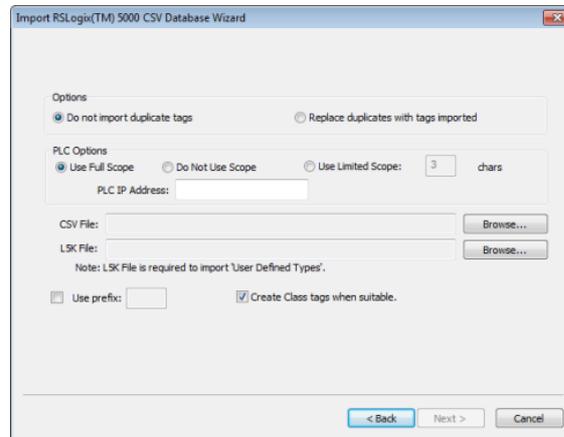
 **Note:** See Steps 1, 2 and 3 of **Import Wizard** for the settings and fields that are common for all Source Types.

 **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 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.

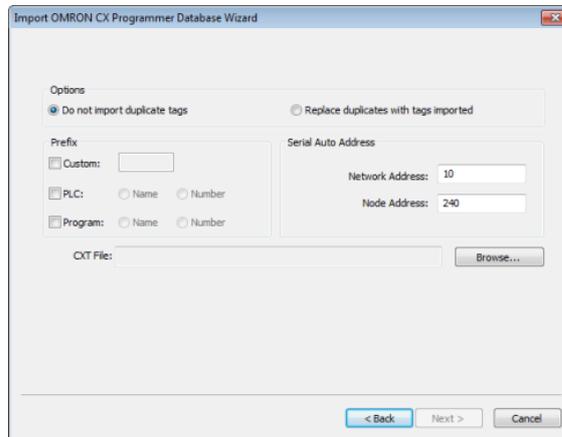
 **Note:** See Steps 1, 2 and 3 of **Import Wizard** for the settings and fields that are common for all Source Types.

IMPORTING FROM 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.

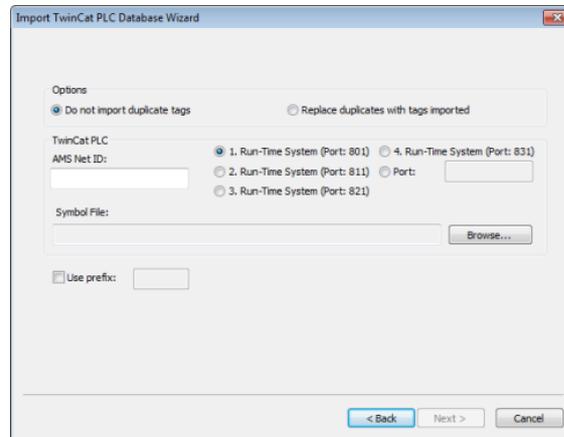
In the grid displayed in Step 3 for this Data Source Type, there is an additional field with the label **Address**, which displays the name of the tags from the CX Programmer program.

 **Note:** See Steps 1, 2 and 3 of **Import Wizard** for the settings and fields that are common for all Source Types.

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, IWS automatically creates and configures a TWCAT [driver worksheet](#), eliminating the need to manually configure the communication between IWS and TwinCAT.

 **Note:** Please refer to Step 2 of the [Import Wizard](#) instructions for settings that are common to all sources.



Import TwinCAT PLC Database Wizard

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 select one of the standard ports (e.g., 801, 811, 821 or 831), or enter 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, please see "Exporting the Symbol File from TwinCAT" below.

You are now ready to import the variables into IWS. Return to Step 3 of the [Import Wizard](#).

Exporting the Symbol File from TwinCAT

The TwinCAT development software automatically exports the program database to a "symbol file" every time you rebuild your TwinCAT project. However, TwinCAT exports the entire database by default, including many system and library variables that IWS cannot import. Before you run the import wizard, you must reconfigure your TwinCAT project options to export *only* the POU's and Global Variables and then rebuild your TwinCAT project to generate a fresh symbol file.

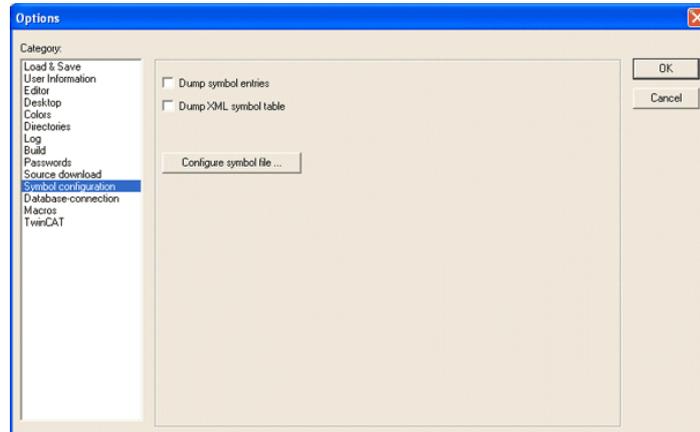
 **Note:** As of version 2.8, the TwinCAT development software exports the symbol file in both **.SYM** and **.TPY** formats:

- **.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 reconfigure your project options and generate a fresh symbol file:

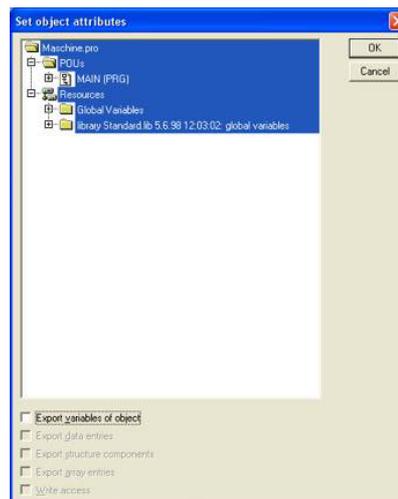
1. Open your TwinCAT project using the TwinCAT development software.
2. Choose **Project > Options** from the menu bar. The *Options* window is displayed.

3. Select **Symbol configuration** from the **Category** list:



Selecting "Symbol configuration"

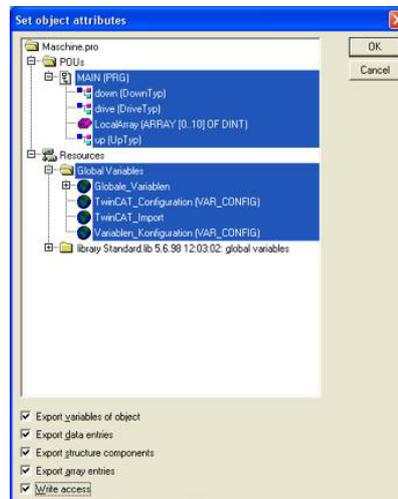
4. Click (check) the **Dump symbol entries** option.
5. Click the **Configure symbol file...** button. The *Set object attributes* dialog is displayed.
6. For the sake of expediency, you should first disable the export of all objects and then reenable only the objects that you want to export to IWS — typically, the POU's and Global Variables. Select all of the objects in the tree and uncheck all options for them at the bottom of the dialog. For example:



Configuring the symbol file

 **Note:** You may need to check **Export variables of object** in order to activate the other checkboxes and then uncheck them.

7. Reselect only the POU's and Global Variables that you want to export to IWS. **Do not select libraries.** With the objects selected, check all of the options at the bottom of the dialog. For example:



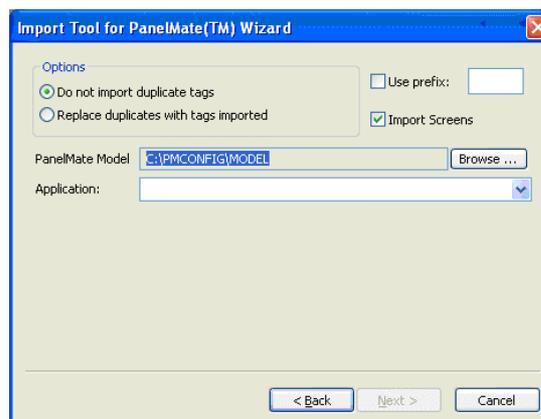
8. Click **OK** to close the *Set object attributes* dialog, and then click **OK** again to close the *Options* window.
9. Choose **Project > Rebuild All** from the menu bar. The system will rebuild the project, generating a symbol file that contains the desired POU's and Global Variables .

IMPORTING FROM PANELMATE PLUS DATABASES

 **Note:** This wizard is sold as an add-on and requires a license to be enabled. Consult your software vendor for further information.

Also, if you are running the IWS development application on a Windows operating system that has User Account Control (UAC) enabled — such as Windows Vista, Windows 7, or Windows Server 2008 — then you may problems using this import wizard. Close the development application, and then run it again as an administrator (i.e., right-click the IWS application icon, and then click **Run as administrator** on the shortcut menu).

This wizard allows you to import not only the tags, but also the screens, alarm configuration and communication interface from a project created with PanelMate Plus™ software. Using this wizard, you can convert a PanelMate™ program (developed with PanelMate Plus™) into the IWS format and run it under any platform supported by IWS.



- **Import Screens:** Check this option to import the graphical screens (including their objects and animations) to IWS.

- **PanelMate Model:** Press the Browse button to select the directory where the database files of the PanelMate Plus project that you intend to import are stored.
- **Application:** After selecting the correct path on the PanelMate Model field, the programs available in this directory will be available in this combo-box. Select the program that you intend to import before pressing the Next button.

 **Tip:** Please consult the documentation for this import wizard for detailed information about how to export an program from the PanelMate Plus™ software into IWS.

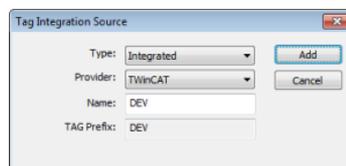
 **Note:** See Steps 1, 2 and 3 of **Import Wizard** for the settings and fields that are common for all Source Types.

Integrating the project database with a TwinCAT PLC

More than simply importing the tags and interface from a TwinCAT program, you can fully integrate your project database with a running TwinCAT PLC so that tags are synchronized between the systems, without the extra configuration required by an OPC Client worksheet.

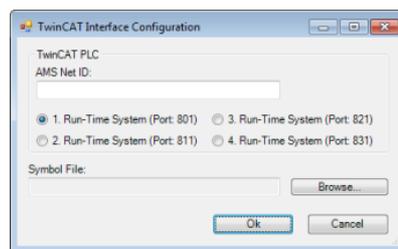
Before you can integrate with a TwinCAT PLC, you must do the following:

- Make sure the PLC is running and available on your network, and then note its AMS Net ID and runtime system port number; and
 - Export a new symbol file (*.TPY) from the TwinCAT development application containing only the program variables that your project can use, and then copy the file to a location where the development application can access it. (Typically, you should copy the symbol file to your project folder.) For more information, see [Exporting the Symbol File from TwinCAT](#).
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.



Tag Integration Source dialog

3. In the **Type** list, select **Integrated** if it's not already selected.
4. In the **Provider** list, select **TwinCAT** if it's 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, a TwinCAT PLC tag named **switch1** would subsequently be named **DEV_switch1** in your project.
6. Click **Add**.
The *TwinCAT Interface Configuration* dialog is displayed.



TwinCAT Interface Configuration dialog

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 number on which the TwinCAT runtime system has been configured to run.
9. To the right of the **Symbol File** box, click **Browse**.
A standard *Open* dialog is displayed.
10. Use the *Open* dialog to locate and select the TwinCAT symbol file (*.TPY).
11. Click **OK** to close the *Open* dialog.
12. Click **OK** to close the *TwinCAT Interface Configuration* dialog.

The integrated tags are listed in the [Shared Database](#) folder. They can be used in your project like normal tags, and they will be synchronized between the systems during project runtime.

Screens and Graphics

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).

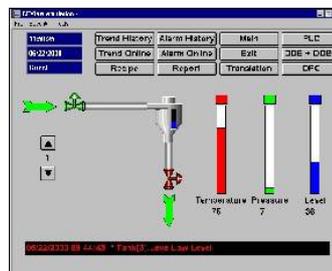
IWS allows you to create projects that can monitor processes using high-resolution color screens.

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.

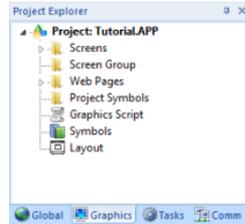


Sample CEView Emulation Screen

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

The folders on the *Graphics* tab are described on the following pages:

- **Screens** contains all of the [screens](#) created for the current project.
- **Screen Group** contains the entire [screen groups](#) (individual screens combined into manageable groups) created for the current project.
- **Web Pages** contains all of the [Web pages](#) (i.e., screens saved in HTML format) created for the project.
 - **Mobile Access** allows [configuration of the mini-site](#) that is targeted to cell phones, PDAs, and other mobile devices.
- **Project Symbols** contains all of the [user-defined symbols](#), which can be groups of images and/or text. You can create custom symbols for the project and save them into this folder.
- **Graphics Script** contains [predefined functions that are executed when certain screen actions occur](#), such as when the Thin Client is launched on a remote station.
- **Symbols** contains the [library of common symbols and graphics](#) provided with the project. Double-click the **Library** icon to open the *Symbol Library*.
- **Layout** displays all screens currently open in the Screen Editor and allows you to [visualize how the screens fit together](#) during runtime.

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

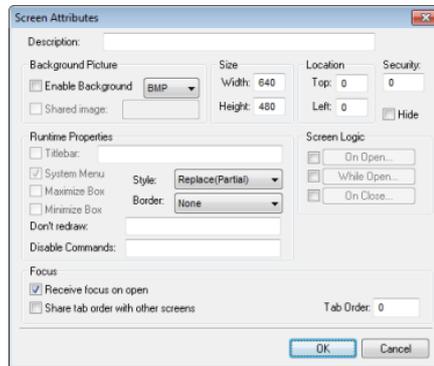
The *Screen Attributes* dialog is automatically displayed when you add a new Screen worksheet.

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

Elements in Screen Attributes dialog

Area / Element Name		Description
Description		A brief description of the project screen. This is not shown anywhere during runtime.
Background Picture	Enable Background	Enables the background picture layer and specifies the file type of the picture. 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., <code>\project_name\Screen\screen_name.BMP</code>). You can then edit this image using a third-party image editor. For more information, see Changing a screen's background color or image .
	Shared Image	Uses the specified image file located in the Screen sub-folder of your project folder. 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 <i>not</i> 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. <div style="border: 1px solid black; padding: 5px;"> Note: Only BMP files are supported in projects developed for Windows Embedded target systems.</div>
Size	Width	The default width of the screen (in pixels) when it is initially displayed. The user can change the size during runtime if the screen is set to be resizable; see Border below.
	Height	The default height of the screen (in pixels) when it is initially displayed. The user can change the size during runtime if the screen is set to be resizable; see Border below.
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.
Security		The minimum security level that a user must have to access this screen.
Hide		Keeps the screen loaded in memory after it is called the first time, so that it opens more quickly every time thereafter. Any user action or system process that would close the screen in fact only hides it. <div style="border: 1px solid black; padding: 5px;"> Caution: This option should be selected only for critical screens that must open quickly. If too many screens are kept in memory, then overall project performance will be affected.</div>
Runtime Properties	Style	The general runtime behavior of the screen: Overlapped

Area / Element Name	Description
	<p>Opens the screen without closing any other screens.</p> <p>Popup</p> <p>Forces the screen in front of all other screens but does not close them.</p> <p>Replace (Partial)</p> <p>Opens the screen and closes all other Replace and Popup screens. This is the default.</p> <p>Dialog</p> <p>Similar to Popup except that the other screens are also disabled until the dialog is closed.</p> <p>Replace (Complete)</p> <p>Similar to Replace (Partial) except that it also closes Overlapped and Dialog screens.</p>
Border	<p>The type of border around the screen:</p> <p>None</p> <p>No border; the screen is a flat, immovable rectangle on the computer display. This is the default.</p> <p>Thin</p> <p>A thin border that makes the screen a movable window. Includes title bar.</p> <p>Resizing</p> <p>A thick border that makes the screen a movable, resizable window. Includes title bar.</p>
Titlebar	<p>Shows the window's title bar with the specified window name.</p> <p>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></p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p> Tip: It is useful to specify a window name even when the title bar is not shown, because when the screen is printed, the window name is included in the page header.</p> </div>
System Menu	Provides a menu of basic window commands at the left end of the title bar.
Maximize Box	Shows the Maximize button at the right end of the title bar.
Minimize Box	Shows the Minimize button at the right end of the title bar.
Don't Redraw	While this tag/expression evaluates as TRUE, the screen's graphics are not updated.
Disable Commands	While this tag/expression evaluates as TRUE, the screen is locked against user interaction but the graphics continue to be updated.
Screen Logic	<p>On Open</p> <p>Lists expressions to be evaluated once when the screen is opened, similar to a Math worksheet.</p>
	<p>While Open</p> <p>Lists expressions to be continuously evaluated while the screen is open, similar to a Math worksheet. If you also configure a tag/expression in Trigger, 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.</p>
	<p>On Close</p> <p>Lists expressions to be evaluated once when the screen is closed, similar to a Math worksheet.</p>
Focus	<p>Receive Focus on Open</p> <p>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.</p>
	<p>Share Tab Order with Other Screens</p> <p>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.</p>
	<p>Tab Order</p> <p>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.</p>

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](#) is displayed.
3. Select the **Enable Background** option. 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`).
4. Click **OK** to close the *Screen Attributes* dialog.
5. On the Graphics tab of the ribbon, in the Screen group, click **Background Image**. Microsoft Paint is started and the BMP file is opened for editing.
6. Use Paint to edit the background image as needed.

Specifying an existing image file as the background

To select an existing image file, especially if it's in a format other than BMP:

1. Copy the image file that you want to use to the Screen sub-folder of your project folder.
2. In the development application, 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 is displayed.
4. Select the **Enable Background** option.
5. Select the **Shared Image** option, and in the box, type the name of the image file. Do not include the file extension.
6. In the list to the right of the **Enable Background** option, select the graphics format of the image file.

 **Note:** Only BMP files are supported in projects developed for Windows Embedded target systems.

7. Click **OK** to close the *Screen Attributes* dialog. If the development application can find and load the specified file, then it will be shown in the screen. If not, then a warning message will be displayed.

Screen Group Folder

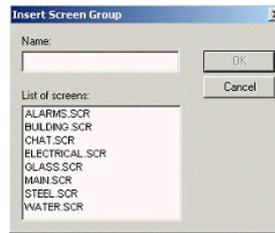
The *Screen Group* folder combines individual screens from the [Screens](#) folder into more manageable groups.

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 a 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.

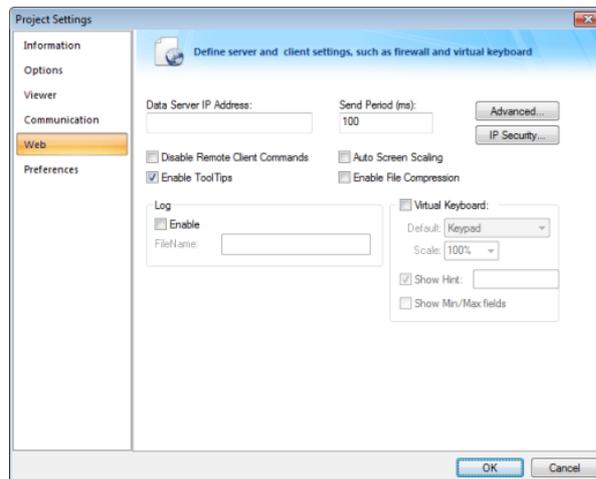
Web Pages Folder

IWS also enables you to save screens in the HTML format. You cannot create the HTML pages contained in the *Web* folder directly; instead they are generated from pre-existing screens.

To create an HTML page, you must first create a screen. Configure a screen as you usually would (create objects, add properties, and so on) but keep in mind that this screen will become a Web page. Save the screen as usual when you finish. Then, with the screen still open, click **Save As HTML** on the Application menu.

Caution: The Web pages that are generated when you click **Save As HTML** are independent of the screen files from which they were generated. Consequently, if you change that screen, the changes will not appear on the Web page until you click **Save As HTML** again.

To view your Web pages during runtime, you must first configure the project settings (**Thin Client** on the Project tab of the ribbon).



Project Settings dialog — Web tab

1. Open the dialog and type the IP address (from which to run the project) in the **Data Server IP Address** field.
2. Type a value in the **Send Period (ms)** field to specify the send period (in milliseconds) used to exchange data between the Server and the Thin Client stations.

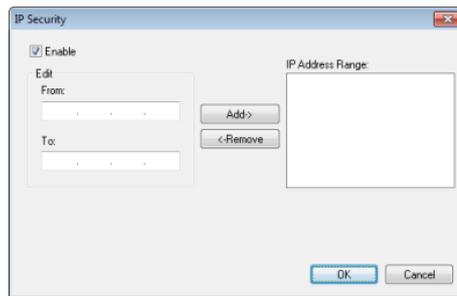
3. In the **URL** field, enter the address of the project file using the following format:

`http://IP address of the server/path from the server root to the directory containing your Web pages/project name.app`

You can also enter a local file path using the following format:

`file:///volume name/path to the directory containing your Web pages/project name.app`

4. Click (enable) the following checkboxes if applicable:
 - **Disable Remote Client Commands**checkbox: Click (enable) this box to prevent a remote client from issuing commands from your Thin Client to your Server.
 - **Enable ToolTips**checkbox: Click (enable) this box to see Windows ToolTips when viewing the project screens on the Thin Client (browser).
 - **Auto Screen Scaling**checkbox: Click (enable) this box to automatically scale screens displayed in a Browser window.
Enable this parameter if you are running remotely on a Thin Client, and you want IWS to scale screens automatically when you resize the Browser window.
 - **Enable File Compression**checkbox: Click (enable) this box to compress the files stored in the web sub-folder of your project folder. This option is useful for reducing download time, particularly if you have a slow connection between your Server and the Thin Client.
5. Click the **IP Security** button to open the IP Security dialog (see figure). Use the parameters on this dialog to specify the range of IP addresses for the computers that are allowed to access the project as Thin Clients.



IP Security dialog

When the *Edit* pane parameters become active, type IP addresses in the **From** and **To** fields to specify the IP address range. Use the **Add** and **Remove** buttons to move the IP addresses into the **IP Address Range** list. IWS permits the computers listed in this pane to access the project as Thin Clients

6. To enable logging for the Thin Client, move to the *Log* pane and click (enable) the **Enable** checkbox and type a file name into the **Filename** field to generate a log file on the Thin Client station. You can use this log file for debugging purposes.
7. Click **OK** to close the *Project Settings* dialog.

 **Note:** If you change any of the Web information on the *Project Settings* dialog, you must re-verify the project for the new setting to take effect. To verify the project, click **Verify** on the Home tab of the ribbon. (If you have any windows open in the development system, IWS prompts you close them before you verify the project).

Also, because the Web pages display information from the project through the Web server, you must be running the runtime system, Web server, and the TCP/IP server to view your Web pages.

Mobile Access

Studio Mobile Access (SMA) enables your project to send Alarms and Process information to cell phones, PDAs, and other mobile devices.

SMA is somewhat different from the traditional **Thin Client** solution, however. When you create your project for an operator interface, you do not want to worry about details like creating additional screens

that would fit on a cell phone. SMA takes care of these details and provides an easy-to-use interface for getting [alarm notifications](#) and [tag values](#) on almost any mobile device.

How It Works

When you enable the Mobile Access feature in your IWS project and then run your project, IWS creates a Collaboration Data Object (CDO) on the server and periodically refreshes it with alarm notifications and whatever tag values you choose to make available. (CDO is a Microsoft .NET technology that is used to share data between programs. It was previously known as Active Messaging.)

Once the SMA data object is in place, an ASP-powered Web application parses the data and builds lightweight pages for mobile browsers. 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 Web application and log on.

The connection between your project and the SMA data object goes both ways, so you can also acknowledge alarms and write new tag values through the Web application. These actions are recorded by the SMA data object and then passed back to your project.

If you're an experienced ASP developer, then you can modify the default Web application or build your own to access the SMA data object in new and creative ways. Doing so, however, is beyond the scope of this documentation. Please contact Customer Support for help.

Licensing

One SMA Client is included with every IWS runtime license. That means the SMA Web application will allow only one user to connect at a time. If you want the Web application 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

To enable Mobile Access and configure the data to be served:

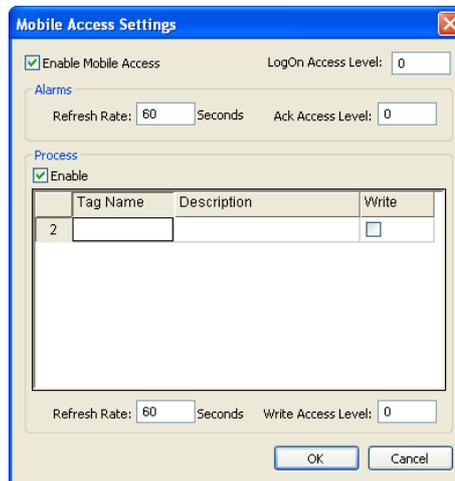
1. In *Graphics* tab of the [Project Explorer](#), open the **Web Pages** folder.



Opening the Web Pages folder

2. Double-click the **Mobile Access** icon.

The *Mobile Access Settings* dialog is displayed.



Mobile Access Settings dialog

3. Select **Enable Mobile Access**.
4. In the **LogOn Access Level** text-box, type the user security level needed to log on to the Web application. For more information about security levels, see [Security](#).
5. The Web application 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.
6. To have the Web application show tag values, select **Enable** in the **Process** group-box.
 7. 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 Web application 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 Web application.
 8. 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.
 9. 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.
 10. Click **OK** where you are done.

The following screenshot show Mobile Access enabled with a selection of tags:



Example of Mobile Access settings

Installing and Configuring IIS

Studio Mobile Access (SMA) uses Collaboration Data Objects (CDO) and Active Server Pages (ASP) to build the Web application pages for mobile browsers. The mobile browser does not need to support Java®, Flash™, or any other advanced features because the pages are built entirely on the server-side and then sent to the browser as simple HTML. The Web server, however, must support CDO and ASP, and that typically means it must be Microsoft IIS running on Windows. For more information about installing and configuring IIS, see "[Configuring a Web server to host your project pages](#)" as well as the Microsoft IIS documentation.

Accessing the Web Application

Once you've enabled Mobile Access, 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
http://server_address/SMA/LogOn.asp
- If the IIS home directory is set to the \Web\SMA sub-folder in your project folder, then the URL is
http://server_address/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 SMA 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 Web application.

This menu is also displayed in the Alarms and Process pages described below.

Alarms

Alarms

[Home](#) [Process](#) [Log Off](#)

State	Message	Type	Time
	Hi Pressure	HiHi	14:59
	Low Temperature	Lo	14:44
	Hi Level	HiHi	14:59

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

Process

[Home](#) [Alarms](#) [Log Off](#)

Description	Value
Pressure Set Point	27
Temperature Set Point	90
Level Set Point	60
Main Motor	On

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.

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 (see figure on the left) provides tools for general screen editing.



Editing group

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

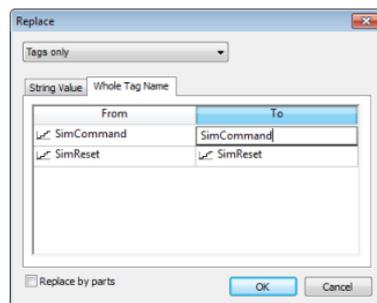
To replace all occurrences of a tag in the current document, do one of the following:

- On the Home tab of the ribbon, in the Tags group, click **Replace**; 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.

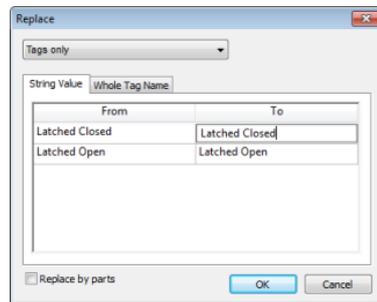
You can replace one or more tags by clicking the **Whole Tag Name** tab. Current tags used are displayed. The original tag names are shown in the **From** column on the left, and you can enter your new tag names in the **To** column on the right.



Whole Tag Name tab

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

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:

Area / Element Name	Description
Pushpin	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 <i>Object Properties</i> window even when you click the objects on the screen.
Replace	Launches the <i>Replace</i> dialog, where you can replace strings, tags or properties for the selected object or group of objects.
Hint	<p>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.</p> <p>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.</p> <p>To show hints/tooltips during runtime, the Enable Tooltip option must be selected in the <i>Project Settings</i> 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).</p>
Button	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.

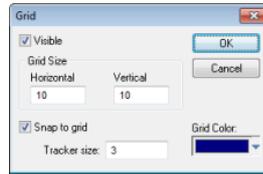
GRID SETTINGS

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

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

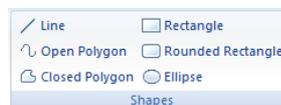
 **Note:** This tab is available only when you've selected one or more objects in a project screen.

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.



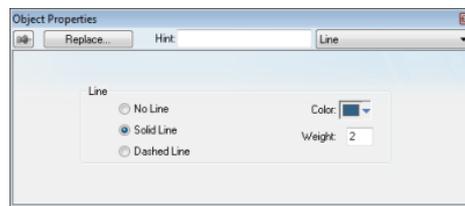
Shapes group

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.

- To view the object properties, double-click on the object. The *Object Properties* dialog displays as follows.



Object Properties: Line

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.

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.

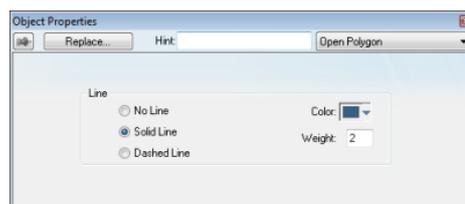
To draw an open polygon in the drawing area:

- Click the left mouse button to set the starting point of the polygon.
- Move the cursor to a new location and click again to place the second vertex.
- Repeat this process until you create the desired polygon shape.
- 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: 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.

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.

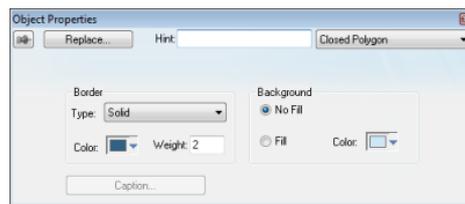
To draw a closed polygon in the drawing area:

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 displays as follows.



Object Properties Dialog: Closed Polygon

Use the *Object Properties* dialog to specify the following parameters for the polygon:

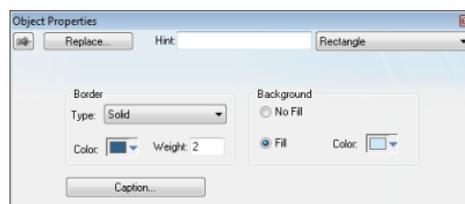
- **Line:** Specify a border line style by clicking the **No Line**, **Solid Line**, or **Dashed Line** button.
- **Color:** Specify a border line color by clicking the **Color** button. When the *Color* dialog opens, click a color to select it and then close the dialog.
- **Weight:** Specify the borderline width (in pixels) by typing a number representing the line width into the text box.
- **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.



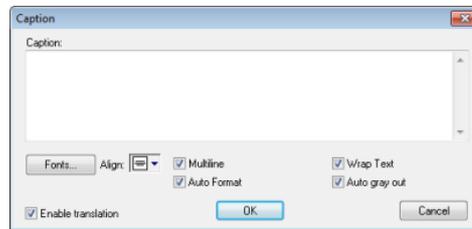
Object Properties: Rectangle

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.
- **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:



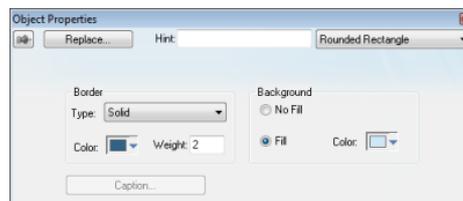
Caption dialog

- **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

Note: You cannot use the rounded rectangle tool to create a bar graph for projects running on Windows Embedded target systems.

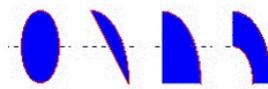
Tip: A rounded rectangle has one extra handle in the bottom-right corner, which enables you to modify the arc angle.

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.
- **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).

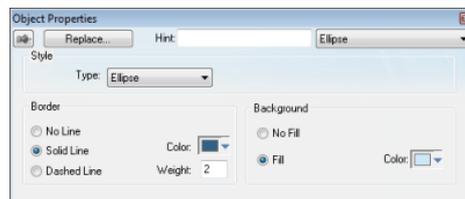


Ellipse, Chord, Arc, and Ring

 **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

Use the *Object Properties* dialog to specify the following parameters for the ellipse:

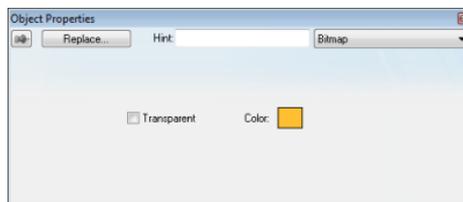
- **Style:** Specify the object style by selecting **Ellipse**, **Arc**, **Chord**, or **Ring** from the drop-down list. Next, select **Left-Bottom**, **Left-Top**, **Right-Bottom**, or **Right-Top** from the **Style** list to choose the quadrant into which the ellipse is drawn.
For example to represent 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.
- **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.
- **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.

ADDING AN IMAGE TO A SCREEN

To add an image to 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.

Tip: Please note that using this method to add an image to the screen will embed the image data in the Screen worksheet file. If you want to link to an external image file, so that you can change the image during project runtime, use a [Linked Picture](#) screen object instead.

1. Open the desired image in an appropriate image editing application, such as Microsoft Paint.
2. Select part or all of the image, either by using the **Select** tool or by pressing **Ctrl-A**.
3. Copy the image selection to the clipboard, either by using the **Copy** tool or by pressing **Ctrl-C**.
4. Switch to the IWS development application.
5. On the **Home** tab of the ribbon, in the **Clipboard** group, click **Paste**, or press **Ctrl-V**. The image selection is added to the worksheet as a Bitmap screen object.
6. Double-click the screen object. The *Object Properties: Bitmap* dialog is displayed.

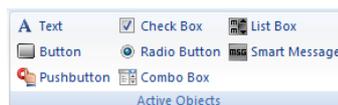


Object Properties: Bitmap

7. If you want the image to be transparent to the screen background, select **Transparent** and then use the color picker to specify which color should be transparent.
8. Close the *Object Properties* dialog.

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.

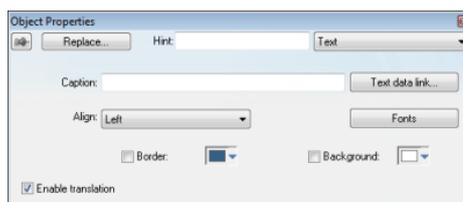


Active Objects group

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:

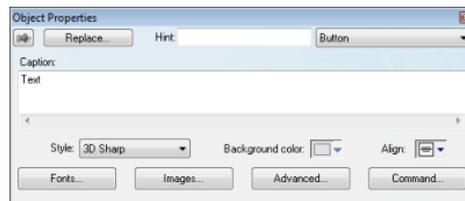
- **Caption:** Specify a text string by typing a caption in the text box.
- **Text data link button:** Click to apply the [Text Data Link animation](#) to the Text object.
If the caption doesn't include any placeholder characters (###) for the text-data link, then clicking this button also automatically 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.

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.

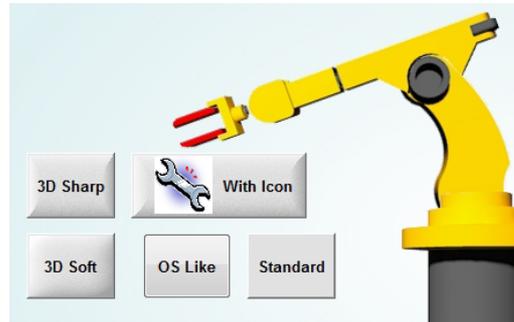


Object Properties: Button

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 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.
 - 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



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**).

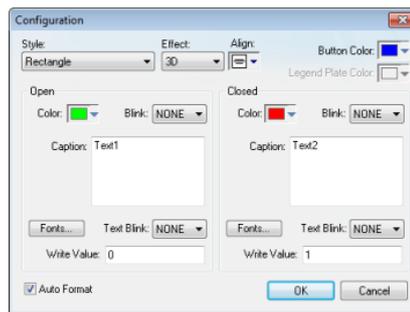
Click the button to toggle between its default and non-default state (according to its specified **Type**). For example, in the button's initial state, it may conform to characteristics specified in the **Open** area of the *Configuration* dialog (see below). Click the button again to toggle to the opposite 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.
- **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:

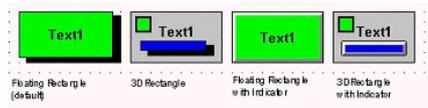


Configuration dialog

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

- **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.

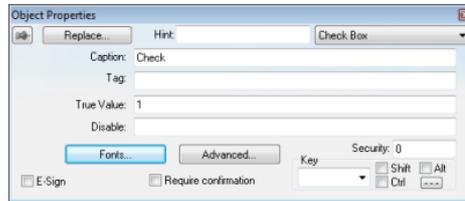
CHECK BOX OBJECT

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.



Object Properties: Check Box

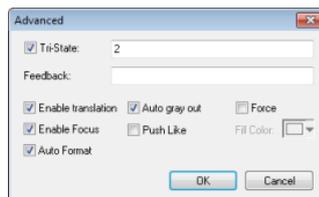
 **Tip:** To change the default size of the check box, edit your project file (*project_name.app*) to add the following setting:

```
[Objects]
CheckBoxSize=height_in_pixels
```

Doing this will change the size of all check boxes in your project, so be careful. Also, this only works for projects running in the project viewer (Viewer.exe); it does not work for projects running in CView or the Web thin client.

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.
- **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



Tip: To change the default size of the radio button, edit your project file (*project_name.app*) to add the following setting:

```
[Objects]
RadioButtonSize=height_in_pixels
```

Doing this will change the size of all radio buttons in your project, so be careful. Also, this only works for projects running in the project viewer (Viewer.exe); it does not work for projects running in CEView or the Web thin client.

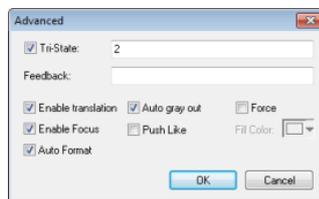
Use the *Object Properties* dialog to specify the following parameters for the radio button object:

- **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.

- **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

- **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

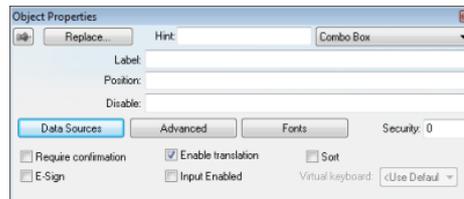
The Radio Button object can operate in two different modes: Normal and Tri-State. For more information, see [Modes of operation for Check Box and Radio Button objects](#).

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 than 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.

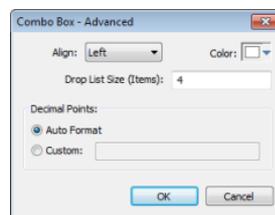
Double-click on the combo-box object to open the *Object Properties* dialog.



Object Properties: Combo Box

You can use this dialog 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 (see below).
- **Advanced** button: Click to open the *Combo Box - Advanced* dialog:



Combo Box - Advanced

- **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 opens, click a color to select it, then click **OK** to close the dialog.

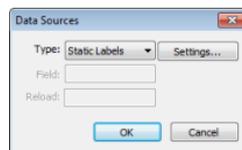
- **Drop List Size (Items)** field: Enter an integer (or a tag of [Integer](#) type) to specify the number of items that should be displayed at one time when the user clicks on the combo box. The higher the number of items, the longer the drop list will appear.

 **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](#). Use this dialog 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 Tool](#).
- **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 to configure the items/labels that will be displayed in the Combo Box object.

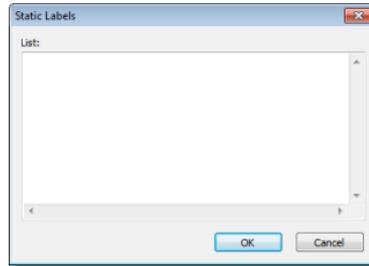


Data Sources dialog

- **Type** combo box: Select the type of data source that you want to use, and click the **Settings...** button 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

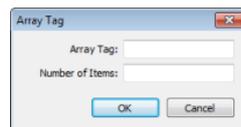
Enter your labels — with one label per line — just as if you were editing a plain text file. You can include tags or expressions in a label by enclosing them in curly brackets. For example: Tank #1 in area {AreaNameTag}.

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

- **Array Tag:** Enter the name of an [array tag](#) of [String](#) type 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

- **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.)

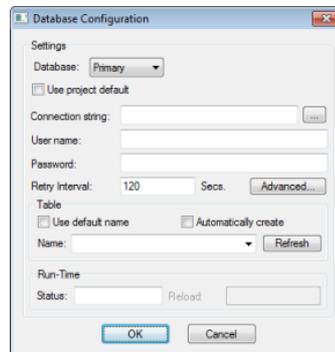
 **Tip:** You can configure tag names between curly brackets (e.g., {TagName}) in the **File** field.

- **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

Please refer to the [Database Configuration dialog](#) for further information about this dialog.

LIST BOX OBJECT

On the **Graphics** tab, in the **Active Objects** group, click **List Box** to create a List Box object on your screen. 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.

You can select a message from the active list box during runtime and write the message value to a tag. (If a list is too long to fit within the viewable area of a List Box object, the object provides scroll bars.)

Use the **Enter Req'd** box on the *Object Properties* dialog to configure selected messages as follows:

- Check (enable) the **Enter Req'd** box and use the keyboard/keypad keys, list control objects from the **Library**, pointing devices, or user-defined keys containing the `PostKeys()` function to scroll through the message list. Then, use the Enter key to select the message and write its value to the write tag. You can use the Esc and Tab keys to return to the previously selected message at any time prior to pressing the Enter key.
- Uncheck (disable) the **Enter Req'd** field to write the value of a selected (highlighted) message the write tag automatically.

To add List Box objects to a screen:

1. On the Graphics tab of the ribbon, in the Active Objects group, click **List Box**.
2. Click in the screen and drag to create/adjust an expanding rectangle.

Height and the font size determine how many messages are visible.

Width determines how much of the message length is visible.

After creating a rectangle, 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

Tip: You also can open the *Object Properties* dialog, by right-clicking on the List Box object or by highlighting the object, pressing the Alt+Enter keys, and selecting **Properties** from the resulting shortcut menu.

You can use this dialog to specify the following parameters:

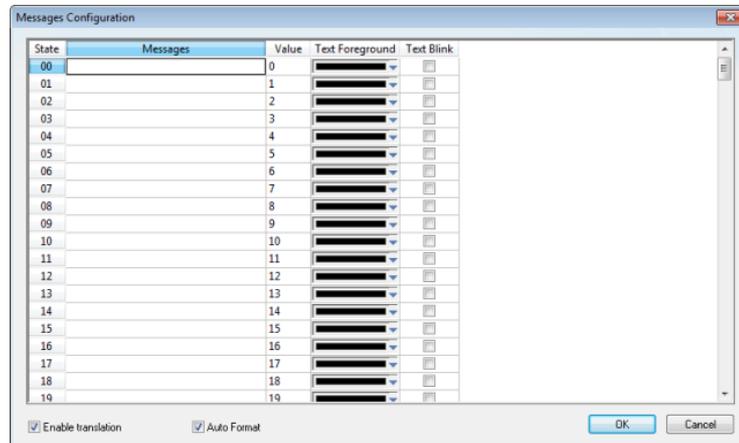
- **Value** drop-down list (located below the **Replace** button): Click to select one of the following the tag values used to index the message list.
 - **Boolean**
 - **Integer** (default)
 - **LSB** (Least Significant Bit)

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.
- **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 allows you to select messages using the Enter key only. It prevents the Tab key from selecting messages.
- **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

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 255 valid states, labeled 1 to 255
 - **LSB**: Provides 32 valid states (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 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

On the **Graphics** tab, in the **Active Objects** group, click **Smart Message** to create one or more smart message objects, which you can use to display messages and graphics based on tag values when you run your project. IWS provides the following smart message object types:

- **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 runtime.

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. Click the **Smart Message** button and position the mouse on the screen.
2. Click and drag to create (and adjust the size of) a rectangle.

You use the rectangle's size and font size to determine how much text and how large a bitmap image you can display on the screen. Later, you can change the rectangle's size and font characteristics to allow longer messages to display in a given space.

3. Double-click on the object to open the *Object Properties* dialog.



Object Properties: Smart Message

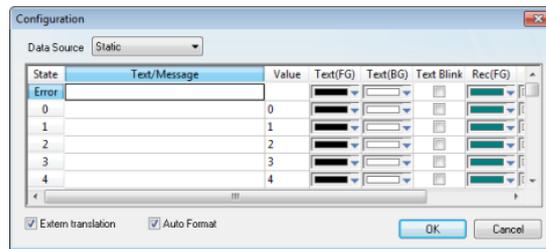
You can use this dialog to specify the following parameters:

- **Type** combo-box: Click to select the smart message object type. The object type sets the behavior of the object during runtime and the features supported by it:
 - **Message Display** (default)
 - **Multistate Indicator**
 - **Multistate Pushbutton**
- **Value** 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 (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.

- **Read Tag/Expr** text box: Enter a project tag or expression. The value determines which message is displayed by the object during runtime.
- **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.
- **Align:** Select the alignment of the text displayed by the Smart Message object.
- **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 runtime.
- **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.
- **E-Sign** (available for **Multistate Pushbutton** only): When this option is checked, the user will be prompted to enter the Electronic Signature before executing the animation.
- **Security** (available for **Multistate Pushbutton** only): System Access Level required for the object/animation.
- **No line:** When this option is checked, the line border of the object is not visible.
- **Line Weight:** Defines the thickness of the line drawn around the object (the border).
- **Fonts:** Launches the Fonts dialog, where you can configure the font settings for the text displayed in the object.
- **Config...:** Launches the *Configuration* dialog, where you can configure the messages for the object, as follows:



Smart Message Configuration Dialog

- **Data Source:** The messages displayed by the object can be either configured directly on the object (**Data Source = Static**) or can be read from an external text file (**Data Source = Text File**). When the **Data Source = Static**, the *Configuration* dialog is displayed as pictured above, and you can configure all the settings on the grid. When the **Data Source = Text File**, the *Configuration* dialog displays a field for entering the path and name of the file from which the messages will be read (the source file). See "Source File Format" below for more information about the format of the text file supported by the Smart Message object when the **Data Source = Text File**.
- **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 tag of **Real** type (see **Text/Message** below), then the value will be formatted according to the virtual table created by the [SetDecimalPoints](#) function.

The following table describes the meaning of the properties associated with each message, regardless of the Data Source:

Property	Description
Text/Message	Message (text) that will be displayed when selected during runtime. You can include tags in a message by enclosing them in curly brackets (e.g., { tagname }).
Value	You must associate a unique value with each message. during runtime, the object will display the message associated with the value that matches the value of the tag configured in the Read Tag field. If there is no such message, the message configured in the first row (State = Error) displays during runtime. When the object Type is set as Multistate Pushbutton , the value associated with the current message is also written to the tag configured in the Write Tag field (if any).
Text (FG)	Foreground color for the messages displayed during runtime.
Text (BG)	Background color for the messages displayed during runtime.
Text Blink	If checked, the message text will blink during runtime.
Rec (FG)	Line color (Border) for the rectangle behind the message.
Rec (BG)	Background (Fill) color for the rectangle behind the message.
Rec Blink	If checked, the rectangle behind the message will blink during runtime.
Graphic File	Path and name of the bitmap file (*.BMP) (if any) that will be displayed when the message associated with it is selected during runtime. If you do not specify the path, the bitmap file must be stored in your project folder.
Transparent	Select the color that will be transparent in the graphic file, if the En. Transparent checkbox is checked.

Property	Description
En. Transparent	If checked, the color selected in the Transparent field will be set as transparent in the graphic file.

 **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 = 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 runtime, 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:

Column #	Property	Default Value
1	Text/Message	-
2	Value	-
3	Text (FG)	0
4	Text (BG)	16777215
5	Text Blink	0
6	Rec (FG)	8421376
7	Rec (BG)	16777215
8	Rec Blink	0
9	Graphic File	-
10	Transparent	0
11	En. Transparent	0

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 runtime.
- 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 = **Unchecked**; 1 = **Checked**), or with the keywords **FALSE** or **TRUE** (**FALSE** = **Unchecked**; **TRUE** = **Checked**).

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
```

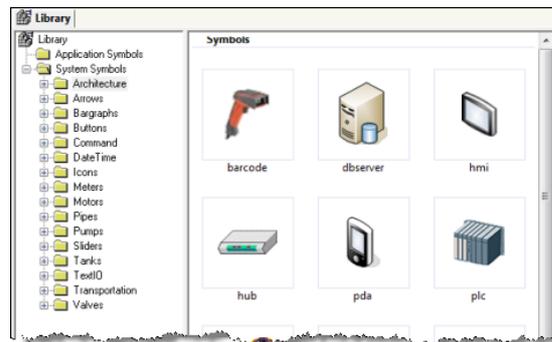
 **Tip:** You can use the *Smart Message* editor (**Data Source = Static**) to configure the messages, values and colors. To do so, select the configuration, copy it and paste it into an Excel worksheet. Then, you can save the Excel worksheet as a CSV file (**File > Save As**). This procedure provides you with a user friendly interface for configuring the color codes.

Libraries

SYMBOLS

The **Library** is a visual browser for all of the [Symbols](#) that are available to be inserted in a project screen. To open the *Library* window, do one of the following:

- On the Graphics tab of the ribbon, in the Libraries group, click **Symbols**; or
- In the Graphics tab of the Project Explorer, double-click **Symbols**.



Symbols Library window

The Library is divided into two main directories: the *Project Symbols* directory contains the same user-made Symbols that are saved in the [Symbols folder](#) of the current project; and the *System Symbols* folder contains all of the premade Symbols that are installed with IWS, sorted by category (e.g., Buttons, Meters, Tanks).

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 *Library* window will automatically close and 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, please see [Inserting a Symbol in a Screen](#).

Making a User-Made Symbol Available to Other Projects

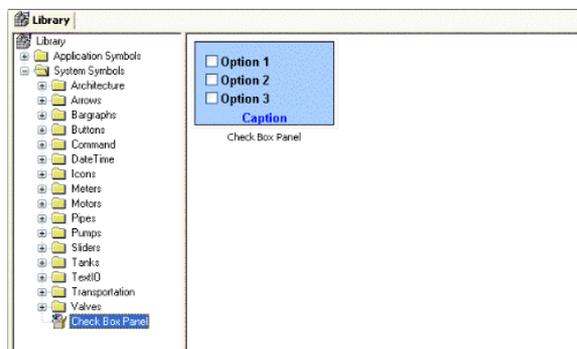
User-made Symbols are normally available only in the project where they were initially created and saved — that is, the *Project Symbols* directory of the Library is specific to each project. However, you can send a user-made Symbol to the *System Symbols* directory, to make it available to all projects:

1. Right-click on the Symbol file (**.sym**) in the *Symbols* folder and choose **Send to System Symbols** from the shortcut menu. A standard *Save As* dialog is displayed, pointing to the **\Symbol1** directory of IWS (instead of the **\symbols** directory of the current project).



Saving a Symbol

2. Choose a location to save the Symbol file. You can choose one of the existing categories/directories, or you can create a new one.
3. Click **Save**. The Symbol file is saved chosen location and the Symbol is displayed in *System Symbols* directory of the Library.



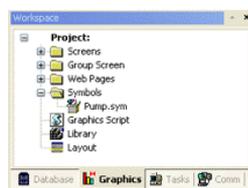
Saving a Symbol

Saving your own project symbols

A **Symbol** is a set of one or more Objects that is saved to the *Symbols* folder (in the *Graphics* tab of the Project Explorer), so that you can reuse it again and again in your projects.

Every time you reuse a Symbol, you actually make a copy that is linked to the master **Symbol** file in the *Symbols* folder. (These linked copies are also called "instances" of the Symbol.) Thereafter, if you make any changes to the Master Symbol, then 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.

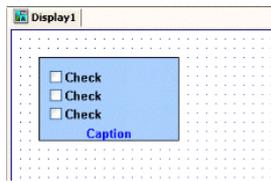


Symbols Folder in the Graphics Tab

Creating a Master Symbol

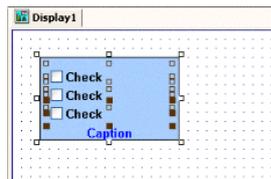
To create a master Symbol and save it to the *Symbols* folder:

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 Screen

2. Select the Object(s) or [Group](#) that you want to save as a Symbol.

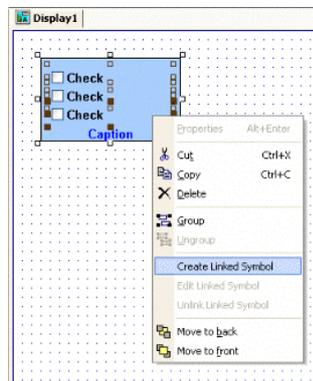


Selecting the Objects

 **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.

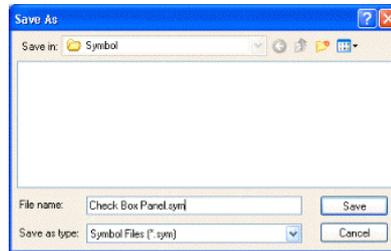
There is a situation, however, where you may want to group the Objects first. A Symbol can have only one Hint. If more than one Object has a Hint configured on it (in the [Object Properties](#)), those Hints are not shown when the Objects are saved together as a Symbol. To specify a Hint for the Symbol 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

4. A standard *Save As* dialog 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

5. Click **Save** to save the file. The Symbol appears in the *Symbols* folder, in the *Graphics* tab of the Project Explorer.



Symbol File in the Project Explorer

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.

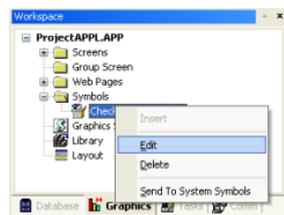
Editing the Master Symbol

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 "Creating 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.

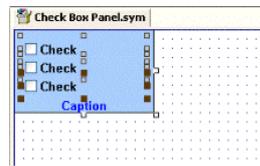


Editing the Symbol File

 **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,

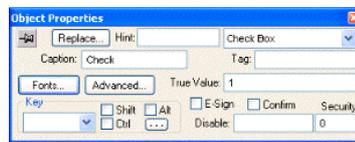
move or delete Objects in the Symbol Editor, then you may change the size or shape of the Symbol and disrupt the layout of any Screens where it is used.



Symbol File Opened for Editing

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 for the First Check Box

3. In any field where you would normally configure a tag, expression, or value, you can instead define a Custom Property using the syntax...

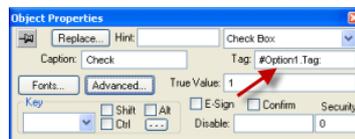
#[Category.]Property:[Value]

...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, then 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.

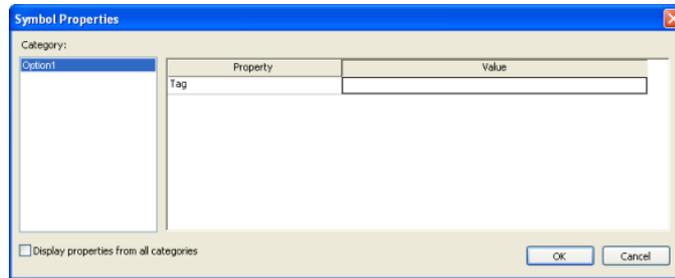
 **Note:** All Tag/Expression syntax rules apply to *Value*, including tag [names](#), [pointers](#), [arrays](#), strings, numerical and boolean values, and scripting functions.

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 Field

When you go to complete the properties on an instance of the Symbol, **#Option1.Tag:** will appear like this:

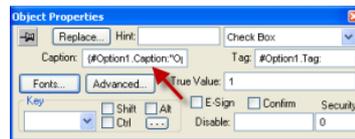


Custom Properties on a Symbol

But more about that later...

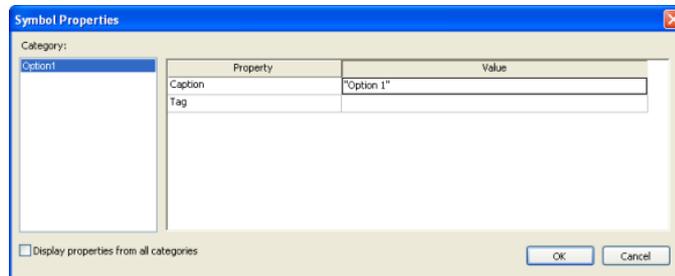
- Depending on the context, some Object properties require a specific type of value like a String, a Boolean or a numerical value. For these properties, you must enclose the Custom Property declaration in curly brackets ({}).

In this example, the **Caption** field requires a String, so type **{#Option1.Caption:"Option 1"}** as shown.



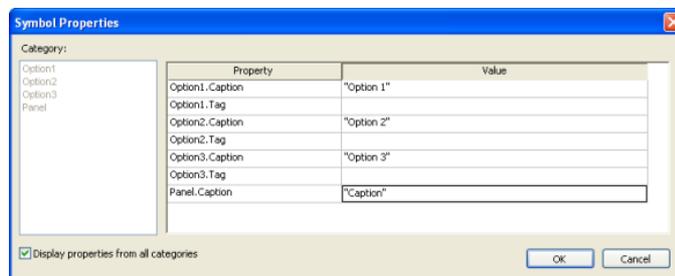
Defining a Custom Property for the Caption Field

Again, when you go to complete the properties on an instance of the Symbol, they will appear like this:



Custom Properties on a Symbol

- 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:

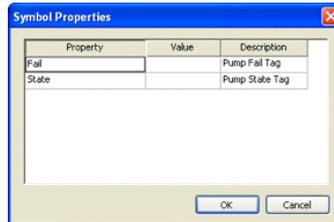


Custom Properties on a Symbol

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.

Adding 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:

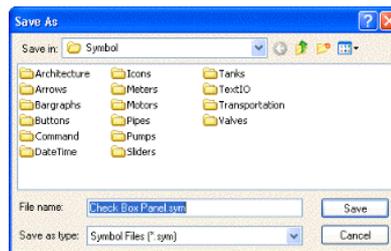


Tooltip Showing Description of the Property

Making 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:

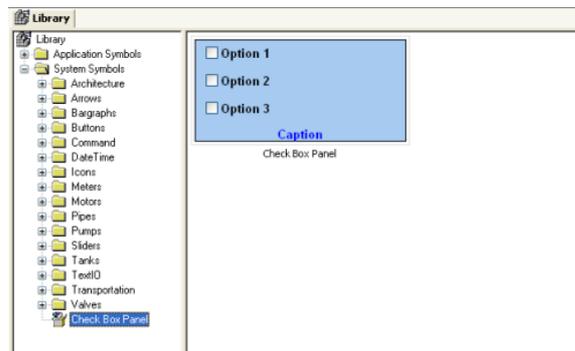
1. In the *Symbols* folder of the *Project Explorer*, right-click the desired Symbol file (**.sym**) and then choose **Send to System Symbols** from the shortcut menu. A standard *Save As* dialog is displayed, automatically pointing to the **\symbol** sub-directory of the IWS program directory instead of the **\symbol** sub-folder of your project folder.



Saving a Symbol

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 Library.



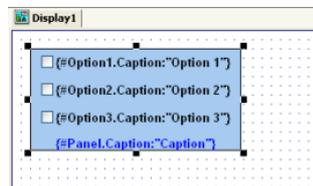
Saving a Symbol

For more information, see [Using the Library](#).

Inserting a Symbol in a Screen

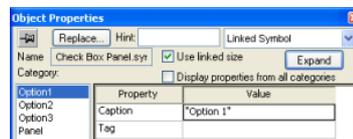
To insert a Symbol in a Screen and then complete its Custom Properties:

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:



Symbol Placed in a Screen

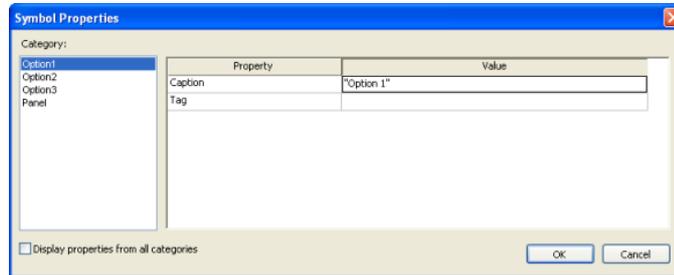
Once the Symbol is inserted, you can manipulate it like any other object in the screen. You can [align and distribute](#) it with other Objects, and you can apply [Animations](#) to it. However, the first thing to do is complete the Custom Properties for this instance of the Symbol.



Object Properties dialog for the Symbol

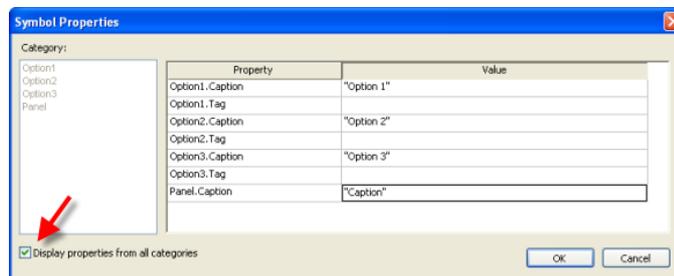
4. Open the Object Properties for the Symbol.

- Click **Expand** to open the *Symbol Properties* dialog.



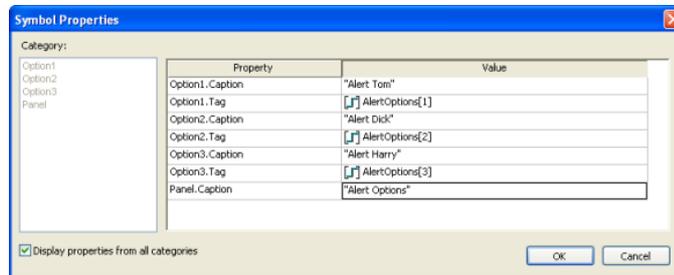
Symbol Properties dialog for the Symbol

To see all of the properties at the same time, select the **Display properties from all categories** check box.



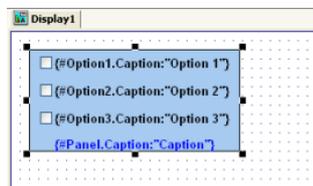
Displaying Properties from All Categories

- 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

- Click **OK** to close the *Symbol Properties* dialog, and then close the *Object Properties* dialog. The Custom Properties are resolved during runtime, as shown below.





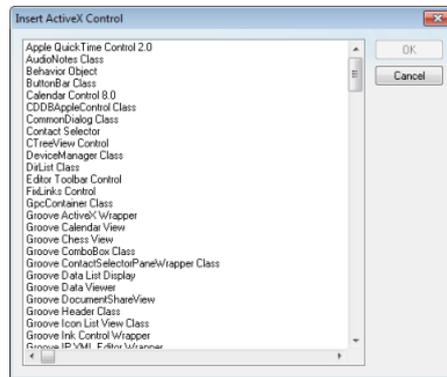
Symbol During Editing (top) and During Runtime (bottom)

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, then those changes automatically propagate to every instance.

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

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
RegFileN=name of a specific .OCX or .DLL file within the .CAB file; see below
...

[UsedControln]
...
```

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:

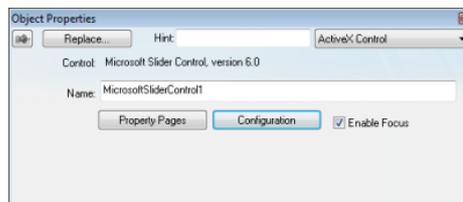
- **Properties:** Variables whose values can be read and/or written for your project (e.g., Object Color, FileName, URL, and so forth)
- **Methods:** Functions from the ActiveX object that can be triggered by your project (e.g., open a dialog, execute a calculation, and so forth)
- **Events:** Internal 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.



Object Properties: ActiveX Control

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 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.

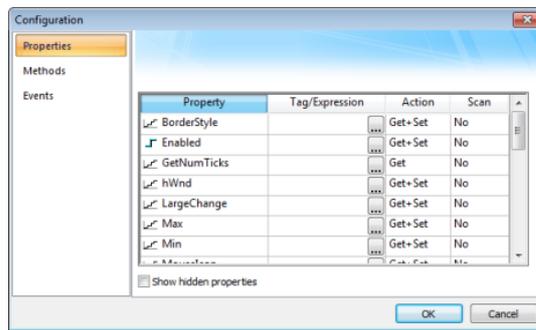
The following sections describe how to configure these interfaces.

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:

Property Icon	Property Type
	Boolean
	Integer
	Real
	String

- **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:

Action	Description
Get	Read the value of the ActiveX property and write it to the tag configured in the Tag/Expression field.
Set	Write the value from the tag or expression configured in the Tag/Expression field into the ActiveX property.
Get+Set	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.
Set+Get	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.

Note: When the value of the property is "Read-only" (cannot be overwritten by your project), the **Action** field is automatically set to **Get**.

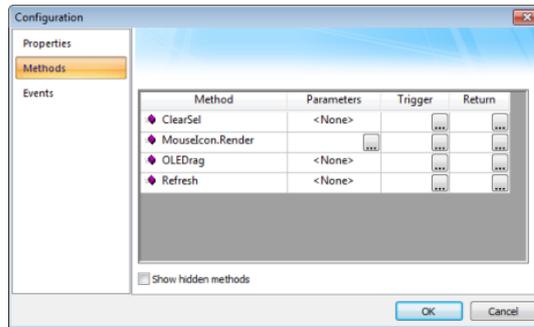
- **Scan:** Defines the polling method to get values from the ActiveX properties, according to the following table:

Scan	Description
No	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.
Always	IWS keeps polling the value of the ActiveX property and updating the tag configured in the Tag/Expression field with this value.

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:



Configuration Dialog – Methods Tab

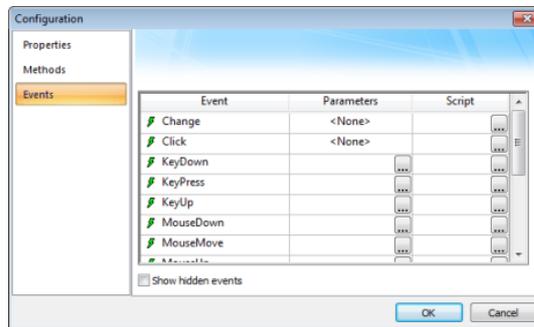
- **Method:** List all methods available from the ActiveX object.
- **Parameters:** The tags configured in this field are associated with the parameters of the method of the corresponding ActiveX object. If the method does not support any parameter, the fixed text <None> is displayed in the Parameters 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:



Configuration Dialog – Events Tab

- **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.

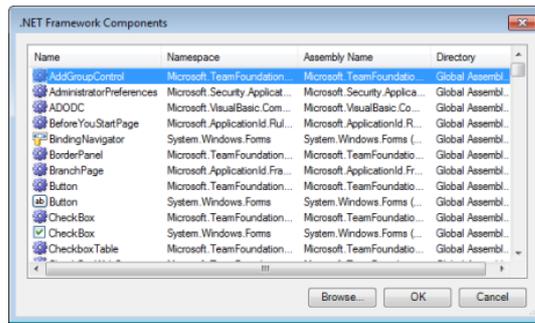
 **Caution:** When using .NET Components in your project, make sure that the target system (runtime station) can support the same components and that they are properly installed and registered. Your project includes links to the .NET Components; however, the installation of these components on the target 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

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:

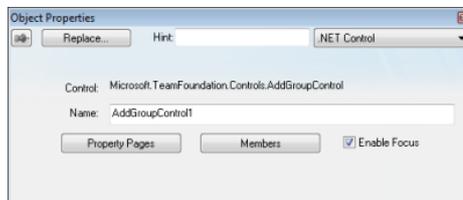
```
[Execution Environment]
DotNetControlPath=OptionalPath
```

For example:

```
[Execution Environment]
DotNetControlPath=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

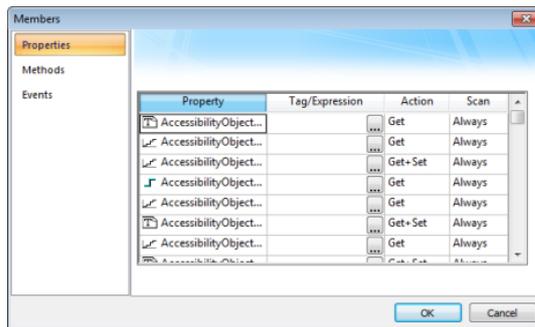
The following sections describe how to configure these interfaces.

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 varies 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:



Members Dialog – Properties tab

- **Property:** List all properties available from the .NET Component, and indicate their types:

Property Icon	Property Type
	Boolean
	Integer
	Real
	String

- **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:

Action	Description
Get	Read the value of the property and write it to the tag configured in the Tag/Expression field.

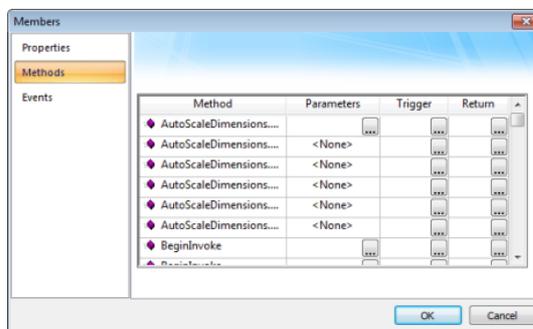
Action	Description
Set	Write the value from the tag or expression configured in the Tag/Expression field into the property.
Get+Set	Executes both actions (Get and Set). However, when opening a screen with the .NET Component, IWS executes the Get command before executing any Set command. That is, the 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.
Set+Get	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.

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 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.

Configuring Methods

The *Methods* tab provides a grid with the following fields:



Members Dialog – Methods tab

- **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**.

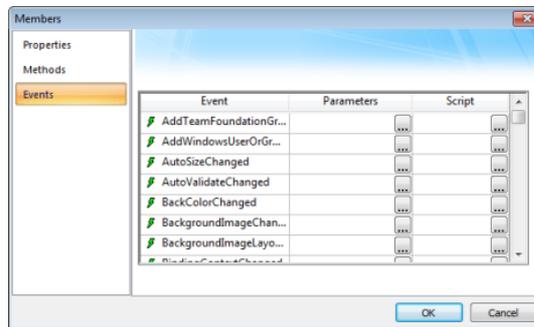
Tip: When you click the Browse button (🔍), it will display the list of parameters supported by the method, allowing you to associate one tag with each parameter.

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:



Members Dialog – Events tab

- **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.

ADDING A LINK TO AN EXTERNAL IMAGE FILE

To add a link to an external image file, so that you can change the image during project runtime, use a Linked Picture screen object.

This task assumes that you have a Screen worksheet open for editing.

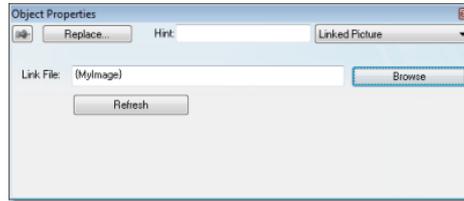
Also, you must decide where exactly the image file will be stored:

- If you want the image file to be downloaded with the rest of the project files to the target system, then it must be stored in the project folder (e.g., [...]\My Documents\InduSoft Web Studio v7.0 Projects \projectname\).
- If the image file will be stored elsewhere on the target system or the network, then note the complete file path.

Tip: If you want to embed the image data in the Screen worksheet file rather than link to an external image file, and if you do not need to change the image during project runtime, consider using a **Bitmap** screen object instead.

1. On the **Graphics** tab of the ribbon, in the **Libraries** group, click **Linked Picture**. An *Open* dialog is displayed.
2. Use the dialog to locate and select the image file, and then click **OK**. 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 is displayed.

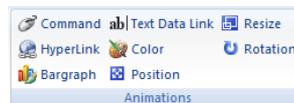


Object Properties: Linked Picture

4. In the **Link File** box, examine the link to the image file.
If the image file is stored in the project folder, then the link is a relative to that folder. If the image file is stored elsewhere, then the link is a complete file path.
5. If you want to be able to change the link during project runtime, type a reference to a String tag (e.g., `{tagname}`).
Then, during project runtime, the link will be refreshed whenever you change the value of the String tag. The tag value must have the same format as a normal link: a relative file path for a file stored in the project folder, or a complete file path for a file stored elsewhere on the target system or the network.
6. Close the *Object Properties* dialog.

Applying animations to screen objects

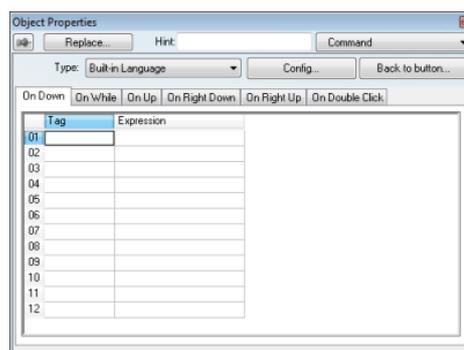
Use the *Animations* group to apply animations to a screen object or group of objects. Animations enable you to modify object properties on the fly (during runtime) according to tag values. Some animations also enable you to execute commands or insert values (set points) to the tags.



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.



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.

 **Note:**

- The runtime project handles touchscreen actions the same as mouse pointer actions.
- The events On Right Down, On Right Up and On Double Click are not supported in projects running on Windows Embedded target systems.
- When creating a project for a touchscreen device, keep in mind that events On Right Down and On Right Up cannot be triggered on such devices.

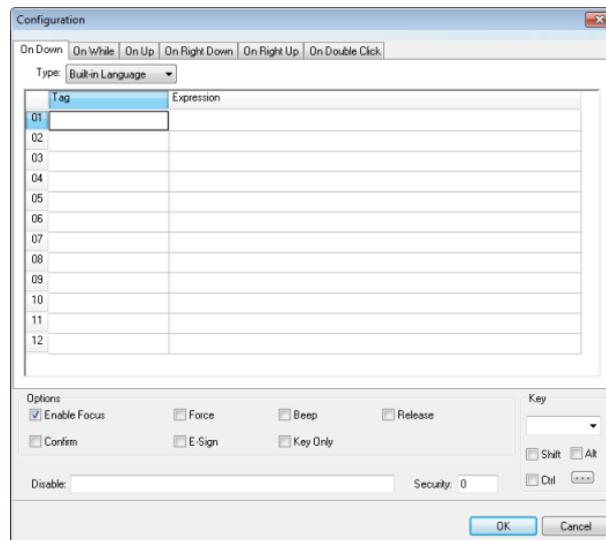
- **Type menu:** This setting defines the type of action that must be executed by the event of the Command animation. Notice that each event has its own type. Therefore, the same Command animation can be configured with different types of action for different events. The following types are supported:

Type	Description
Built-in Language	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 one when the event is triggered. The result of each expression is written to the tag configured in the Tag column (if any). Consult the Built-in Scripting Language chapter for more information.
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

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 checked, any tag that receives a value will generate events based on its change, even if the value of the tag in question does not change. For instance, if a tag has the value 0 and the Command animation overwrites the same value 0 to this tag, any other task in the runtime project will recognize that this tag changed value (even if it did not) after executing the animation. This option is useful when you want to make sure that actions driven by tag changes (e.g., Write on Tag Change from a communication driver) are triggered after the Command animation is executed.

 **Note:** For projects created with InduSoft Web Studio v6.1+SP3 or earlier, the **Force** option 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** pane: 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.

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

You can use this dialog to specify the following parameters:

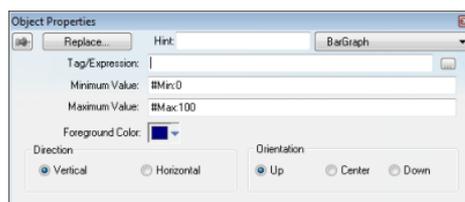
- **Hyperlink Type** combo-box: Click the combo-box button to select a URL protocol from the list. The project uses this protocol when it loads the URL.
- **URL** field: Type the URL address you want to load.

 **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

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

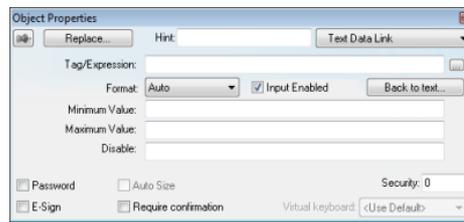
On the **Graphics** tab, in the **Animations** group, click **Text Data Link** to add the animation input or output text property to a selected **Text** object. Applying the **Text Data Link** property allows you to insert and display tag values in real time if you are using the keyboard or on-screen 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, **MyLabel1 #####** 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.

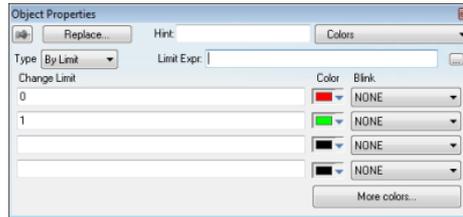
This option does not actually change the specified tag or expression in any way. For example, **Tag/Expression** is set to a tag of **Integer** type, **Input Enabled** is checked, and **Fmt** is set to **Hexa**. You may input a new value in hexadecimal format, but it is saved in your project database 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.



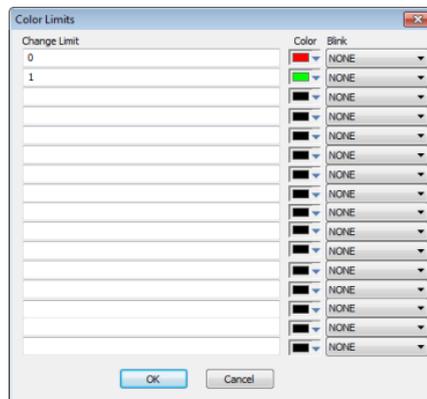
Object Properties: Colors

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 ascendant 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

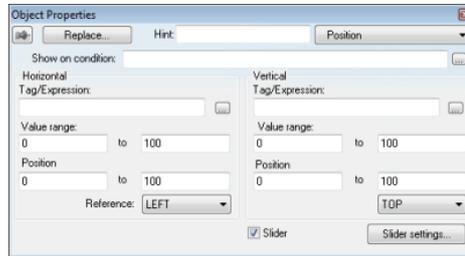
- **Color** combo box: Click the combo-box button to associate a color with each color change limit. When the Color dialog opens, click a color to select it, and then close the dialog.
- **Blink** combo-box: Click the combo-box button to specify whether the color change will blink, and how fast it will do so.

 **Note:** The following fields are automatically disabled (grayed out) when **Type** = By Color: **Change Limit**, **Color** and **Blink**.

POSITION ANIMATION

The Position animation allows you to move an object horizontally and/or vertically during runtime.

On the **Graphics** tab, in the **Animations** group, click **Position** to add the animation to an object. Double-click on the object to open its *Object Properties* dialog.



Object Properties: Position

Use the dialog to configure the following properties:

- **Show on condition** text box: Configure a tag/expression in this field to control when the object is shown. When the tag/expression is TRUE (i.e., any non-zero value), the object is shown. When the value is FALSE (i.e., 0 or a blank space), the object is not shown. You can also leave the field blank to always show the object.
- **Slider** check box: Click (check) this option to operate the object like a slider, which means that you can drag the object around the screen to send values to the configured tags. (See *Move* below.)

Click the **Slider** button to configure additional settings:

- **Disable** text box: Configure a tag/expression in this field to control when sliding is disabled. When the value of the tag/expression is TRUE (i.e., any non-zero value), sliding is disabled. When the value is FALSE (i.e., 0 or a blank space), sliding is enabled.
- **Security** text box: Enter the [security level](#) required to use the object as a slider.
- **Move** area: Configure these settings to determine how the object moves on the screen:
 - **Tag / Expression** text boxes: The meaning of these boxes depends on whether the **Slider** option is enabled (see above):
 - **Expression** text box: If the **Slider** option is not enabled, then configure either a tag *or* an expression in this field. The current value of the tag / expression determines the current position of the object. As the value changes, the object is moved on the screen.
 - **Tag** text box: If the **Slider** option is enabled, then enter only an Integer or Real tag in this field. The tag will receive a value that is equivalent to the current position of the object. As the user moves the object, the value is changed.

For the Horz (horizontal) position, the value increases as the object moves to the right and it decreases at the object moves to the left. For the Vert (vertical) position, the value increases as the object moves to the bottom and it decreases at the object moves to the top.

- **Range** text 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** text 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 runtime, 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 Horz and Vert.

- **Reference** drop-down lists: Select a reference point on the object. The following table shows how your selections for Horz and Vert work in combination:

	LEFT	CENTER	RIGHT
--	------	--------	-------

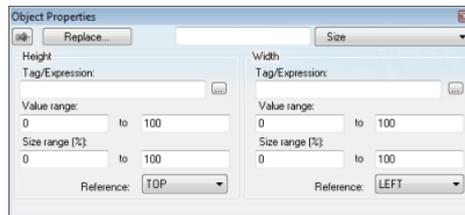
TOP			
CENTER			
BOTTOM			

This reference point is meaningful only if you have the [Size 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.

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: Size

Use the dialog to configure the following properties:

- **Tag** text boxes: Enter the tags that will control the **Height** and **Width** of the object. Leave either field blank if you don't want the object to change size in that dimension.
- **Range** text 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 (%)** text 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 Editor*, so 200% would be double the original size, and so on.

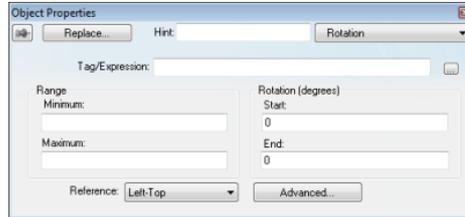
During runtime, the object's size is proportional to the tag value within its range. For example, if **Size (%)** is 0 to 100 and **Range** is 0 to 10, then each increment in the value will increase the object size by 10%. This is true for both Height and Width.

- **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:

	LEFT	CENTER	RIGHT
TOP			
CENTER			
BOTTOM			

ROTATION ANIMATION

On the **Graphics** tab, in the **Animations** group, click **Rotation** to add the animation to a **Line**, **Open Polygon**, **Closed Polygon**, or **Bitmap** object. Double-click on the object to open the *Object Properties* dialog.



Object Properties: Rotation animation

Use this dialog to specify the following parameters:

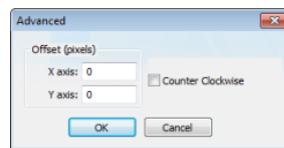
- **Tag/Expression** field: Enter a Tag name or expression to associate with the Rotation animation. The value of **Tag/Expression** determines the actual rotation of the object; as the value changes, so does the amount of rotation.
- **Range** area: Enter the **Minimum** and **Maximum** values allowed 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, where you can configure the following settings:



Object Properties: Rotation animation – Advanced Dialog

- **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.

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.

Move to Front and Move to Back

IWS assigns a unique identification number (ID#) to every object on the screen. These ID#s always start at zero and range up to the total number of objects on the screen. You can click on an object to display its ID# in the [status bar](#).

IWS uses ID#s to determine whether an object displays in front of, or behind, another object on the screen. Objects with lower ID#s display behind objects with higher ID#s.

Use the following object layer tools to move selected object(s) behind or in front of another screen object(s).

Note: You can use these tools only with a single selected object or [group](#) of objects. You cannot use these tools with multiple objects selected.

Also, if you select a group of objects and move them the behind or in front of another object, then the selected group of objects maintain their original display order.

Click the **Move to back** tool to move a selected object or objects behind all other objects on the screen. IWS assigns the object the lowest ID# and moves that object behind all other objects on the screen.



Moving Objects to Back

Note: Alternatively, right-click on an object and select **Move to back** from the object's shortcut menu.

Click the **Move to front** tool to move a selected object or objects in front of all other objects on the screen. IWS assigns the object the highest ID# and moves that object behind all other objects on the screen.



Moving Objects to Front

 **Note:** Alternatively, right-click on an object and select **Move to front** from the object's shortcut menu.

Move Backward and Move Forward

IWS assigns a unique identification number (ID#) to every object on the screen. These ID#s always start at zero and range up to the total number of objects on the screen. You can click on an object to display its ID# in the [status bar](#).

IWS uses ID#s to determine whether an object displays in front of or behind another object on the screen. Objects with lower ID#s display behind objects with higher ID#s. Use the following object layer tools to move selected object(s) behind or in front of another screen object(s).

 **Note:** You can use these tools only with a single selected object or [group](#) of objects. You cannot use these tools with multiple objects selected. Also, if you select a group of objects and move them the behind or in front of another object, the selected group of objects maintain their original display order.

Click the **Move backward** tool to move the selected object or group one layer below the next object on the screen. (Alternatively, right-click on the object and select **Move backward** from the shortcut menu.) IWS assigns the selected object the next available ID# less than the object behind which it was moved.

Click the **Move forward** tool to move the selected object or group one layer above the next object on the screen. (Alternatively, right-click on the object and select **Move forward** from the shortcut menu.) IWS assigns the selected object the first available ID# greater than the object in front of which it was moved.

Group and Ungroup Tools

Use the following tools to group and ungroup two or more selected objects.

 **Note:** All objects with [animations](#) and Group of Symbols objects (which includes most symbols and library objects) have multiple *Object Properties* dialogs and properties. You can use the drop-down list on the *Object Properties* dialog (**Properties** on the Graphics tab of the ribbon) to access these different dialogs and properties.

Click the **Group** tool to combine multiple objects into a single object to facilitate object selection and manipulation. (You can access each part of the group in the *Object Properties* dialog.)

 **Note:** Alternatively, you can right-click on an object and select **Group** from the object's shortcut menu.

Click the **Ungroup** tool to separate a grouped object into its individual components.

 **Note:** Alternatively, you can right-click on an object and select **Ungroup** from the object's shortcut menu.

 **Tip:** A complex grouped object can consist of several sets of grouped objects (known as *subgroups*). Consequently, you may find it necessary to ungroup all of the subgroups to completely ungroup a complex object.

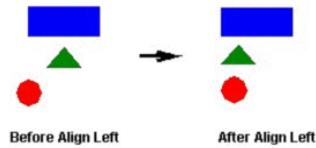
Align, Center and Distribute Tools

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.

 **Note:** In all of the figures provided, the rectangle represents the last object selected.

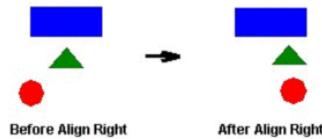
Use the following alignment tools to align a series of objects.

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:



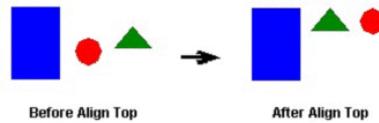
Aligning Objects Left

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:



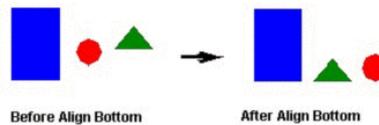
Aligning Objects Right

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:



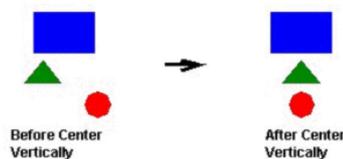
Aligning Object Tops

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:



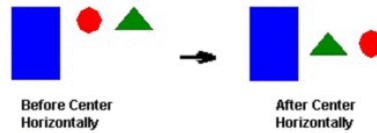
Aligning Object Bottoms

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

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:



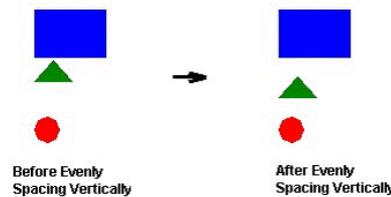
Centering Objects Horizontally

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

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:

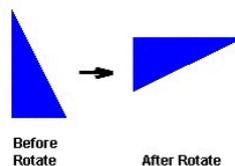


Distributing Objects Vertically

 **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.

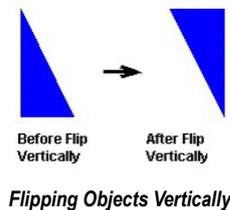


Rotating Objects

 **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

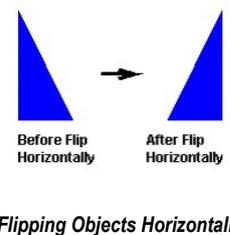
Click the **Flip Vertically** tool to invert the selected object vertically. The object rotates around an imaginary line through its vertical center until it is a mirror image of the original object. For an example, see the following figure:



 **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:



 **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.)

Alarms, Events, and Trends

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.

Alarm worksheet

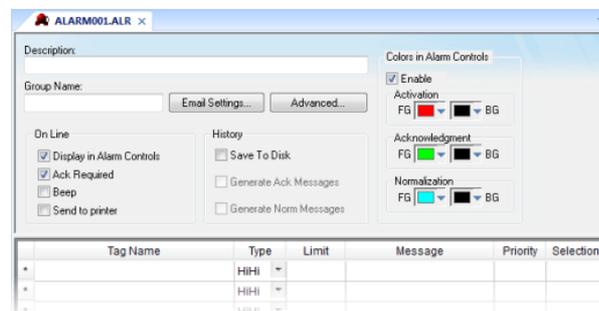
The Alarms folder enables you to configure alarm groups and tags related to each group. The Alarm worksheet defines the alarm messages generated by the project. The primary purpose of an alarm is to inform the operator of any problems or abnormal condition during the process so he can take corrective 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.



Alarm worksheet

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
```



Caution: 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](#):

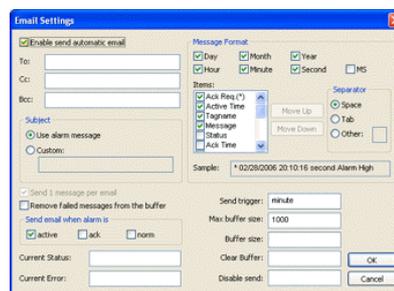
Field	Remarks	Syntax
Description	Description of the alarm group. It is displayed on the workspace. This field is used for documentation only.	Text (up to 80 chars)
Group Name	Name of the Alarm group. During runtime, the operator can filter alarms based on the Group Name by the built-in Filters dialog of the Alarm/Event control object.	Text (up to 32 chars)
Email Settings	Launches the Email Settings dialog , where you can configure the settings for emails sent automatically based on alarm conditions.	Button
Advanced	Launches the Advanced Settings dialog , where you can configure the settings for emails sent automatically based on alarm conditions.	Button
On Line > Display in Alarm Controls	When checked, the alarms are available to be displayed on the Alarm/Event Control object .	Checkbox
On Line > Ack Required	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.	Checkbox
On Line > Beep	When checked, the computer keeps beeping while there are alarm(s) to be acknowledged, currently active.	Checkbox
On Line > Send to Printer	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 <code>project_name.APP</code> file: <pre>[AlarmLog] Device=PrinterPath/PrinterName</pre>	Checkbox
History > Save to Disk	When checked, the alarm messages are stored in the history log when they become active.	Checkbox
History > Generate Ack Messages	When checked, the alarm messages are stored in the history log when they are acknowledged.	Checkbox
History > Generate Norm Messages	When checked, the alarm messages are stored in the history log when they become normalized.	Checkbox
Colors in Alarm Controls > Enable	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.	Color
Colors in Alarm Controls > FG and BG	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: <ul style="list-style-type: none"> Activation: Alarm active and not acknowledged 	Color

Field	Remarks	Syntax
	<ul style="list-style-type: none"> • Acknowledgement: Alarm active and acknowledged • Normalization: Alarm no longer active and not acknowledged. 	

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).

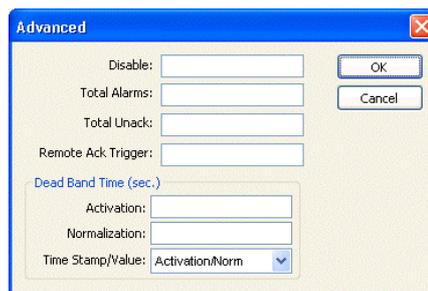


Alarms Worksheet — Email Settings

Field	Remarks	Syntax
Enable send automatic email	Name of the tag associated with the alarm.	Checkbox
To, Cc, Bcc	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 (;).	Text and/or {Tag} (up to 1024 chars)
Subject	When selecting "Use alarm message", the alarm message itself is used as the subject of the email to be sent. When selecting "Custom", you can configure a custom text to be used as Subject when sending the alarm.	Radio-button / Text (up to 1024 chars)
Send 1 message per email	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 "Use alarm message".	Checkbox
Remove failed messages from the buffer	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.	checkbox
Send email when alarm is	Allow you to configure which alarm events should generate emails: <ul style="list-style-type: none"> • Active: When the alarm becomes active. • Ack: When the alarm is acknowledged. • Norm: When the alarm is normalized. Notice that each event can be enabled/disabled individually.	checkbox
Current Status	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: <ul style="list-style-type: none"> • -2: Incorrect version of the INDMail.DLL library. • -1: The INDMail.DLL library is corrupted. • 0: SendEmailExt function is not being executed. • 1: Sending email(s) 	Tag

Field	Remarks	Syntax
	<ul style="list-style-type: none"> 2: Last email was sent successfully. 3: There was an error sending the last email. 	
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.	Tag
Message Format	<p>This interface allows you to configure the actual format of the message sent by email, based on the alarm event(s):</p> <ul style="list-style-type: none"> Day, Month, Year, Hour, Minute, Second, MS: The options checked will compose the timestamp for the alarm messages. MS stands for milliseconds. Items: 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. Separator: Allow you to choose the separator used between the items checked in this interface. <p>While you configure these settings, the Sample field displays an example of the format of the message according to the settings being configured.</p>	Checkbox and Radio-button
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.	Tag
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.	Tag or Number
Buffer size	The tag configured in this field, if any, is updated with the number of messages (events) currently stored in the internal buffer.	Tag
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.	Tag
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.	Tag

ADVANCED SETTINGS FOR ALARM WORKSHEET



Alarm Worksheet, Advanced Settings Dialog

The following table describes the Advanced settings on an [Alarm worksheet](#):

Field	Remarks	Syntax
Disable	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).	Tag
Total Alarms	The tag configured in this field, if any, is updated with the number of alarms from this group, which are currently active.	Tag

Field	Remarks	Syntax
Total Unack	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.	Tag
Remote Ack Trigger	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.	Tag
Dead Band Time > Activation	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.	Tag or Number
Dead Band Time > Normalization	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.	Tag or Number
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: <ul style="list-style-type: none"> • 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

Tag Name	Type	Limit	Message
	HiHi 		
	HiHi 		
	HiHi 		

The following table describes the Body settings on an [Alarm worksheet](#):

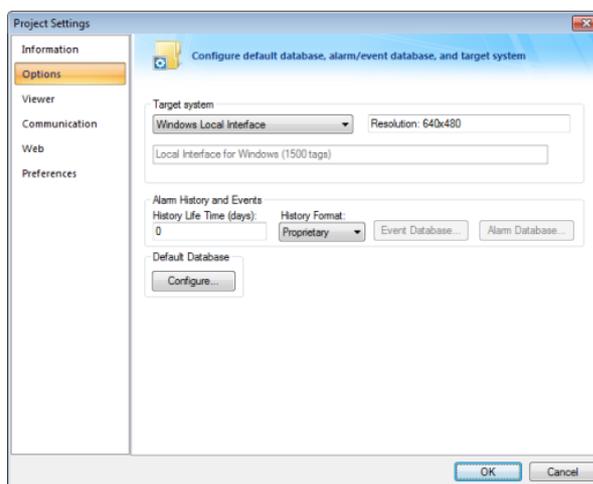
Field	Remarks	Syntax
Tag Name	Name of the tag associated with the alarm.	Tag
Type	Type of the alarm: <ul style="list-style-type: none"> • HiHi: Activates the alarm if the tag value is equal or higher than the limit. • Hi: Activates the alarm if the tag value is equal or higher than the limit. (For Boolean tags, if the value is 1.) • 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. 	Combo-box

Field	Remarks	Syntax
	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).	
Limit	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., TagLevel->HiLimit).	Number
Message	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.	Text and/or {Tag} (up to 256 chars)
Priority	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.	Number (from 0 to 255)
Selection	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.	Text (up to 7 characters)

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.

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Options**. The *Project Settings* dialog is displayed.



Project Settings: 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**.

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 `[...]\My Documents\InduSoft Web Studio v7.0 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:

```
P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11 | P12 | P13 | P14 | P15 | P16 | P17 | P18 | P19 | P20 | P21 | P22 | P23
P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11 | P12 | P13 | P14 | P15 | P16 | P17 | P18 | P19 | P20 | P21 | P22 | P23
.
P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11 | P12 | P13 | P14 | P15 | P16 | P17 | P18 | P19 | P20 | P21 | P22 | P23
```

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:

Proprietary	Database		Description	File Vers.
Field Number	Column Name	Data Type		
P1	—	—	File version (Current = 003)	001
P2	Al_Start_Time	TimeStamp	Start Date (MM/DD/YYYY)	001
P3			Start Time (HH:MM:SS)	001
P4	Al_Tag	String	Tag Name	001
P5	Al_Message	String	Alarm Message	001
P6	Al_Ack	Boolean	Ack, where: <ul style="list-style-type: none"> • 0: Alarm was acknowledged or does not require acknowledgment • 1: Alarm was not acknowledged 	001
P7	Al_Active	Boolean	Active, where: <ul style="list-style-type: none"> • 0: Alarm is not active • 1: Alarm is active 	001
P8	Al_Tag_Value	Real	Tag Value when the alarm occurred	001
P9	Al_Group	Integer	Alarm Group Number	001
P10	Al_Priority	Integer	Priority Number	001
P11	Al_Selection	String	Selection	001
P12	Al_Type	Integer	Type, where: <ul style="list-style-type: none"> • 1 is HiHi • 2 is Hi(On) • 4 is Lo(Off) • 8 is LoLo • 16 is Rate(Change) 	001

Proprietary	Database		Description	File Vers.
Field Number	Column Name	Data Type		
			<ul style="list-style-type: none"> • 32 is Deviation+ • 64 is Deviation- 	
P13	Al_Ack_Req	Boolean	Ack required, where: <ul style="list-style-type: none"> • 0: Alarm requires acknowledge • 1: Alarm does not require acknowledge 	001
P14	Al_Norm_Time	TimeStamp	Normalization Date (MM/DD/YYYY)	001
P15			Normalization Time (HH:MM:SS)	001
P16	Al_Ack_Time	TimeStamp	Ack Date (MM/DD/YYYY)	001
P17			Ack Time (HH:MM:SS)	001
P18	Al_User	String	User Name	002
P19	Al_User_Comment	String	Comment	002
P20	Al_User_Full	String	User Full Name	003
P21	Al_Station	String	Station	003
P22	Al_Prev_Tag_Value	Real	Previous Value	003
P23	Bias	Integer	Time Zone Bias	003
—	Al_Start_Time_ms	Integer	Number of milliseconds for the Start Time timestamp. This field is used when the database does not support ms in a TimeStamp field.	003
—	Al_Norm_Time_ms	Integer	Number of milliseconds for the Norm Time timestamp. This field is used when the database does not support ms in a TimeStamp field.	003
—	Al_Ack_Time_ms	Integer	Number of milliseconds for the Ack Time timestamp. This field is used when the database does not support ms in a TimeStamp field.	003
—	Al_Deleted	Boolean	Deleted, where: <ul style="list-style-type: none"> • 0: Alarm message was not deleted by the user (not visible). • 1: Alarm message was deleted by the user (visible). 	003
—	Last_Update	TimeStamp	Timestamp of the last update for this alarm.	003
—	Last_Update_ms	Integer	Number of milliseconds for the Last_Update timestamp. This field is used when the database does not support ms in a TimeStamp field.	003

 **Tip:** You can customize the names of the columns in the database table by manually editing the project file (e.g., [...] \My Documents \InduSoft Web Studio v7.0 Projects \projectname \projectname .APP) as follows:

```
[Alarm]
DefaultName=NewName
```

For example:

```
[Alarm]
Message=Alarm_Message
Ack=Acknowledgment
```

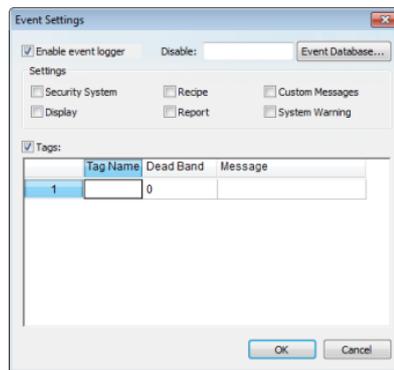
About events and event logging

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

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.

Option

Security System

Display

Recipe

Report

Description

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

Open Screen and Close Screen events.

Recipes loaded, saved, initialized, or deleted.

Reports saved to disk or sent to printer.

Option	Description
Custom Messages	Events generated by calling the SendEvent function.
System Warning	Various runtime warnings and errors, including: <ul style="list-style-type: none"> • Errors that occur when sending alarms by email • Tag was blocked/unblocked • Division by zero • Connection/Disconnection of the remote security system

5. To log changes in specific project tags, select **Tags**, and then in the table, specify the tags.

Column	Description
Tag Name	The name of the project tag that you want to log to the history file.
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 <code>{tagname}</code> 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](#).

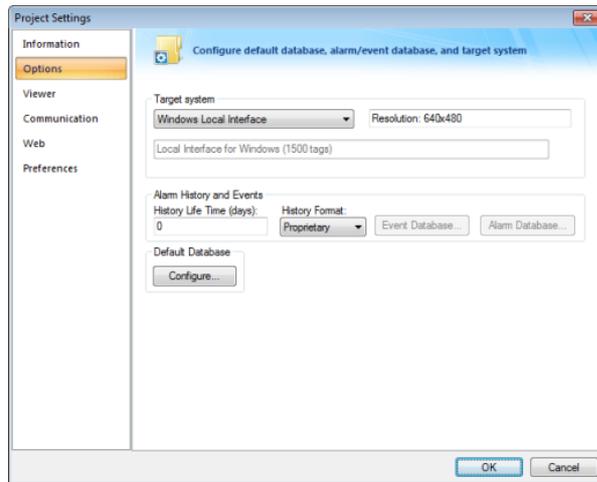
Alternatively, you can save the event log to an external SQL database. For more information, see [Saving your event log to an external database](#).

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.

1. On the **Project** tab of the ribbon, in the **Settings** group, click **Options**.

The *Project Settings* dialog is displayed.



Project Settings: 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**.

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
- *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 [...]*My Documents\InduSoft Web Studio v7.0 Projects\projectname\Alarm\EV030507.EVT*.

 **Tip:** To change where your project saves these files, use the [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:

```
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
:
```

P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11 | P12 | P13

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:

Proprietary	Database		Description	
Field Number	Column Name	Data Type		
P1	Version	Integer	This field is created only when the History Format is Proprietary. Current version: 002	
P2	Event_Type	Integer	1	Security System
			2	Display
			3	Recipe
			4	Report
			5	Custom Message
			6	System Warning
			7	Log Tags
P3	Event_Time	TimeStamp	Time stamp indicating when the event occurred. When the History Format is Proprietary, the project saves the event time in the following format: MM/DD/YYYY HH:MM:SS.MSS . When the History Format is Database, the project saves the event time in the default timestamp format of the database.	
P4	Event_Info	String	Tag name.	
P5	Value	Real	Tag value when the event occurred.	
P6	Source	String	Name of the task that generated the event.	
P7	User	String	User logged on when the event occurred.	
P8	User_Full	String	Full name of the user logged on when the event occurred.	
P9	Message	String	Event message.	
P10	Station	String	Name of the station (computer) where the event occurred.	
P11	Comment	String	Comment (optional) typed by the operator when the event occurred. This field only exists for Version >= 2.	
P12	Previous_Value	Real	Tag value that occurred before the event. This field only exists for Version >= 2.	
—	Deleted	Boolean	This field is created only when the History Format is Database. <ul style="list-style-type: none"> 0 (FALSE): Event message was not deleted. 1 (TRUE): Event message was deleted. 	
P13	Bias	Integer	Difference (in minutes) between Event_Time and the GMT time. This field only exists for Version >= 2.	
—	Last_Update	TimeStamp	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.	

 **Tip:** You can customize the names of the columns in the database table by manually editing the project file (e.g., [...]\My Documents\InduSoft Web Studio v7.0 Projects\projectname\projectname.APP) as follows:

```
[EventLogger]
DefaultName=NewName
```

For example:

```
[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 an 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.
3. Double-click on the object to open the following *Object Properties* dialog.



Object Properties: Alarm/Event Control

You can use this dialog to specify the following parameters:

- Select an alarm object mode in the *Type* pane:
 - **Alarm Online:** Display only current alarm messages.
 - **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.
- Click (enable) the **Show gridlines** checkbox to display gridlines in the object.



Displaying a Grid

- Click (enable) the **Show Header** checkbox to display a header on the object.



Displaying a Header

- Use the **Win** color box to select a background color for the object. Click the color box to open the color palette pop-up, then simply click a color to select it.
- Click (enable) the **Ext translation** checkbox to enable the external translation of messages using the Translation Tool. (See [The Translation Tool](#) for more information.)
- **E-Sign** checkbox: When this option is checked, the user will be prompted to enter the Electronic Signature before executing the animation.

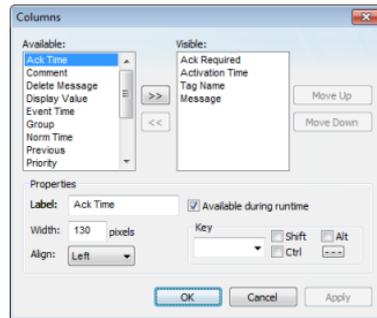
- **VK:** Virtual Keyboard type used for this object. You need to select the Virtual Keyboard option in the *Viewer* settings (**Viewer** on the Project tab of the ribbon) before configuring the Virtual Keyboard for this interface.

Fonts

Click the **Fonts** button to open a [standard Fonts interface](#) where you can specify display properties for the message text.

Columns

Click the **Columns** button to open the *Columns* dialog where you can specify display properties for columns in the object.



Columns dialog

- 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.

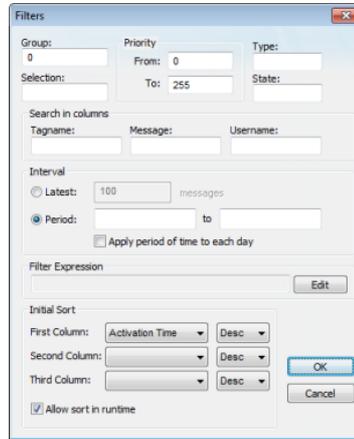
- Use the **Label** and **Width** fields in the *Properties* pane to change the default column labels and widths at runtime.
- Use the **Align** combo box to specify alignment (**Left**, **Center**, or **Right**) for the alarm message text within a specified column.
- Click (enable) the **Available during 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.

Note: When acknowledging an alarm, the Alarm Control object sends a message to the **Alarm task** with the following information: **Tag Name**, **Type**, **User** and **Station**. This is a solution to control acknowledged alarms from a Thin Client.

Filters

To filter alarm messages during runtime, click the **Filters** button. The *Filters* dialog displays so you can specify filtering parameters for the Alarm Control object.



Filters dialog

- Use the **Group** field to filter messages by the [Alarm 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 or dashes to specify a range of groups; for example, 1, 3, 5-6.
- Use the **Selection** field to filter messages by the Selection text configured on the Alarm Worksheet.
- In the *Priority* pane, 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:

Value	Description
0	All alarms (default)
1	All active and unacknowledged alarms
2	All active and acknowledged alarms
3	All inactive and acknowledged alarms
4	All inactive and unacknowledged alarms

Leaving this field blank is effectively the same as entering a value of 0.

- In the *Search in columns* pane, 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 parameters in the *Interval* pane 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* pane. These fields cannot take values directly.
- You can specify Integer tags in the **From** and **To** fields *Priority* pane, the **State** field, and the **Latest** field from the *Interval* pane.

- You can use wildcards (* and ?) when specifying values for the **Selection**, **Tagname**, **Message**, and **Username** fields.

- Use the *Filter Expression* pane to configure an expression that will filter unwanted messages out of the display. Only messages that satisfy the expression will be shown.

To enter an expression, click on the **Edit** button; the *Alarm Filter Expression* dialog is displayed. The filter expression must follow the basic syntax of..

```
[Column Name]Comparison Operator'Value'
```

...where the **Column Name** is the name of a column in the Alarm/Event Control object. 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.

 **Note:**

- The maximum number of characters is 1024 for Engineering Mode and 2048 for Runtime Mode.
- The **Display Value** and **State** columns are not supported by the filter expression.

 **Tip:**

- You can combine several conditions simultaneously by using the logic operators AND, OR, and NOT. For example:

```
[Type]='HiHi' OR [Type]='LoLo' AND [Activation Time]>'08/17/2007 15:00'
```

- You can use wildcards (* and ?) in the filter expression.
- It is not necessary to use the square brackets when the **Column Name** is only one word (e.g., **Value**).
- You can change the filter expression during runtime by specifying **String** tags in curly brackets. For example:

```
[Value]='{AlarmFilterValue}'
```

- To use more than 1024 characters in the filter expression during runtime, you must use more than one tag between curly brackets using the **{TagName1} AND {TagName2}** syntax.

- Use the parameters in the *Initial Sort* pane 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 (*project_name.app*) to change the following setting:

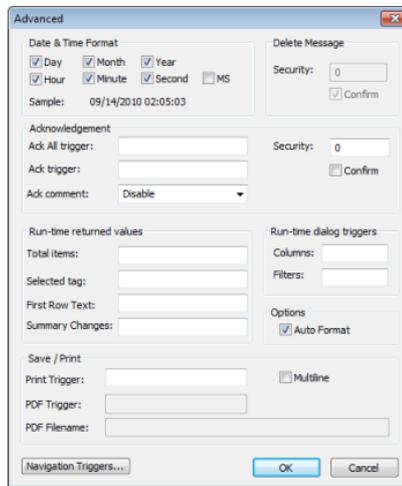
```
[Objects]
DescendingAlarmListTime=TRUE or FALSE
```

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.

Advanced

Click the **Advanced** button to open the *Advanced* dialog where you can specify advanced properties for the Alarm Control object.



Advanced dialog

- Use the parameters in the *Date & Time Format* pane 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 parameters in the *Ack* pane to control how alarms are acknowledged.
 - Security** field: Type a numeric value to specify which security level can acknowledge an alarm message. Only those users with the specified level can respond.
 - Ack All trigger** field: Type a tag to receive a value. When the tag changes value, it indicates that all messages in the alarm object have been acknowledged.
 - Ack trigger** field: Type a tag to receive a value. When the tag changes value, it indicates that the message at the top of the alarm object has been acknowledged.
 - Confirm** checkbox: Click (enable) this box to display a confirmation dialog when the user tries to acknowledge a single alarm.
 - Enable comment (individual ack only)** checkbox: Click (enable) this box to allow the user to enter comments about the alarm, just after acknowledging it.
- Use the parameters in the *runtime dialog triggers* pane to control
 - Columns** field: Type a tag to receive a value. When the tag changes value, it opens a dialog allowing the user to customize the columns visible in the object.
 - Filters** field: Type a tag to receive a value. When the tag changes value, it opens a dialog allowing the user to filter the columns visible in the object.
- Use the parameters in the *Delete Message* pane 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:** Click (enable) this box to require the user to confirm a message deletion before IWS actually deletes the selected alarm message.
- 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** field: 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** 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.
- **Total items** field: Type an integer tag to see how many alarms remain after IWS filters the alarm object using parameters specified on the [Filters dialog](#).
- **Auto Format** checkbox: When checked, decimal values in the Display Value, Previous and Value columns will be formatted according to the virtual table created by the [SetDecimalPoints\(\)](#) function.
- **Selected tag** field: 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 alarm event.
- **First Row Text** field: Type a string tag. This tag will receive the text of all fields from the first row of the Alarm/Event Control. The fields are tab delimited. Whenever the first row changes — either due to a new Alarm/Event, or simply because the rows are reordered — the value of the configured tag is updated.
- **Summary Changes** field: Type an integer tag. This 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.
- Click the **Navigation Triggers** button to open the following dialog:



Navigation Triggers dialog

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 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.

Trend worksheet

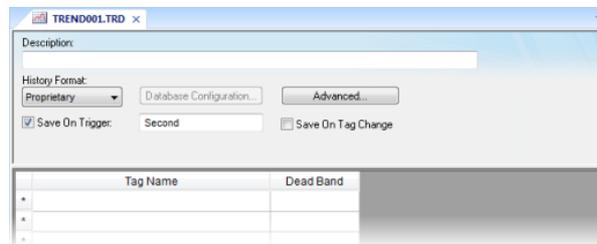
The Trend folder enables you to configure history groups that store trend curves. You can use the Trend worksheet to declare which tags must have their values stored on disk, and to create history files for trend graphs. The project stores the samples in a binary history file (*.hst), and shows both 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 whole group
- Body area (bottom section), where you define each tag in the group. This section contains several columns (only two are shown in the preceding figure).

Use the Header parameters on this worksheet as follows:

- **Description** field: Type a description of the worksheet for documentation purposes.
- **Type** combo box: Click the arrow button to select a trend history format from the list. The available options are:
 - **Proprietary**
 - **File Format:** Binary
 - **Default Path:** ...*project_name*\Hst\GGYYDDMM.HST, where:
 - **YY** = Two last digits of the year
 - **MM** = Month
 - **DD** = Day

 **Note:** IWS provides the HST2TXT.EXE and TXT2HST.EXE programs, which enable you to convert trend history files from binary (**.hst**) to text (**.txt**) and vice versa. For more information about these programs, see [Converting Trend History Files from Binary to Text](#) and [Converting Trend History Files from Text to Binary](#).

- **Database**
 - **Database Type:** Chosen by the user
 - **Default Table Name:**TRENDGGG (GGG = Trend Worksheet Number; e.g., TREND001 for the Trend Worksheet 001)

For more information about the structure of the database table that IWS uses to save history files, see [Database Interface](#).

-  **Caution:** You can specify String tags in many fields of the Trend worksheet, to change those values during runtime, but doing so may affect how those values are saved in the trend history:
- When the history format is **Proprietary**, the value of the String tag is converted to a numerical value (if possible) and then saved to the history file. If numeric conversion is not possible, then a value of 0 is saved.
 - When the history format is **Database**, the actual value of the String tag is saved in the database.

- **Database Configuration:** Opens the [Database Configuration](#) dialog, where you can enter the requisite settings to link the project to an external SQL Relational Database for the purpose of saving the trend history.
- **Save On Trigger** checkbox and field: 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** checkbox: 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](#).

Use the Body parameters on this worksheet as follows:

- **Tag Name** field: Type the tag name to be saved in the history file.
- **Dead Band** field: 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** field: Name of the field in the database where the tag will be stored. If this field is left blank, the name of the tag will be used as the tag name. Array tags and classes will have the characters "[", "]" and "." replaced by "_". Examples:

Tag Name	Default Field
MyArray[1]	MyArray_1
MyClass.Member1	MyClass_Member1
MyClass[3].Member2	MyClass_3_Member2

-  **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, then the Trend task will generate an error when you run the finished project.

See also:

- [Converting Trend History Files from Binary to Text](#)
- [Converting Trend History Files From Text to Binary](#)
- [Creating Batch History](#)
- [Configuring a Default Database for All Task History](#)

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 Bin
```
2. At the command prompt, copy the **Hst2txt.exe** into the same directory where the **.hst** file is located.

- 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.
- 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:

- From a DOS window, change directory (cd) to the IWS \Bin directory:

```
cd Bin
```

- At the command prompt, copy the **Txt2hst.exe** into the same directory where the **.txt** file is located.
- 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.

- 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:

Field Name	Data Type	Description
Group_Number	Integer	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.
Batch_Name	String	Name of the batch
Start_Time	TimeStamp	Date and Time that the batch was started.
End_Time	TimeStamp	Date and Time that the batch was finished
Pri_Table	String	Reserved
Sec_Table	String	Reserved
Description	String	Batch description
Deleted	Boolean	0: Batch has not been deleted 1: Batch has been deleted

 **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:

```
[Trend]
DefaultName=NewName

[TrendGroupPRI/SEC]
BatchHistory=TableName
```

For example:

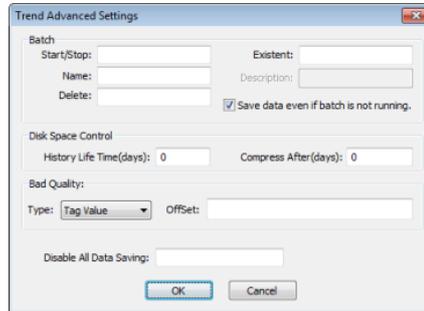
```
[TREND001PRI]
BatchHistory=MyTableForPrimaryDB

[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

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).
- **Existent (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:

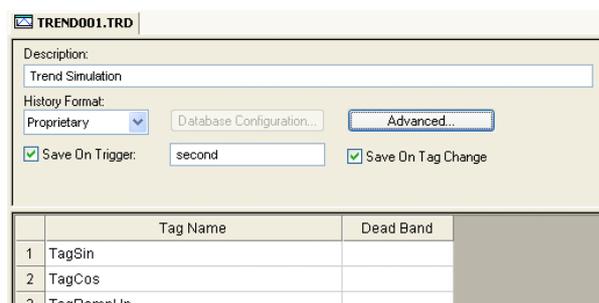
Type	Description
Tag Value	The actual value of the project tag when the tag quality was BAD, plus the specified Offset (if any).
Min Value	The minimum historical value of the project tag, minus the specified Offset (if any).
Max Value	The maximum historical value of the project tag, plus the specified Offset (if any).
Value	The specified Value only.
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.

Setting the Trend Database

1. Click the *Tasks* tab, and right-click the *Trend* folder. Double-click the applicable *Trend*.



2. Click the arrow to the right of the **History Format** field. Select **Database**.
3. Click the **Database Configuration** button. This opens the *Database Configuration dialog*. Enter the applicable data in this window. Click **OK** when you are finished.
4. Click the **File** drop-down menu, then click **Save** to save the changes to the *Trend*.

Trend Table on a Relational Database

The fields saved in the History Trend are described in the following table:

Field Name	Data Type	Remarks
Time_Stamp	TimeStamp	TimeStamp (Date and Time) when the data was saved.
<TagName>	Integer or Real (depending on the tag type)	IWS will create one field (column) in the database for each tag configured in the Trend worksheet.

Trend Control object

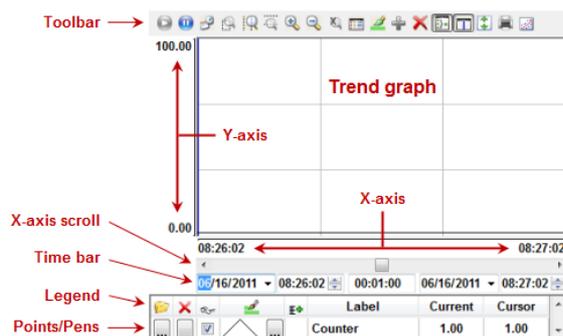
The Trend Control object displays data points (values) from different data sources in a graphic format.

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

About the trend control runtime interface

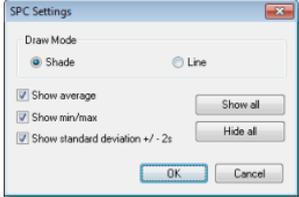
During project runtime, a trend control has its own built-in interface that the operator can use to change how trends are displayed. This section describes the major parts of the interface and how they are used.



Trend control runtime interface

Toolbar

Command/Tool	Icon	Description	Activation Tag
Run		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.	0 = Run Mode on 1 = Run Mode off
Stop		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.	0 = Stop Mode on 1 = Stop Mode off
Period		Opens a dialog which can be used to modify the X-axis scale main settings.	When the Activation Tag changes value (e.g., toggles), this command is executed.
Window Zoom		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.	
Horizontal Zoom		Allows the user to click on two points in the trend graph, defining the horizontal scale that must be available.	

Command/Tool	Icon	Description	Activation Tag
Vertical Zoom		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		Allows the user to zoom in (display half of the current X and Y scales) each time they click on the trend graph.	0 = Zoom In on 1 = Zoom In off
Zoom Out		Allows the user to zoom out each time they click on the trend graph.	0 = Zoom Out on 1 = Zoom Out off
Cancel Zoom		Cancels the current Window , Horizontal or Vertical Zoom and returns the trend graph to its original scale.	When the Activation Tag changes value (e.g., toggles), this command is executed.
Legend Properties		Opens a dialog which can be used to modify the Legend main settings.	
Pen Style		Opens a dialog which can be used to modify the pen style of the selected trend.	
Add Pen		Opens a dialog which can be used to add a new trend to the trend control.	
Remove Pen		Removes the selected trend from the trend control.	
Multiple Sections		Switches the Y scale between Multiple Sections (a section for each trend) and Single Section (all trend share the same Y scale section).	
Cursor		Switches the cursor (ruler) between visible and hidden.	0 = Cursor on 1 = Cursor off
Auto Scale		Changes the Y axis scale to fit all values from the trends that are currently being monitored.	When the Activation Tag changes value (e.g., toggles), this command is executed.
Print		Prints the current state of the trend control. (Historical data are not printed.)	When the Activation Tag changes value (e.g., toggles), this command is executed.
SPC		<p>Opens a dialog which can be used to show the statistical process control (SPC) information for the selected trend:</p>  <ul style="list-style-type: none"> • Draw Mode... <ul style="list-style-type: none"> • Shade: Draws the average value as a dashed line, and draws the min/max values and standard deviation as shaded areas. • Line: Draws the average value and standard deviation as dashed lines, and draws the min/max values as solid lines. • Show average: Show the calculated average of all of the trend's historical values. <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <p> Note: 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.</p> </div> <ul style="list-style-type: none"> • Show min/max: Show the minimum and maximum historical values of the trend. • Show standard deviation: Show the standard deviation of the trend. A low standard deviation indicates that the actual value tends to stay 	<p>When the Activation Tag changes value (e.g., toggles), this command is executed.</p> <p>Also, the tag's Bit properties (B0–B4) can be used to pre-select options in the dialog:</p> <ul style="list-style-type: none"> • tagname->B0 <ul style="list-style-type: none"> • 0 = close dialog • 1 = open dialog • tagname->B1 <ul style="list-style-type: none"> • 0 = Draw Mode: Line selected • 1 = Draw Mode: Shade selected • tagname->B2 <ul style="list-style-type: none"> • 0 = Show average cleared • 1 = Show average selected • tagname->B3 <ul style="list-style-type: none"> • 0 = Show standard deviation cleared • 1 = Show standard deviation selected • tagname->B4 <ul style="list-style-type: none"> • 0 = Show min/max cleared • 1 = Show min/max selected

Command/Tool	Icon	Description	Activation Tag
		close to the average; a high standard deviation indicates that the actual value tends to vary greatly from the average.	

 **Note:** Activation tags are configured in the trend control's object properties. For more information, see [Toolbar](#).

Legend

Command	Icon	Description
Selection		Launches a dialog, where the user can replace the data point associated with the selected trend on the legend
Remove		Removes the selected trend from the trend control
Hide		When checked, the selected trend is visible; otherwise, it is hidden.
Pen Style		Launches an embedded dialog, where the user can modify the pen style of the selected trend.
Scale		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.

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:

Area / Element Name		Description
Border	Type	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.)
	Color	Sets the color of the border, if the border type is Solid . For more information, see Selecting colors and fill effects .
Background	No Fill / Fill	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.
	Color	Sets the color and fill effect of the background fill, if it is enabled. For more information, see Selecting colors and fill effects .

Area / Element Name	Description
Points	Opens the, which allows configuration of the trend control's data points (or pens). For more information, see Trend Control: Points dialog .
Axes	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 .
Toolbar	Allows configuration of the user toolbar that is displayed above the trend control. For more information, see Trend Control: Toolbar dialog .
Data Sources	Allows configuration of multiple data sources for the trend. For more information, see Trend Control: Data Sources dialog .
Legend	Allows configuration of the legend that is displayed below the trend control. For more information, see Trend Control: Legend dialog .
Advanced	Allows configuration of the trend control's advanced properties, such as runtime options and tag triggers. For more information, see Trend Control: Advanced dialog .

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

The following table summarizes the properties of each data point:

Column Name	Description
Point	A unique ID number for the point, which is assigned automatically when the point is created in this interface.
Label	The label associated with the Point can be displayed on the Legend, during runtime, providing a short reference to the user for each Point.
Color	The color of the pen used to draw the values of the Point on the Trend Control object.
Data Source	The data source for this point. Tag is available by default, but all other sources must be configured in the <i>Data Sources</i> dialog.
Tag/Field	The meaning of this parameter depends on the Data Source type associated with the data point: <ul style="list-style-type: none"> 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 .

Column Name	Description
	 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

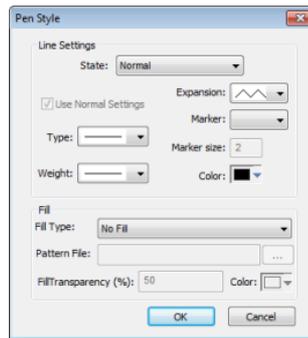
This dialog allows you to configure the style of the pen used to draw the data Point values on the object during runtime.

Accessing the dialog

To access the *Pen Style* dialog for a specific data point, first access the *Points* dialog and then click the **Style** column for that data point.

Also, it can be launched during runtime, allowing the user to customize these settings on-the-fly.

The dialog in detail



Pen Style dialog

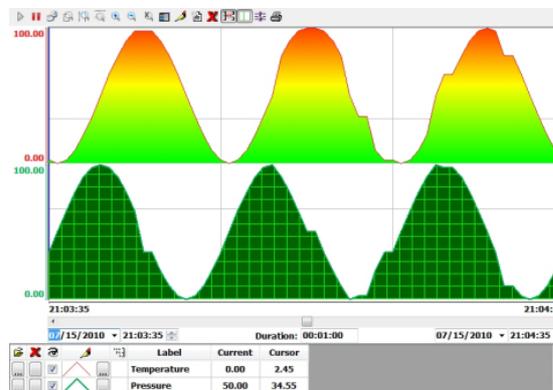
The *Pen Style* dialog includes the following elements:

Elements in the *Pen Style* dialog

Area / Element Name	Description	
Line Settings	State	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).
	Use Normal Settings	Available only for the Out of Limits 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.
	Type	The type of line (e.g., solid, dashed, dotted) that connects the data points.
	Weight	The weight of the line that connects the data points.
	Expansion	The algorithm used to connect the points, as follows: <ul style="list-style-type: none">  : 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.

Area / Element Name		Description
	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 when 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.
	Color	The color used to fill the trend area. Available only when Fill Type is set to Solid Color .
	Fill Transparency (%)	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 Custom Pattern and Solid Color .

Note: When viewing your project on either a Windows Embedded device or a Thin Client (any OS), the *Pen Style* dialog — available during runtime — allows the user to change the pen color only.



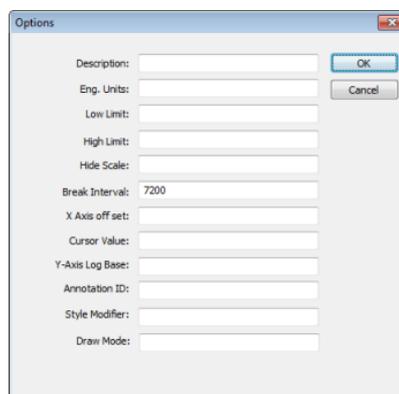
Trends with custom fill patterns

Options dialog

Accessing the dialog

To access the *Pen Style* dialog for a specific data point, first access the *Points* dialog and then click the *Options* column for that data point.

The dialog in detail



Options dialog

The *Options* dialog includes the following elements:

Element Name	Description
Description	This text can be displayed in the legend, providing a short description about the data point, during runtime. When tags are used, the default description is the one configured for the tag.
Eng. Unit	This text can be displayed in the legend, providing the engineering unit (i.e., the unit of measurement) associated with the data point, during runtime. When tags are used, the default units are the ones configured for the tag.
Lo Limit	When the data point value is below this limit, its pen can be displayed with a different style (e.g., color) during runtime. See Pen Style dialog for further information. When tags are used, the default Low Limit is the Low Alarm value configured for the tag.
Hi Limit	When the data point value is above this limit, its pen can be displayed with a different style (e.g., color) during runtime. See Pen Style dialog for further information. When tags are used, the default High Limit is the High Alarm value configured for the tag.
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	Maximum interval between two consecutive points. If the time between two consecutive samples is higher than this number (in seconds), the Trend Control assumes that there was no data collection in this period and does not draw a line linking both samples. When the X Axis is configured as numeric, the value on this field represents a numeric scalar value. If the X Axis is configured as date/time, the value for this field is given in seconds. This field has some special values: <ul style="list-style-type: none"> • -1 : Do not connect the points. • -2 : Connect only points in ascending order.
X-Axis Offset	Off-set for this data point from the X-Axis scale configured for the object. This option is useful when you want to display data from two or more data points using a different X scale (period of time/value) for each one, so you can compare them. When the X Axis is configured as numeric, the value on this field represents a numeric scalar value. If the X Axis is configured as date/time, the value for this field is given in seconds.
Cursor Value	Type the name of a project tag. During runtime, the trend control updates the value of this tag with the value of the intersection between the data point pen and the vertical cursor (if any).
Y-Axis Log Base	Type a tag name or numerical value. When the value is 1, the Y-axis of the trend is displayed on a logarithmic (e.g., 1, 10, 100, 1000) rather than linear scale.
Annotation ID	N/A
Style Modifier	N/A
Draw Mode	Type a tag name or numerical value. When the value is 1, the historical data for this trend are decimated before the trend is drawn. That means the trend's X-axis is divided into a number of intervals (determined by Max Points in the Advanced settings) and all of the data points within each interval are averaged together to be drawn as a single point. This is similar to the Decimation feature in the Advanced settings, except that the decimation is done only for this trend rather than for all trends in the Trend Control object.

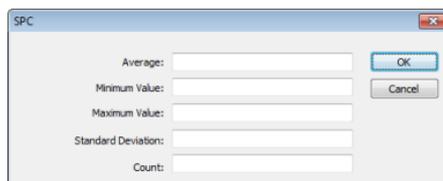
SPC dialog

In the *SPC* dialog, you can specify project tags that will receive certain statistical values that are calculated from the data point's entire history. 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

To access the *SPC* dialog for a specific data point, first [access the Points dialog for the Trend Control screen object](#) and then click the **SPC** column for that data point.

The dialog in detail



SPC dialog

The *SPC* dialog includes the following elements:

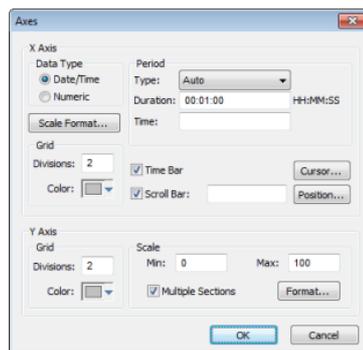
Element Name	Description
Average	Project tag (Real type) that will receive the calculated average of all of the data point's historical values. <div style="border: 1px solid black; padding: 5px;"> <p> Note: When a value is not a number (NaN) — for example, when a tag is flagged as BAD quality — it is counted as 0 for the purpose of calculating the average.</p> </div>
Minimum Value	Project tag (Real type) that will receive the minimum historical value of the data point.
Maximum Value	Project tag (Real type) that will receive the maximum historical value of the data point.
Standard Deviation	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	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



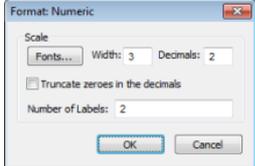
Axes dialog

The *Axes* dialog contains the following elements:

Area / Element Name		Description
X-axis	Data Type	Date/Time
		Numeric
		Scale Format
	Period (when Data Type is Date/Time)	Type <ul style="list-style-type: none"> Auto: When this option is selected, the Trend Control object works with Start Date/Time when is it triggered to Pause Mode, and it works with Time Before Now when it is triggered to Play Mode. Start Date/Time: 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. Time Before Now: 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.
	Duration	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

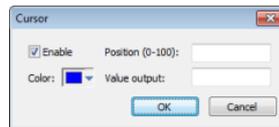
Area / Element Name		Description
		by this property is HH:MM:SS. For example, 36 : 00 : 00 (thirty six hours).
	Time	<p>This field is optional. The value of the tag configured in this field represents a period of time, rather than a specific date or time. The meaning of this value depends on the option set for the Type property.</p> <ul style="list-style-type: none"> 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. 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: <ol style="list-style-type: none"> Time (string value). For example, 48 : 00 : 00 (forty eight hours). Number of hours (real value). For example, 2 . 5 (two hours and thirty minutes). <p>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.</p>
	Period (when Data Type is Numeric)	<p>Min / Max</p> <p>Minimum and maximum values displayed on the X-axis.</p> <p>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.</p>
		<p>Eng. Units</p> <p>Engineering Unit (e.g., Kg, BTU, psi) that is associated with the X-axis during runtime.</p>
	Grid	<p>Divisions</p> <p>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.</p>
		<p>Color</p>
		<p>Time Bar</p> <p>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.</p>
		<p>Scroll Bar</p> <p>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. Optionally, 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.</p>
		<p>Cursor</p> <p>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:</p>
		<p>Positon</p> <p>Defines the position of the X-axis, as well as its direction and orientation, as follows:</p>
Y-axis	Grid	<p>Divisions</p> <p>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.</p>
		<p>Color</p>
	Scale	<p>Min / Max</p> <p>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).</p>
		<p>Multiple Selections</p> <p>When checked, the Y scale is divided automatically into one section for each pen; otherwise, all pens share the same Y scale.</p>
		<p>Format</p> <p>Launches a dialog for configuring the format of the labels displayed by the Y-axis.</p>
<p> 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.</p>		

- **Data Type:** The X-axis can display either Date/Time values or numeric values, according to this setting.

Data Type	Scale Format
Date/Time	
Numeric	

 **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.

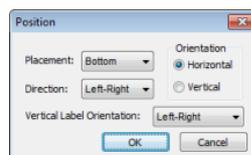
- **Cursor:** The cursor is an optional ruler orthogonal to the X-axis, which can be used during runtime to obtain the value of any pen at a specific point (intersection of the pen with the cursor). When you click this button, the Cursor dialog launches, where you can configure the settings for the optional vertical cursor as follows:



Cursor Dialog

Property	Description
Enable	When checked, the vertical cursor is visible during runtime.
Color	Color of the line drawn for the cursor.
Position (0:100)	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.
Value Output	You can configure a string tag in this field that returns the value of the X-axis in which the cursor is currently positioned.

- **Position:** Defines the position of the X-axis, as well as its direction and orientation, as follows:



Position Dialog

Property	Description
Placement	Side of the trend control on which the X-axis will be placed.
Direction	Direction of the X-axis.
Orientation	Orientation of the X-axis.
Vertical Label Orientation	The orientation of the text labels on the vertical axis, regardless of whether the vertical axis is X or Y.

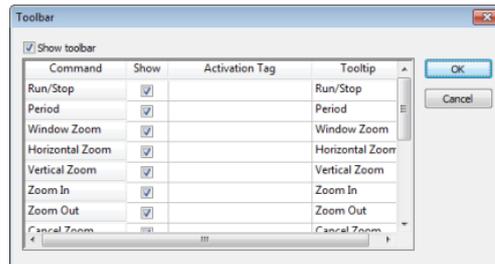
TOOLBAR DIALOG

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

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:

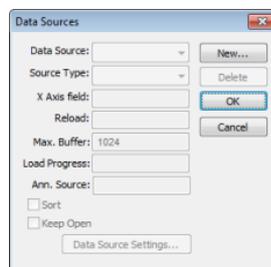
Column Name	Description
Command	The name of the command/tool. For more information about each tool, see
Show	The option to show the tool on the toolbar.
Activation Tag	An optional tag trigger — when the value of the tag changes from FALSE (0) to TRUE (any non-zero value), the command is activated as if the operator clicked the tool. This can be used to script changes in the trend display during runtime.
Tooltip	The tooltip that is displayed when the mouse cursor hovers over the tool.

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 dialog in detail



Data Sources dialog

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:** Enter the name of the field (column) from the data source that holds the X axis data.
- **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**:

Data Source Type	Description	X-Axis field	Data Source Settings
Batch	Batch generated by the Trend task of IWS	Disabled. The X-Axis data will be retrieved automatically on the correct position from the proprietary Batch file generated by IWS.	 <p>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.</p>
Database	SQL Relational Database	Field name that contains the X axis data	 <p>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.</p>
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.	 <p>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.</p>

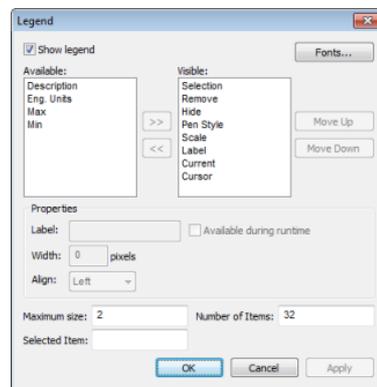
Data Source Type	Description	X-Axis field	Data Source Settings
			You can also choose one or more delimiters for the data stored in the text file. The value of each row is written in the text file between two delimiters. When using a comma as a delimiter, the grid object is able to read data from CSV 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.

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



Legend dialog

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:

Item	Legend Icon	Description
Eng Units		The tag/pen's Engineering Units.
Min		The tag/pen's minimum possible value.
Max		The tag/pen's maximum possible value.
Selection		Press button to select another tag for this pen.
Remove		Press button to completely remove this pen from the legend and the Trend chart.
Hide		Select (check) option to hide this pen in the Trend chart.
Pen Style		Press button to change the pen's line style, weight, color, markers, and so on.
Scale		Select (check) option to show the pen's scale on the Trend chart.
Description		Description of the tag/pen.
Current		The current value of the tag configured to the pen.

Item	Legend Icon	Description
Cursor		The value of the pen where it intersects the cursor line.

- **Properties:** Allows you to configure the properties for the field highlighted in the **Available** or **Visible** box:

Property	Description
Label	Label for the field displayed during runtime
Width	Width for the field (in pixels) during runtime.
Align	Alignment of the data displayed in the field.
Available during runtime	When this option is checked, the user can show or hide the field during runtime.

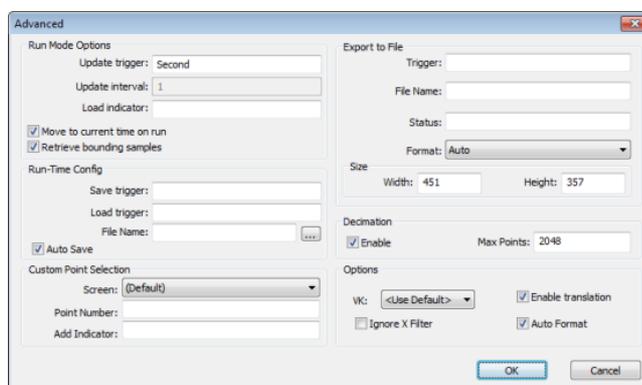
- **Maximum size:** Defines the size of the legend in terms of number of rows. For instance, the user might have 8 points being displayed in the trend object, if the maximum size is set to two, the legend will have a scroll bar to allow the user to scroll to the other points.
- **Number of items:** Number 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.

ADVANCED DIALOG

Accessing the dialog

To access the *Advanced* dialog for a specific Trend Control screen object, first [access the Object Properties dialog for that screen object](#) and then click **Advanced**.

The dialog in detail



Advanced dialog

The *Advanced* dialog contains the following elements:

Area / Element Name	Description	
Run Mode Options	Update trigger	When the tag configured in this field changes value, the trend object is updated (refreshed).
	Update interval	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.
	Load indicator	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.
	Move to current time on run	When this box is checked, X axis shifts to the current time automatically when the object is triggered to Play mode, during runtime.
	Retrieve bounding samples	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

Area / Element Name		Description
		the history. On the other hand, the object will not draw lines linking the first and last samples to the extremities of the object.
Run-Time Config	Save trigger	<p>The settings of the Trend object modified during runtime can be saved in temporary files. This option can be used to:</p> <ul style="list-style-type: none"> 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. 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. <p>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.</p>
	Load trigger	<p>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.</p> <div style="border: 1px solid black; padding: 5px;"> <p> 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.</p> </div>
	File Name	<p>If this field is left blank, the temporary file is saved in your project's Web sub-folder with the syntax <i>ScreenNameObjectIDTrendControl.stmp</i> (e.g., <i>MyScreen10TrendControl.stmp</i>). The Thin Client station saves/loads the temporary file in the standard Temp directory of the operating system (e.g., <i>\Documents and Settings\CurrentUser\Local Settings\Temp</i>).</p> <p>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.</p>
	Auto Save	<p>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.</p>
Custom Point Selection	Screen	<p>This interface allows you to create your custom dialog to modify or insert pens to the object.</p> <p>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.</p>
	Point Number	<p>Point number (from the Points dialog), indicating the point associated to the pen that will be inserted or modified during runtime.</p>
	Add Indicator	<p>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).</p>
Export to File	Trigger	<p>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.</p>
	File Name	<p>The file path and name of the exported file.</p> <p>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.</p> <div style="border: 1px solid black; padding: 5px;"> <p> Tip: You can specify a project tag in curly brackets (e.g., <code>{ tagname }</code>), to programatically change the file name during runtime.</p> </div>
	Status	<p>The tag configured in this field receives status codes that indicate the success or failure of the export.</p>
	Format	<p>The graphic format of the exported file.</p> <p>If Auto is selected, then the format is determined by the extension specified in File Name. If Auto is selected but no extension is specified, then the default format is BMP.</p>
	Size	<p>The image file is exported at full size by default. However, you can specify the width and height (in pixels).</p>
Decimation	Enable	<p>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. That means for each trend, the X-axis is divided into a number of intervals (determined by Max Points) 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.</p> <p>Decimation only works when the trend control is in Stop Mode (a.k.a. Historical Mode).</p>

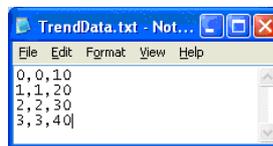
Area / Element Name	Description
	Please note that when this option selected, decimation is done for all trends that are configured to show historical data. To do it for only for a single trend, use Draw Mode in the Options settings.
	Max Points The maximum number of data points used to draw each trend. Default is 2048.
Options	VK (Virtual Keyboard) Virtual Keyboard type used for this object.
	Ignore X Filter When this box is checked, the X Filter is ignored to avoid adding the WHERE or querying clause to the Data Sources.
	Enable translation Enable the external translation for the text displayed by this object.
	Auto Format When checked, decimal values in the Current, Cursor, Max, Min and Scale columns will be formatted according to the virtual table created by the <code>SetDecimalPoints()</code> function. <div style="border: 1px solid black; padding: 5px;"> <p> Note: For the Auto Format to work, decimals formatting on the X-axis must be disabled — that is, the Decimals field in the Axes dialog must be left blank.</p> </div>

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:

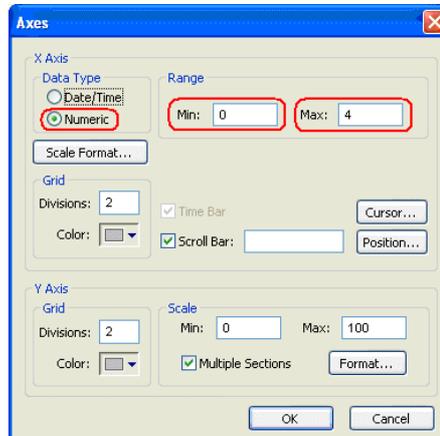
X Value	Y1 Value	Y2 Value
0	0	10
1	1	20
2	2	30
3	3	40

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:

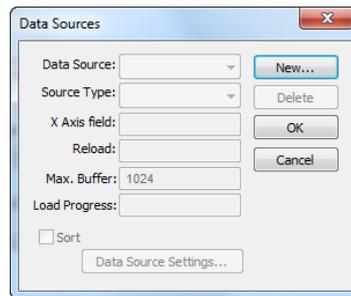


We strongly recommend that you save the file in the same folder where the project is. By doing so, you do not have to specify the entire path and your project will still work, even if it is copied to a different computer.

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:

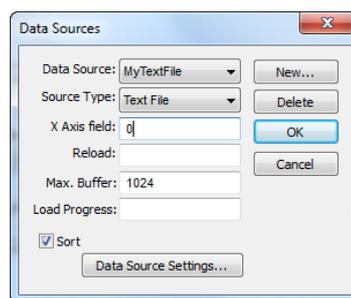


Click **OK** on this window and then, in the *Object Properties* window, click on the **Data Sources...** button. The following window will display:



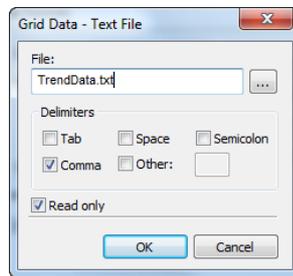
Trend Control – Data Sources dialog

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

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:

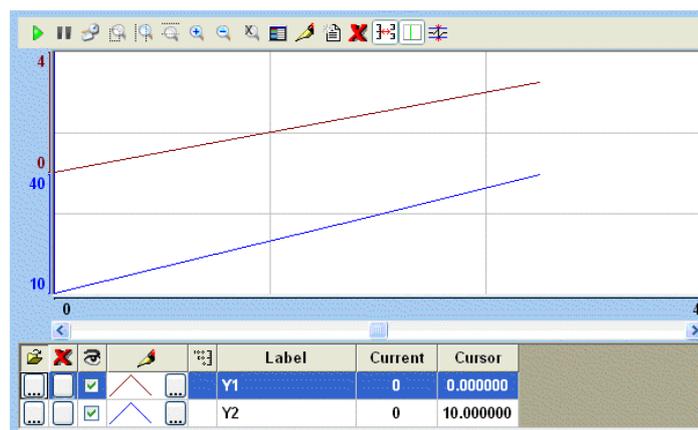
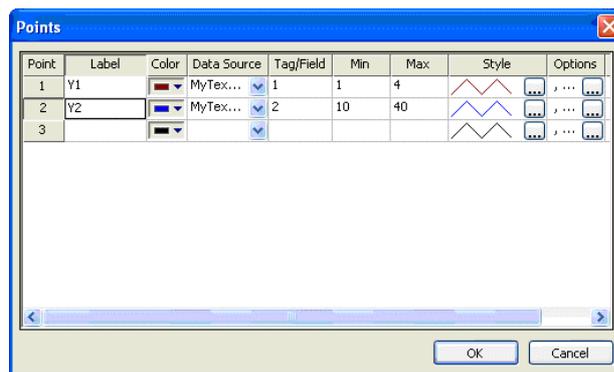


Selecting the text file

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:

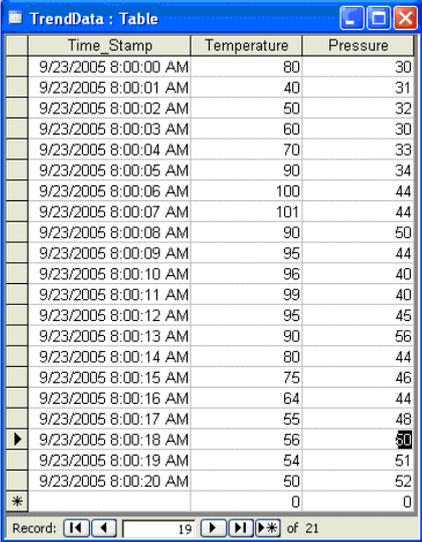


Using the Data Source Database

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

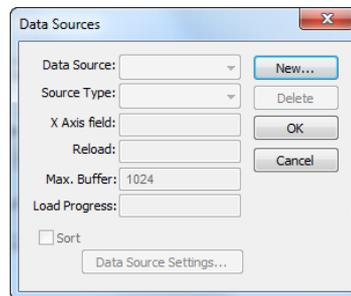
using another type of database, almost all the definitions will apply, however you will need to configure your connection on a different way. For information on how to configure other databases, please refer to the Appendixes in the Database Interface section of this manual.

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:



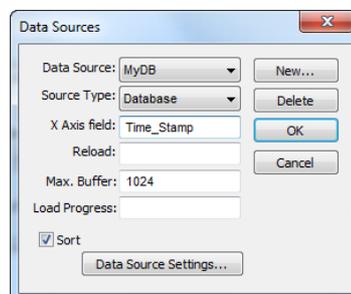
Time_Stamp	Temperature	Pressure
9/23/2005 8:00:00 AM	80	30
9/23/2005 8:00:01 AM	40	31
9/23/2005 8:00:02 AM	50	32
9/23/2005 8:00:03 AM	60	30
9/23/2005 8:00:04 AM	70	33
9/23/2005 8:00:05 AM	90	34
9/23/2005 8:00:06 AM	100	44
9/23/2005 8:00:07 AM	101	44
9/23/2005 8:00:08 AM	90	50
9/23/2005 8:00:09 AM	95	44
9/23/2005 8:00:10 AM	96	40
9/23/2005 8:00:11 AM	99	40
9/23/2005 8:00:12 AM	95	45
9/23/2005 8:00:13 AM	90	56
9/23/2005 8:00:14 AM	80	44
9/23/2005 8:00:15 AM	75	46
9/23/2005 8:00:16 AM	64	44
9/23/2005 8:00:17 AM	55	48
9/23/2005 8:00:18 AM	56	50
9/23/2005 8:00:19 AM	54	51
9/23/2005 8:00:20 AM	50	52
*	0	0

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:



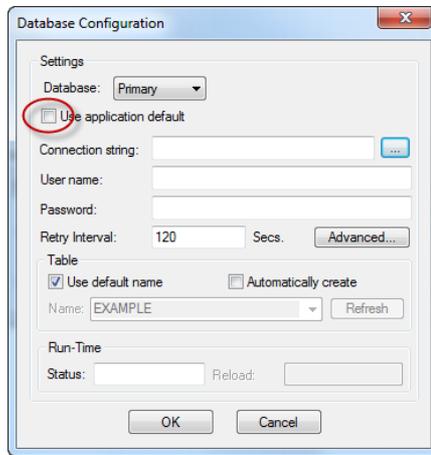
Trend Control – Data Sources dialog

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:



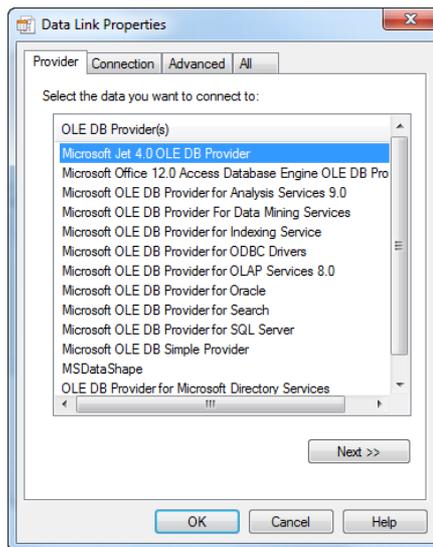
Setting X Axis field to Time_Stamp

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:



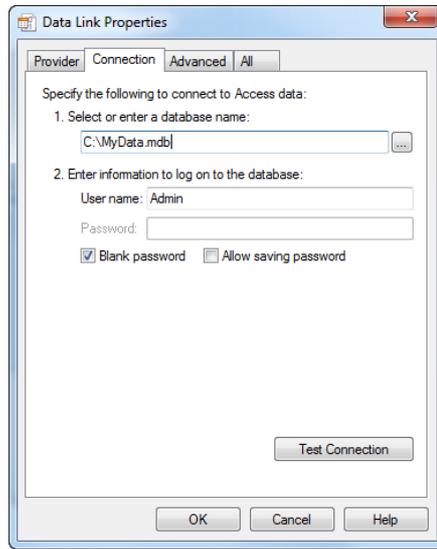
Clearing the Use project default option

Uncheck the checkbox **Use project default** and click on the browse button ... in order to configure the connection string. The following window will display:



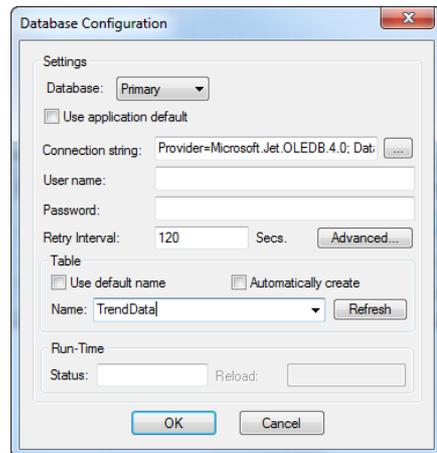
Selecting the OLE DB Provider

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

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:



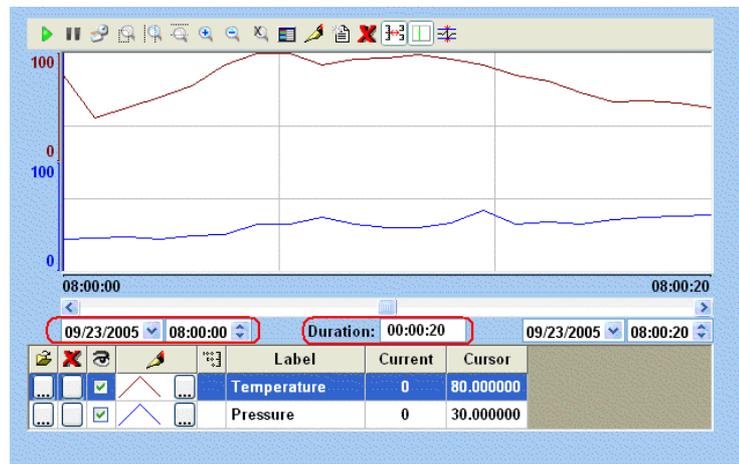
Selecting the table in the database

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:

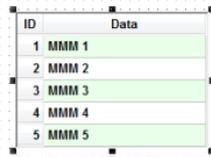


Grid object

The Grid object allows you to read/write data in a tabular format from the data source configured in the object.

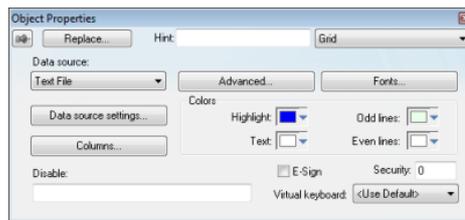
To draw one, do the following:

1. Click the Grid tool 
2. Click on the screen, click the left mouse button, and drag the mouse across the screen to create 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

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:

Data Source	Description
Text File	Displays data from a text file in the ASCII or Unicode format (e.g., CSV text files).
Class Tag	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.
Database	Displays data from an SQL Relational Database, using ADO (ActiveX Database Object) to exchange data with the database.

- **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.
- **VK:** 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.
- **Security:** Enter the security system access level required for the object/animation.
- **Disable:** You can enter an expression in this field to disable data input or action by the user.
- **Highlight Color:** Select a background color for the selected row, during runtime.
- **Text Color:** Select a text color for the selected row, during runtime.
- **Win Color 1:** Select a background color for the odd rows.
- **Win Color 2:** Select a background color for the even rows.
- **Fonts:** Click to launch the *Fonts* dialog, where you can configure the font settings for the text displayed in 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.
- **Data:** Click to launch the [Data dialog](#), where you can specify a data source for the Grid object.
- **Advanced:** Click to launch the [Advanced dialog](#), where you can configure several settings for the Grid object.

Columns dialog

You can configure the settings for each column displayed by the [Grid object](#) during runtime, as follows:



Columns dialog

- **Column:** The ID Number defines the position of the column in the table.
- **Label:** Enter a Title for each column, which will display as the heading (first) row of the Grid object.

 **Tip:** You can configure tags between curly brackets in the **Label** field to modify it dynamically during runtime. When the label is blank (e.g., ""), then the width of the column is set to 0 during runtime. This option is useful to hide columns during runtime.

- **Field:** Enter the name of the field (column) in the SQL Relational Database to which the Grid object is linked. If this field is left in blank, the text configured in the **Label** field will be used as the 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:

Type	Description
Text	Displays alphanumeric values
Numeric	Displays numeric values
Picture	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 Web sub-folder of your project folder to support this feature on the Thin Client stations. CEView projects support pictures in bitmap format (*.bmp), but not in icon format (*.ico).
Checkbox	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.
Time	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 .
Date	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 .
Date/Time	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 .

 **Note:**

- 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 runtime (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 runtime. 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.
- **Align:** Select an Alignment for the data shown in the column. There are three options: Left, Right or Center.
- **Input:** Click (check) to allow the user to enter data in this column during runtime.
- **Key:** Use this field to designate a shortcut for sorting the values. A shortcut is a combination of keys pressed on a keyboard at one time (e.g., CTRL+C, CTRL+V, etc.). This option is especially useful when creating projects for runtime devices that do not provide a mouse or touchscreen interface and only have a keyboard for interacting with the project during runtime.
- **Unit:** Enter the name of the engineering unit (i.e., the unit of measurement), if any, that applies to the data displayed. You can also enter a String tag using the `{tagname}` syntax, which allows you to change the value of **Unit** during runtime.
- **Decimal Points:** Enter the number of decimal places to be displayed. You can also enter an Integer tag using the `{tagname}` syntax, which allows you to change the value of **Decimal Points** during runtime.

 **Note:** When the **Data Source** type is set to Class Tag, and the Columns dialog is left blank, the object displays the values from all members of the Class tag with the following default column settings:

- Label = <Name of the Member from the Class tag>
- Type = Text
- Width = <Minimum size to display the name of the member from the class tag on the header of the grid object>
- Align = Center
- Input = Enabled (checked)
- Key = <None>
- Unit = <Unit of the Member from the Class tag>

The Unit of a class member or tag is set using the [Tag Properties](#) tool.

- **Show ID Column:** Check to display the number of each row, automatically.
- **Allow sorting columns:** Check to enable the user to sort the values in the columns during runtime, either by clicking on the label or by using the shortcut configured for each column. This option is disabled if the **Show header** option from the [Advanced dialog](#) is not checked.

 **Tip:** Use the **Move Up** and **Move Down** buttons to reorder the display of the columns.

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:



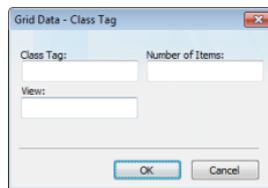
- **File:** Enter the name of the text file source. You can either type the file name and its path or click the ... button to browse for it. (If the file is stored in your project folder, you can omit the path in the name.)

 **Tip:** You can configure tag names between curly brackets {TagName} in the **File** field.

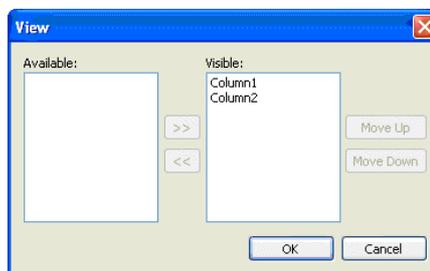
- **Delimiters:** Set the delimiter(s) used in the data source file. For instance, if the data will be read from a CSV (comma separated values) file, you would select the **Comma** option. You can even choose a custom delimiter by checking the **Other** option and typing the custom delimiter in the field 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:

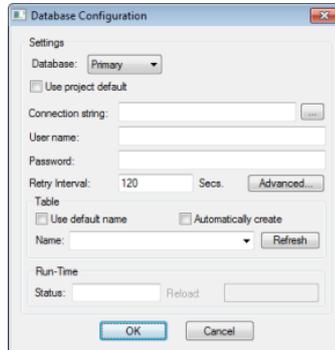


- **Class Tag:** Enter the name of the main class tag source. (Do not specify a specific member of the class tag.) You can specify the initial array position in this field (e.g., Mytag[10]); otherwise, 0 (zero) will be used as the initial position by default.
- **Number of Items:** Enter the number of 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

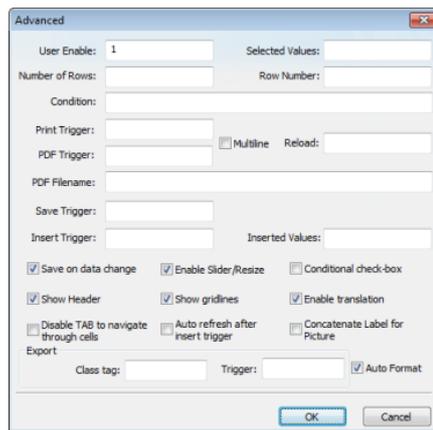
When the **Data Source** type is set to Database, you can configure the following settings:



Please refer to the [Database Configuration dialog](#) for further information about this dialog.

Advanced dialog

This dialog allows you to configure the advanced settings for a Grid object.



Advanced dialog

- **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...

`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

Comparison	Data Source is Text File...	Data Source is Database...
equal to	=	LIKE
not equal to	<>	NOT LIKE
wildcard, single character	?	—
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 uncheck a checkbox on the Grid during runtime, unless all preceding checkboxes in the same column are already unchecked. 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:

Field	Description
Class tag	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.
Trigger	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.

 **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.

Alarm worksheet

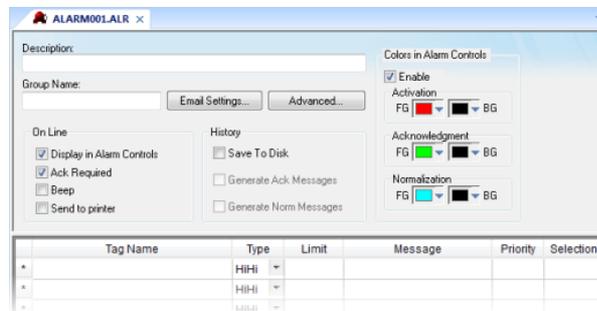
The Alarms folder enables you to configure alarm groups and tags related to each group. The Alarm worksheet defines the alarm messages generated by the project. The primary purpose of an alarm is to inform the operator of any problems or abnormal condition during the process so he can take corrective 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.



Alarm worksheet

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
```



Caution: 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).

Trend worksheet

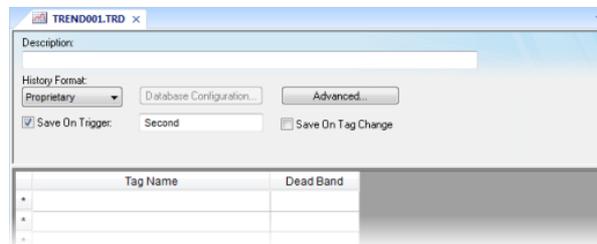
The Trend folder enables you to configure history groups that store trend curves. You can use the Trend worksheet to declare which tags must have their values stored on disk, and to create history files for trend graphs. The project stores the samples in a binary history file (*.hst), and shows both 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 whole group
- Body area (bottom section), where you define each tag in the group. This section contains several columns (only two are shown in the preceding figure).

Use the Header parameters on this worksheet as follows:

- **Description** field: Type a description of the worksheet for documentation purposes.
- **Type** combo box: Click the arrow button to select a trend history format from the list. The available options are:
 - **Proprietary**
 - **File Format:** Binary
 - **Default Path:** ...*project_name*\Hst\GGYYDDMM.HST, where:
 - **YY** = Two last digits of the year
 - **MM** = Month
 - **DD** = Day

 **Note:** IWS provides the HST2TXT.EXE and TXT2HST.EXE programs, which enable you to convert trend history files from binary (**.hst**) to text (**.txt**) and vice versa. For more information about these programs, see [Converting Trend History Files from Binary to Text](#) and [Converting Trend History Files from Text to Binary](#).

- **Database**
 - **Database Type:** Chosen by the user
 - **Default Table Name:** TRENDGGG (GGG = Trend Worksheet Number; e.g., TREND001 for the Trend Worksheet 001)

For more information about the structure of the database table that IWS uses to save history files, see [Database Interface](#).

 **Caution:** You can specify String tags in many fields of the Trend worksheet, to change those values during runtime, but doing so may affect how those values are saved in the trend history:

- When the history format is **Proprietary**, the value of the String tag is converted to a numerical value (if possible) and then saved to the history file. If numeric conversion is not possible, then a value of 0 is saved.
- When the history format is **Database**, the actual value of the String tag is saved in the database.

- **Database Configuration:** Opens the [Database Configuration](#) dialog, where you can enter the requisite settings to link the project to an external SQL Relational Database for the purpose of saving the trend history.
- **Save On Trigger** checkbox and field: 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** checkbox: 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](#).

Use the Body parameters on this worksheet as follows:

- **Tag Name** field: Type the tag name to be saved in the history file.
- **Dead Band** field: 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** field: Name of the field in the database where the tag will be stored. If this field is left blank, the name of the tag will be used as the tag name. Array tags and classes will have the characters "[", "]" and "." replaced by "_". Examples:

Tag Name	Default Field
MyArray[1]	MyArray_1
MyClass.Member1	MyClass_Member1
MyClass[3].Member2	MyClass_3_Member2

 **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, then the Trend task will generate an error when you run the finished project.

See also:

- [Converting Trend History Files from Binary to Text](#)
- [Converting Trend History Files From Text to Binary](#)
- [Creating Batch History](#)
- [Configuring a Default Database for All Task History](#)

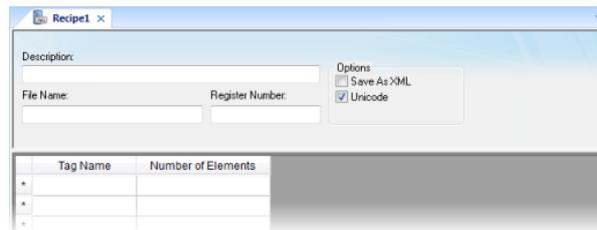
Recipe worksheet

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.



Recipe worksheet

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 an 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, `level1[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.

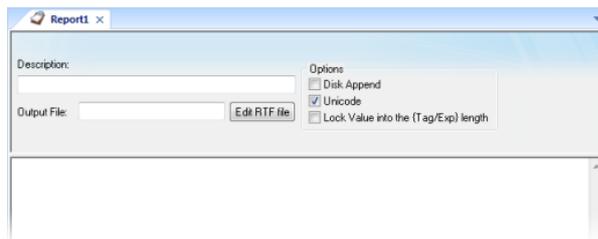
Report worksheet

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.



Report worksheet

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.

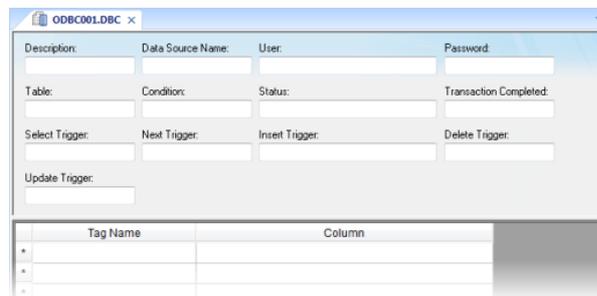
- Note:** In addition to configuring the *ODBC* worksheet, you must configure the Windows ODBC standard driver. IWS refers to the User DSN (Data Source Name), which you configure through the Control Panel. For more information, refer to your Windows documentation.
- Also, the ODBC interface is not available 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.



Description:	Data Source Name:	User:	Password:
Table:	Condition:	Status:	Transaction Completed:
Select Trigger:	Next Trigger:	Insert Trigger:	Delete Trigger:
Update Trigger:			

Tag Name	Column
*	
*	
*	

ODBC worksheet

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.
- **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 ODBCINDCOL 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 ODBCCOMMITE function.
- Delete command
 - 12 - Error in the ODBCPREPARE function.
 - 13 - Error in the ODBCEXECUTE function.
 - 14 - Error in the ODBCCOMMITE function.

Math worksheet

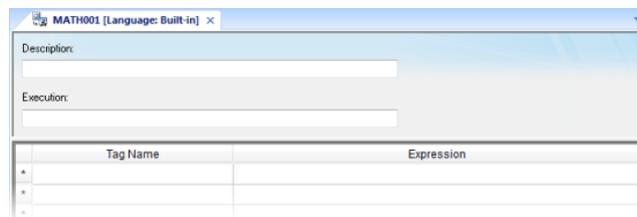
A Math worksheet is used to implement program logic (using the Built-in Scripting Language) that should be continuously executed during runtime, rather than on specific actions like the user pressing a button on a screen.

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.



Math worksheet

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.

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.



Script worksheet

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:

```
'Variables available only for this group can be declared here
Dim myVar, myTest
myTest = 1

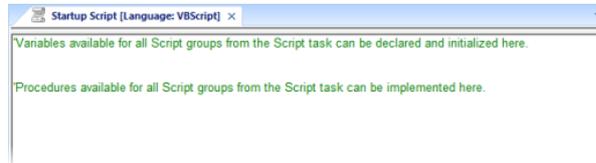
'The code configured here is executed while the condition configured in the Execution
field is TRUE
myVar = $FindFile("c:\*.txt")
If MyVar > 0 Then
    $TagNumOfFiles = myVar
End If
```

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

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:

```
'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
```

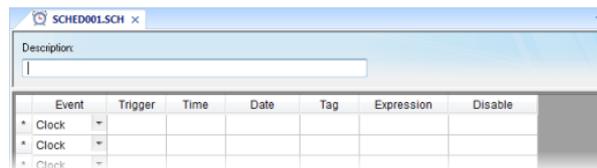
Scheduler worksheet

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.



Scheduler worksheet

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](#).

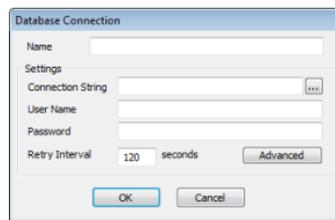
To interface with an external database, you must first [configure a connection](#) to the database and then [build a worksheet](#) that associates project tags with the database fields.

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.

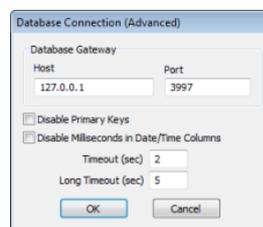


Database Connection dialog

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
```

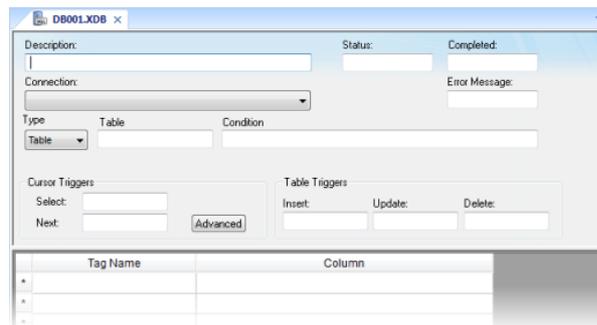
Database Worksheet

 **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 worksheet

Database worksheets are saved in the `\project_name\Config` directory, with the `.XDB` file extension. Each new worksheet is automatically numbered in the order of its creation. For example, the first worksheet created is saved in the file...

```
\project_name\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:

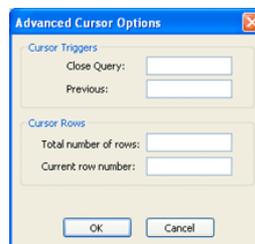
Status codes for external database operations

Status Code	Description
4	Result set is empty
3	Cursor released and query successfully closed
2	Beginning of result set reached, usually while moving cursor to previous row
1	End of result set reached, usually while moving cursor to next row
0	No errors; status normal
-1	Error while connecting to specified database (see Connection below)
-2	Error while selecting result set
-3	Error while moving cursor to next row (see Next trigger below)
-4	Error while moving cursor to previous row (see Previous trigger below)
-5	Error while closing the query (see Close Query trigger below)
-6	Error while inserting rows in result set (see Insert trigger below)
-7	Error while updating result set (see Update trigger below)
-8	Error while deleting result set (see Delete trigger below)

- **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.

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.

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.

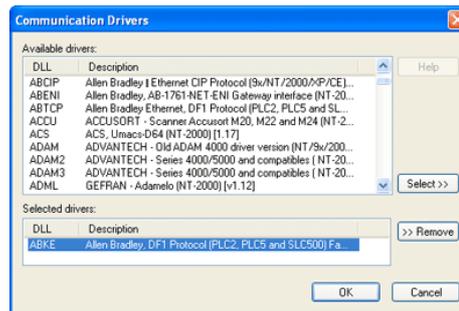
Note: Consult the **Help** menu for a description of the functions and characteristics that are standard for all drivers. When developing a project, you can also refer to the specific documentation provided with each communication driver. This documentation is usually located in the **DRV** directory.

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

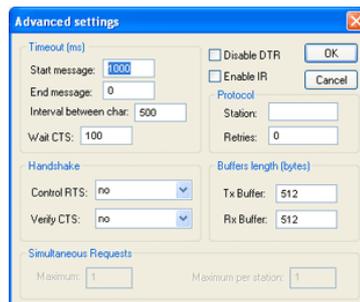
Use the parameters on this dialog, as follows:

- **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.

 **Note:** This section covers only the **None** option, which enables the driver to connect using a normal serial channel. Please refer to "Using TCP/IP and UDP Encapsulation" and "Using Modem Connections" below for more information about other encapsulation modes. "Serial Encapsulation Tests" below lists the drivers that have been tested with modem, TCP/IP and UDP modes.

- **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

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.
 - **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.

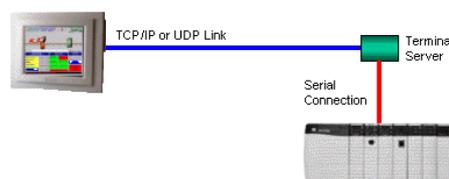
The development application provides two interfaces, which you can use to configure the driver (associating project tags to device addresses):

- **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:

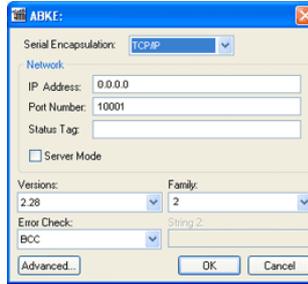


TCP/IP Encapsulation

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.

- In the **Serial Encapsulation** field, select **TCP/IP** or **UDP/IP**:

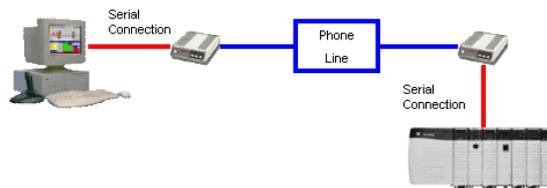


The following fields are available:

- **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

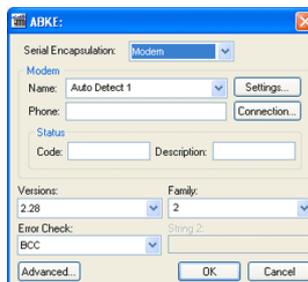
Most of the serial drivers allow the use of modem connections. The modem connection has been designed to enable communications with remote serial devices connected through a phone line. The following diagram provides one example of applying this solution:



Modem Connection

You can enable the modem connection by following the steps below:

- Right-click on the driver's folder, and then choose **Settings** from the shortcut menu.
This will give you access to the communication parameters.
- In the **Serial Encapsulation** menu, select **Modem**:



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

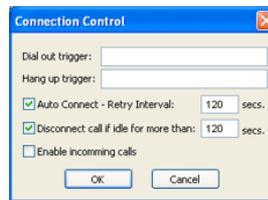
available, **Auto Detect 2** will use the second, **Auto Detect 3** will use the third, and **Auto Detect 4** will use the fourth.

- **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 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.

Driver	Modem	TCP/IP	UDP/IP
MODSL	X	X	X
ABKE	X	X	X
MODBU	X		X
OMETH	X		

X = Item has been tested

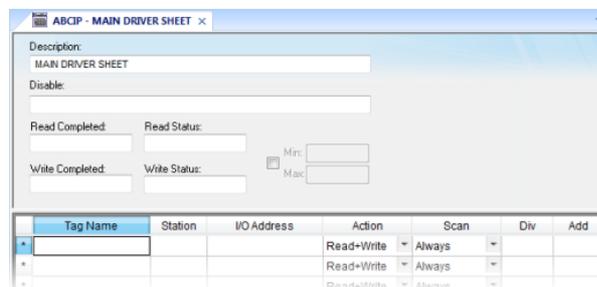
Main Driver Sheet

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

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:

`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.

 **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.

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:

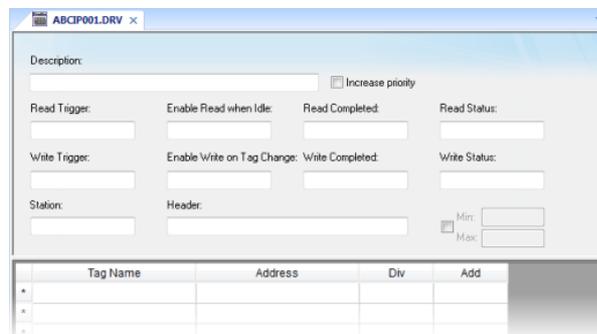
```
[Options]
MainDrvAlwaysTrigger=tagname
```

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.

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

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.

 **Note:** The maximum number of tags in each driver communication worksheet is 512. For some drivers, this number may be smaller. For more information, refer to your [driver documentation](#).

For read operations:

`tag = (value in the equipment) / Div + Add`

For write operations:

`value in the equipment = (tag - Add) * Div`

If you leave the cells empty in the **Div** and **Add** fields, this function is ignored.

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.

Tag Name	Item	Scan	Div	Add
*		Always		
*		Always		
*		Always		

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 an addition factor in a read operation and a subtraction factor in a write operation.

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**.

 **Note:** IWS 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 or CEView. You must configure the OPC Client in one station and you must execute the OPC Server in the other station.

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 in 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.

If you cannot use the same user in both computers because of some specific requirement of your project, or if the problem persists even after you have logged on as the same user, please read the documents below. They will help you solve the security issues:

- [DCOM Security Configuration](#) (external link)
- [Using DCOM with Windows XP + SP2](#) (external link)

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 v7.0\Redist\Wince x.x\processor\

If you do not find the .dll in the folder for your processor, contact your IWS distributor.

Configuring an OPC UA Client connection to an OPC UA Server

The OPC UA Client task/worksheet is to communicate with any system that implements the OPC UA Server protocol.

About OPC UA

The OPC UA Client task/worksheet uses the new OPC Unified Architecture protocol 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 protocol that is also backwards-compatible with OPC "Classic" systems. For more information, go to <http://www.opcfoundation.org/UA/>.

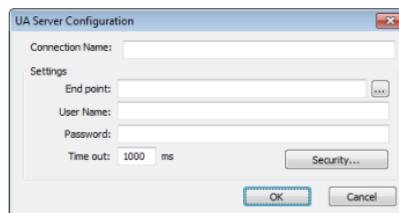
 **Note:** This feature includes cryptographic software written by Eric Young (eay@cryptsoft.com).

Configuring a new OPC UA server 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 *UA Server Configuration* dialog is displayed:

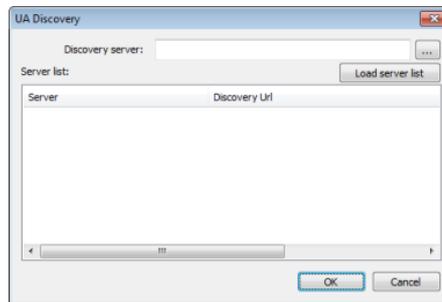


UA Server Configuration dialog

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.

If you don't know the URL, then click the browse button to the right of the box. The *UA Discovery* dialog is displayed:



UA Discovery dialog

Use this dialog 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.
5. If the OPC UA server is configured to require a secure connection, then you must take the extra steps of installing the server certificate in your project and generating a client certificate to be installed on the server:
 - a. Get the server certificate and then save it in the Config sub-folder of your project folder (e.g., [...]\My Documents\InduSoft Web Studio v7.0 Projects\project_name\Config\). The method for getting the server certificate depends on the server, so please consult the server vendor.
 - b. In the *UA Server Configuration* dialog, click **Security**. The *OPC UA Security* dialog is displayed.
 - c. In the **Server Certificate** list, select the server certificate. (All certificates saved in the Config sub-folder should be listed here.)
 - d. In the **Security Policy** and **Message Security Mode** lists, select the appropriate options for how the OPC UA server is configured. Again, please consult the server vendor.
 - e. Click **Generate Certificate**. The client certificate is generated and saved in the Config sub-folder.
 - f. Click **OK** to close the *OPC UA Security* dialog.
 - g. Install the client certificate on the OPC UA server. The method for doing this depends on the server, so again, please consult the server vendor.

For more information, download the white paper "[The OPC UA Security Model for Administrators](#)" from the OPC Foundation.

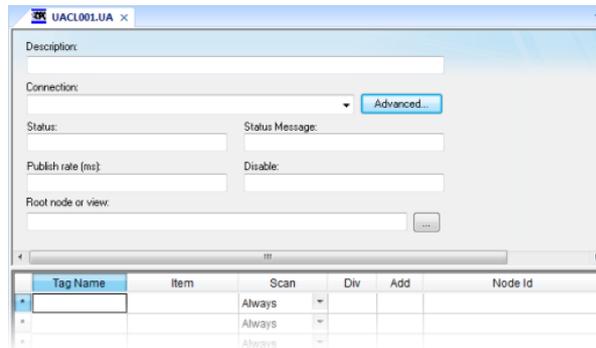
6. Click **OK**. The connection is saved in the OPC UA Connections folder in the Project Explorer.

Configuring 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.

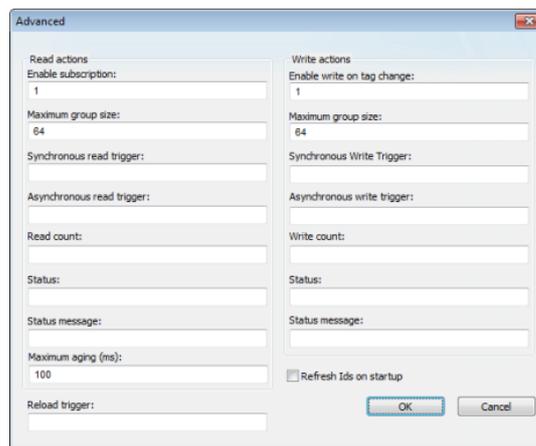
A new OPC UA Client worksheet is displayed:



OPC UA Client worksheet

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 is displayed:



Review the options and configure as needed:

Area / Element	Description
Read actions	Enable subscription When this value is TRUE (non-zero), the client will constantly request updates from the server. <div style="border: 1px solid black; padding: 5px; margin-top: 5px;"> Note: This is enabled by default. If it is disabled, then you must use read triggers (see below). </div>
	Maximum group size The maximum number of tag reads that may be performed 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, then 10 read operations will be performed during each scan of the worksheet.
	Synchronous read trigger When the value of this tag/expression changes, the worksheet is scanned and all tag values are read from the server. The project waits for the scan to complete before continuing.
	Asynchronous read trigger When the value of this tag/expression changes, the worksheet is scanned and all tag values are read from the server. The project continues running without waiting for the scan to complete.
	Read count The name of a tag (Integer type) that will receive a count of the number of read operations performed since the project was run.

Area / Element		Description
	Status	The name of a tag (Integer type) that will receive a status code for the last read operation performed by a trigger.
	Status message	The name of a 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, then the server will be forced to get the latest value from the target device.
Write actions	Enable write on tag change	When this value is TRUE (non-zero), a write will be automatically performed whenever the value of the project tag changes. <div style="border: 1px solid black; padding: 5px;">  Note: This is enabled by default. If it is disabled, then you must use write triggers (see below). </div>
	Maximum group size	The maximum number of tag writes to be performed 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 worksheet is scanned and all tag values are written to the server. The project waits for the scan to complete before continuing.
	Asynchronous write trigger	When the value of this tag/expression changes, the worksheet is scanned and all tag values are written to the server. The project continues running without waiting for the scan to complete.
	Write count	The name of a 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 tag (Integer type) that will receive a status code for the last write operation performed by a trigger.
	Status message	The name of a tag (String type) that will receive the corresponding status message.
Reload trigger	Indirect tags (e.g., { MyTag }) configured in the body of the worksheet will be reloaded only when the value of this tag/expression changes.	
Refresh IDs on startup	When this option is selected, the node IDs in the worksheet will be refreshed from the specified item paths every time the project is run. <div style="border: 1px solid black; padding: 5px;">  Note: Refreshing IDs like this may cause the project to take longer to start up, so if you select this option, then you should also specify a root node (see below) to limit how much of the server's list of items must be scanned. </div>	

 **Note:** IWS's OPC client does not normally use the Triggering Mode that is defined in 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 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 indirect tags (e.g., {**MyTag**}), 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.
- **Node ID:** The node ID is automatically generated from the full path of the server item.

12. Save and close the worksheet.

Enabling 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 is displayed.

2. In the list of tasks, select **OPC UA 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.

Configuring 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.

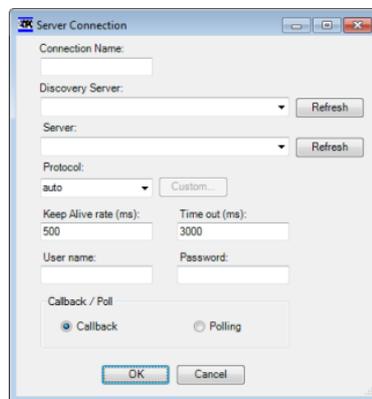
In other words, OPC .NET is intended to be a new version of OPC that leverages Microsoft's latest technologies. For more information, go to [the OPC Foundation website](#).

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:



OPC Server Connection dialog

2. In the **Connection Name** box, type a name for the connection.

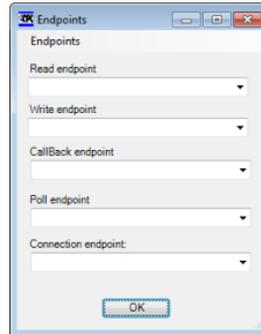
This name will be displayed in the OPC .Net Connections folder in the Project Explorer, and it is the name you will look for when you configure the OPC .Net Client worksheet.
3. In the **Discovery Server** list, type or select the URL of the discovery server that publishes a list of OPC .NET servers on the network. If no discovery server appears to be available, click **Refresh** to scan the network again.

 **Note:** This feature requires that the Peer Name Resolution Protocol service be running on both the client station and the discovery server. PNRP has been built into Microsoft Windows since Windows XP Service Pack 2, so this shouldn't be a problem in most situations.

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** to 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.

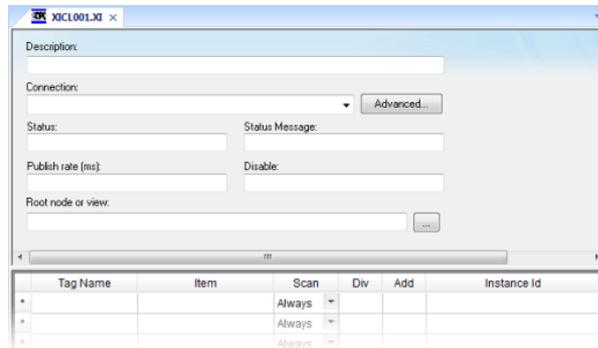
8. Click **OK**. The connection is saved in the OPC .Net Connections folder in the Project Explorer.

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.

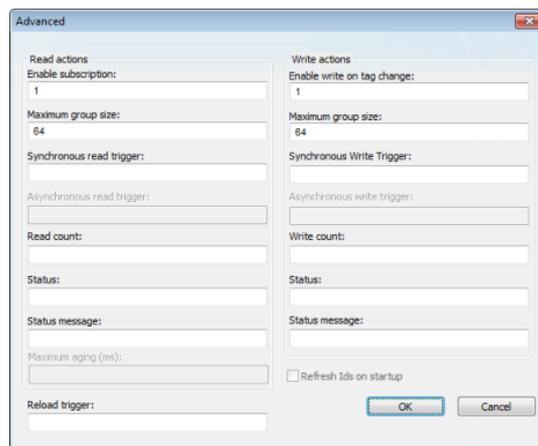
A new OPC .Net Client worksheet is displayed:



OPC .Net Client worksheet

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:



Review the options and configure as needed:

Area / Element	Description
Read actions	Enable subscription When this value is TRUE (non-zero), the client will constantly request updates from the server. <div style="border: 1px solid black; padding: 5px; margin-top: 5px;"> Note: This is enabled by default. If it is disabled, then you must use a read trigger (see below). </div>
	Maximum group size The maximum number of tag reads that may be performed 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, then 10 read operations will be performed during each scan of the worksheet.
	Synchronous read trigger When the value of this tag/expression changes, the worksheet is scanned and all tag values are read from the server. The project waits for the scan to complete before continuing.
	Asynchronous read trigger N/A for OPC .Net Client.
	Read count The name of a tag (Integer type) that will receive a count of the number of read operations performed since the project was run.

Area / Element		Description
	Status	The name of a tag (Integer type) that will receive a status code for the last read operation performed by a trigger.
	Status message	The name of a tag (String type) that will receive the corresponding status message.
	Maximum aging	N/A for OPC .Net Client.
Write actions	Enable write on tag change	When this value is TRUE (non-zero), a write will be automatically performed whenever the value of the project tag changes. <div style="border: 1px solid black; padding: 5px;">  Note: This is enabled by default. If it is disabled, then you must use a write trigger (see below). </div>
	Maximum group size	The maximum number of tag writes to be performed 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 worksheet is scanned and all tag values are written to the server. The project waits for the scan to complete before continuing.
	Asynchronous write trigger	N/A for OPC .Net Client.
	Write count	The name of a 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 tag (Integer type) that will receive a status code for the last write operation performed by a trigger.
	Status message	The name of a tag (String type) that will receive the corresponding status message.
Reload trigger	Indirect tags (e.g., {MyTag}) configured in the body of the worksheet will be reloaded only when the value of this tag/expression changes.	
Refresh IDs on startup	N/A for OPC .Net Client.	

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 indirect tags (e.g., **{MyTag}**), 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 OPC XML-DA communication protocol 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. It also standardizes the SOAP messages exchanged between clients and server, which allows implementation on different operating systems.

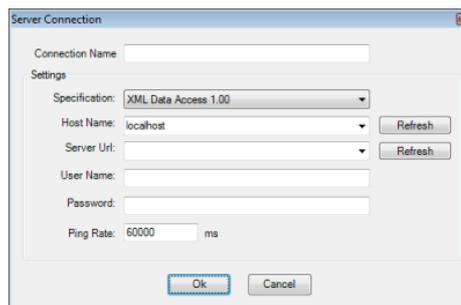
For more information, go to [the OPC Foundation website](#).

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:



OPC XML/DA Server Connection dialog

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.

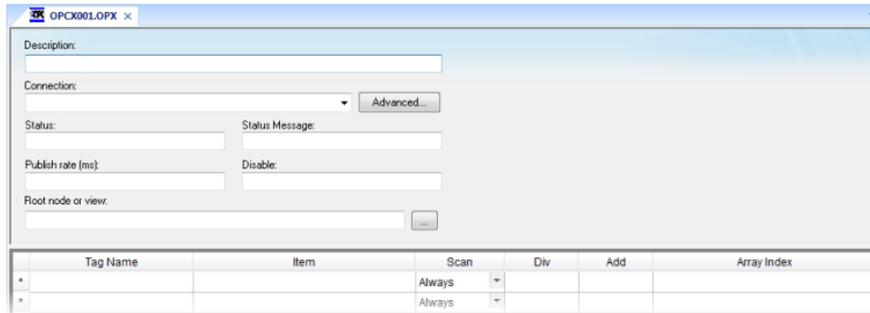
Configuring a new OPC XML/DA Client worksheet

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.

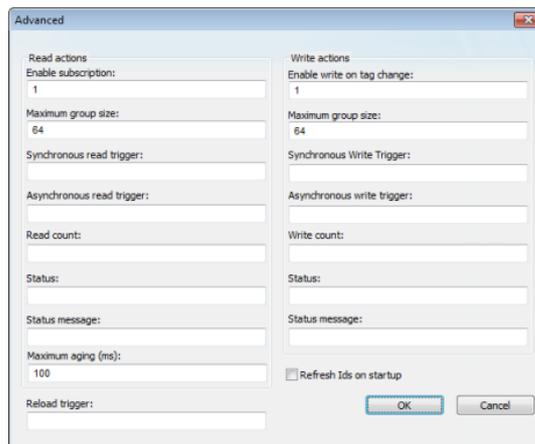
A new OPC XML/DA Client worksheet is displayed:



OPC XML/DA Client worksheet

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 is displayed:



Review the options and configure as needed:

Area / Element		Description
Read actions	Enable subscription	When this value is TRUE (non-zero), the client will constantly request updates from the server. <div style="border: 1px solid black; padding: 5px;"> Note: This is enabled by default. If it is disabled, then you must use read triggers (see below). </div>
	Maximum group size	The maximum number of tag reads that may be performed 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, then 10 read operations will be performed during each scan of the worksheet.
	Synchronous read trigger	When the value of this tag/expression changes, the worksheet is scanned and all tag values are read from the server. The project waits for the scan to complete before continuing.
	Asynchronous read trigger	When the value of this tag/expression changes, the worksheet is scanned and all tag values are read from the server. The project continues running without waiting for the scan to complete.

Area / Element		Description
	Read count	The name of a 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 tag (Integer type) that will receive a status code for the last read operation performed by a trigger.
	Status message	The name of a 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, then the server will be forced to get the latest value from the target device.
Write actions	Enable write on tag change	When this value is TRUE (non-zero), a write will be automatically performed whenever the value of the project tag changes. <div style="border: 1px solid black; padding: 5px;">  Note: This is enabled by default. If it is disabled, then you must use write triggers (see below). </div>
	Maximum group size	The maximum number of tag writes to be performed 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 worksheet is scanned and all tag values are written to the server. The project waits for the scan to complete before continuing.
	Asynchronous write trigger	When the value of this tag/expression changes, the worksheet is scanned and all tag values are written to the server. The project continues running without waiting for the scan to complete.
	Write count	The name of a 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 tag (Integer type) that will receive a status code for the last write operation performed by a trigger.
	Status message	The name of a tag (String type) that will receive the corresponding status message.
Reload trigger	Indirect tags (e.g., {MyTag}) configured in the body of the worksheet will be reloaded only when the value of this tag/expression changes.	
Refresh IDs on startup	When this option is selected, the node IDs in the worksheet will be refreshed from the specified item paths every time the project is run. <div style="border: 1px solid black; padding: 5px;">  Note: Refreshing IDs like this may cause the project to take longer to start up, so if you select this option, then you should also specify a root node (see below) to limit how much of the server's list of items must be scanned. </div>	

 **Note:** IWS's OPC client does not normally use the Triggering Mode that is defined in 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 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 indirect tags (e.g., {**MyTag**}), 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.
- **Array Index:** If the server item is an array, then the array index with which the project tag should be associated.

12. 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.

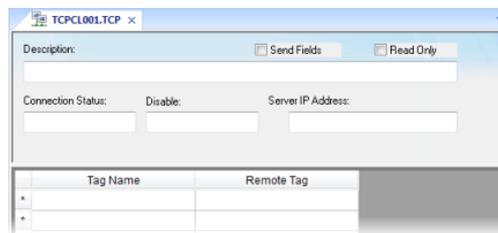
Configuring a TCP/IP Client connection to another project

The TCP/IP Client task/worksheet is used to communicate with another IWS project server.

The TCP/IP Client and Server 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 TCP/IP Client and Server 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 IWS TCP/IP Server 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 IWS TCP/IP Server task: Right-click the **TCP/IP Server** icon in the system tray and click **Exit**.
- To configure the client: You must use the *TCP/IP Client Configuration* program to specify the server IP address and the tags you want to share with the server on the client system.

The *TCP/IP Client Configuration* program is located on the *Comm* tab and it uses the same commands as the *Driver Configuration* program.



TCP/IP Client Worksheet Configuration

Use the following parameters to complete the TCP/IP Client Configuration:

- **Description** field: Type a description of the TCP/IP Client worksheet, for documentation purposes only. The TCP/IP Client task ignores this information.
- **Connection Status** field: 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**: 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** field: 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** checkbox:
 - *Disable* this box and the TCP/IP Client/Server tasks exchange *only* the tag values, and their TimeStamp and Quality.
 - *Enable* this box and the TCP/IP Client/Server tasks *also* exchange the Min, Max, Ack, Unit, LoLoLimit, LoLimit, HiLimit, HiHiLimit, RateLimit, DevSetPoint, DevpLimit, and DevenLimit tag fields.

 **Note:** It is possible to add other fields to the TCP/IP communication or to disable any field individually. Contact your vendor for more information.

- **Read Only** checkbox: 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.
- **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.

You can run the TCP/IP Client task automatically on start up or run the program 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.

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.

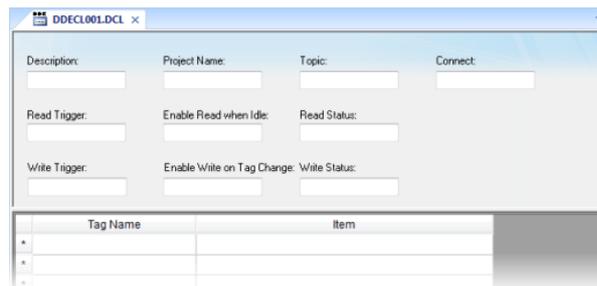
Network Dynamic Data Exchange (NetDDE) is an extension of DDE that works across computers on a network.

- 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 **w**rite 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

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:

- **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
```

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.

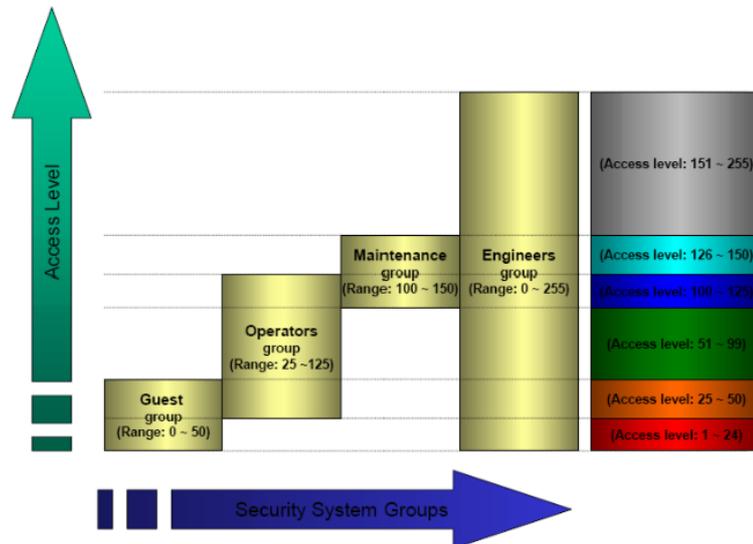
Domain (LDAP)

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.



Example of security access levels

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.

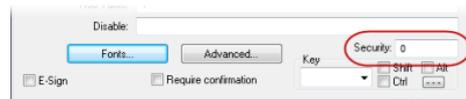
Setting access levels for different types of items

For project screens, the access level can be set in the [Screen Attributes dialog](#).



Security (access level) setting in Screen Attributes dialog

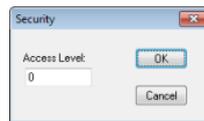
For screen objects and object animations, the access level can be set in the individual [Object Properties dialog](#). (If the option is not available in the main dialog, then it should be available in one of the sub-dialogs.)



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.



Security dialog

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

This page always shows how the security system is currently configured.

2. Click **Next**.

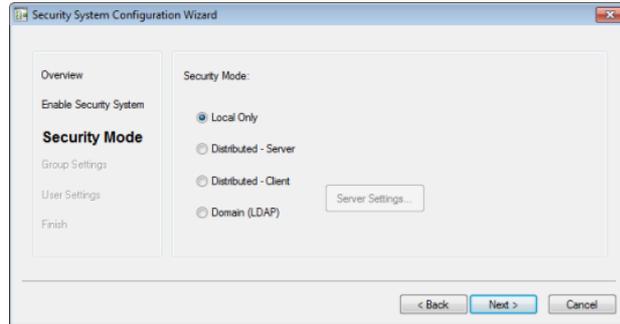
The second page of the wizard is displayed.



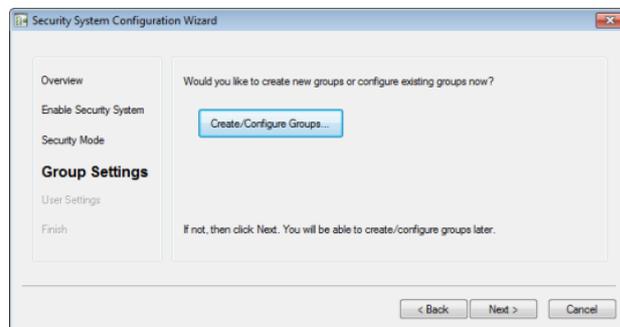
3. Select **Enable Security System**.

4. Click **Next**.

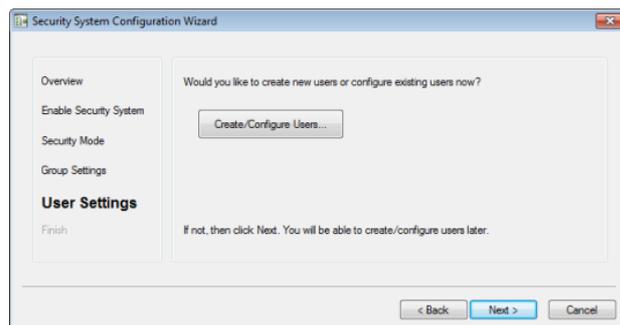
The third page of the wizard is displayed.



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.

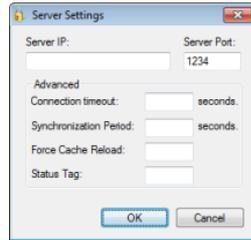
Configuring server settings for security modes

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:



Server Settings dialog for Security Mode: Distributed – Client

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.

While the project is running, the possible values are:

Value	Description
0	No cache
1	Updated cache
2	Outdated local cache
3	Outdated server cache
4	Disconnected from server

For more information, see [GetSecuritySystemStatus](#).

6. Click **OK**.

Configuring the server settings for Security Mode: Domain (LDAP)

To configure the server settings:



Server Settings dialog for Security Mode: Domain (LDAP)

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:

Value	Description
0	Connection timeout
1	Bind timeout
2	Query timeout
3	Disconnected
4	Connected
5	No users or groups returned by query
6	Invalid user or group

For more information, see [GetSecuritySystemStatus](#).

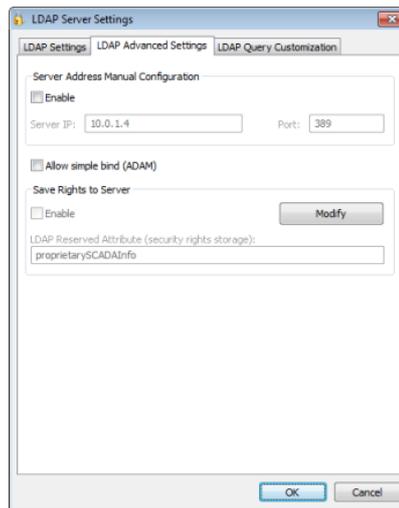
6. Click **Check Connection** to confirm that the project can connect to the specified domain. If it cannot, review and correct your settings.
7. Click **OK**.

 **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 still be able to log onto the project. 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.

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)**.

In the *LDAP Advanced Settings* tab of the dialog, there are additional settings that should be configured only by experienced LDAP administrators:



LDAP Advanced Settings

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

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

LDAP Server	User name attribute	Group name attribute	User lock attribute
Active Directory	sAMAccountName	sAMAccountName	userAccountControl
Active Directory Application Mode (ADAM)	Name	Name	userAccountControl

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.

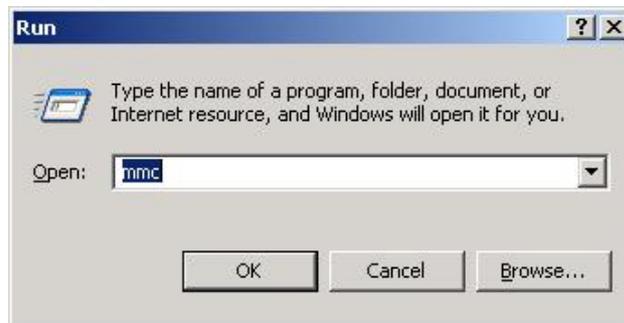
Caution: Extending a server's LDAP schema cannot be undone.

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.



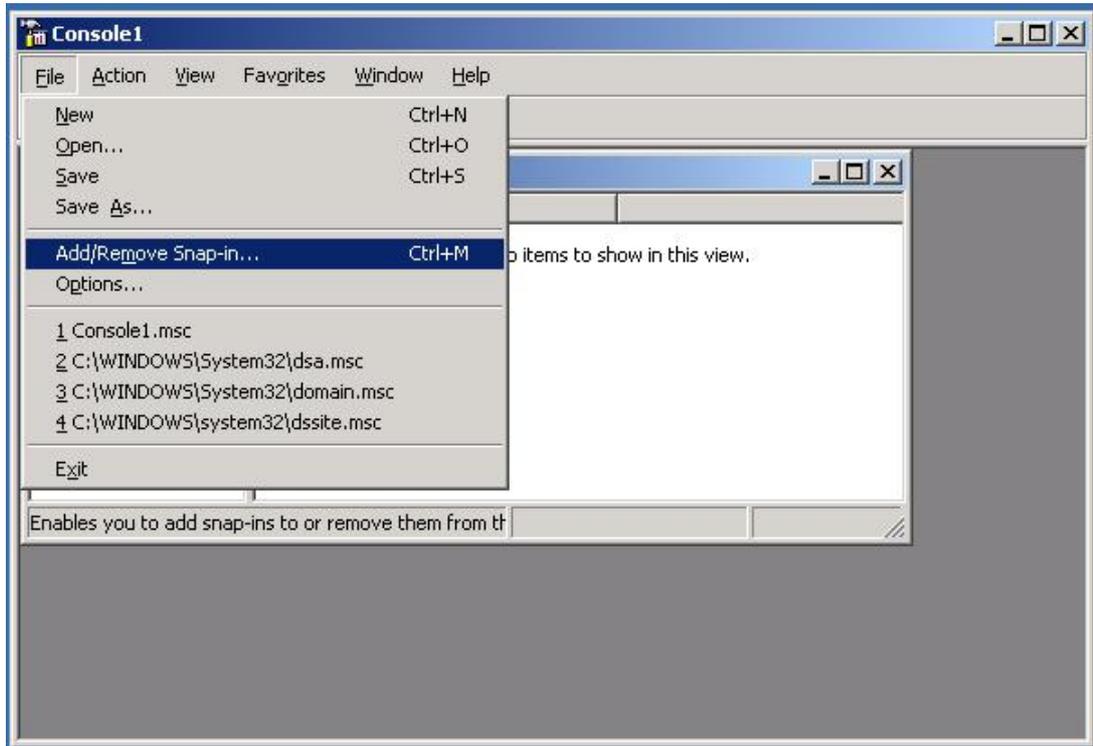
- d) Click **OK** to dismiss the message.
- e) Close the *Command Prompt* window.
2. Add the Active Directory Schema snap-in to the console root.
 - a) Click **Start > All Programs > Accessories > Run**.
A *Run* dialog is displayed.
 - b) In the **Open** box, type `mmc`, and then click **OK**.



(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.

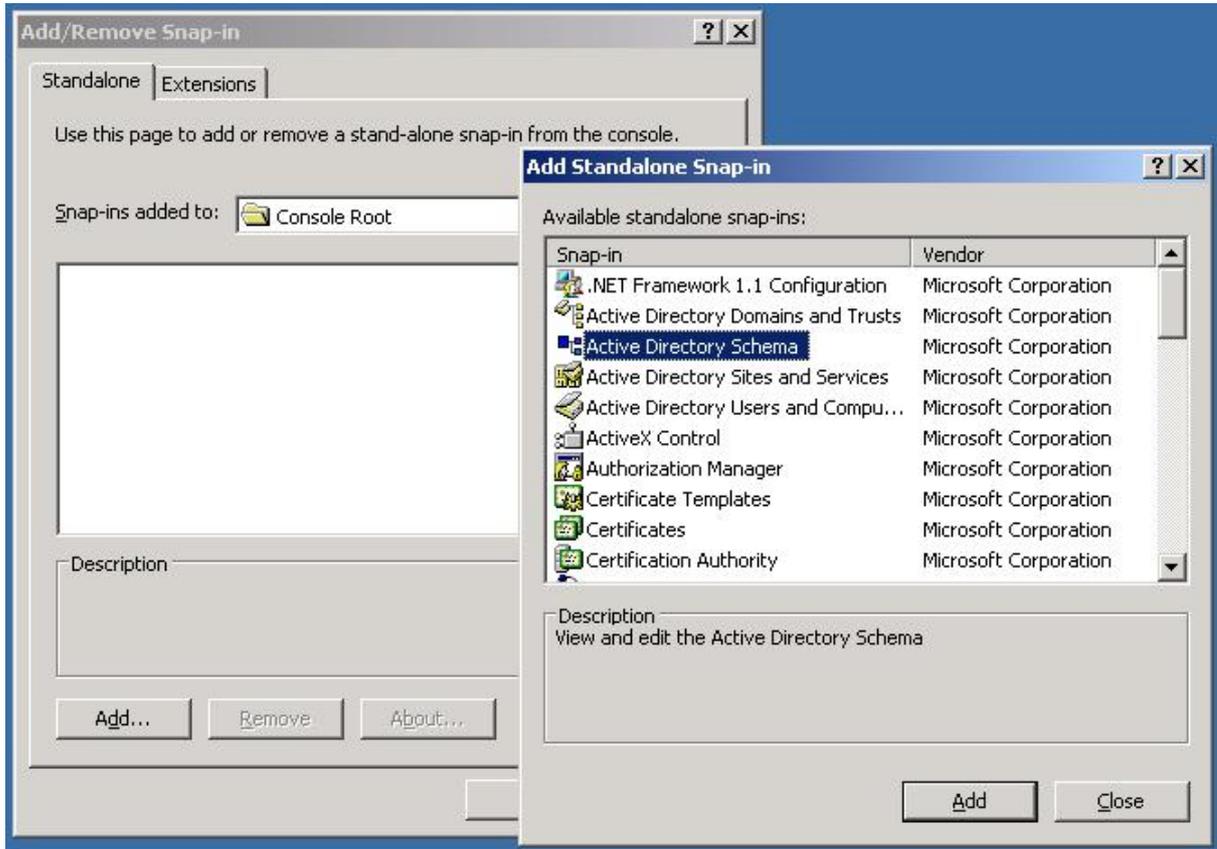
- c) In the console window, click **File > Add/Remove Snap-in**.



The *Add/Remove Snap-in* dialog is displayed.

- d) In the **Snap-ins added to** list, select **Console Root**, and then click **Add**.
The *Add Standalone Snap-in* dialog is displayed.

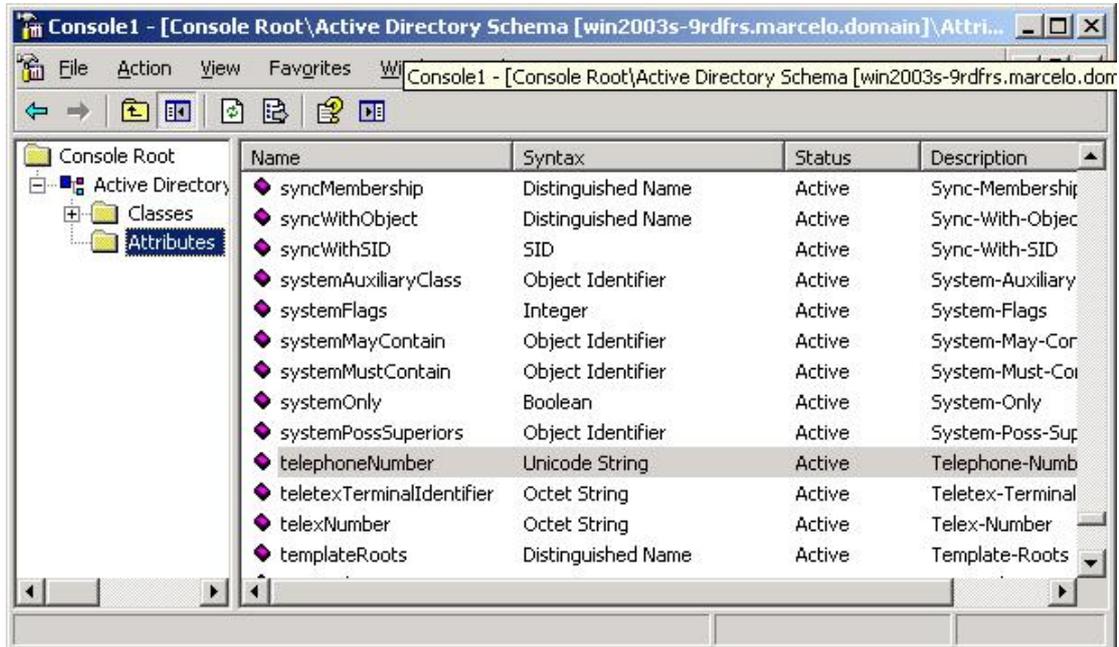
- e) In the list of available snap-ins, select **Active Directory Schema**, and then and click **Add**.



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**.

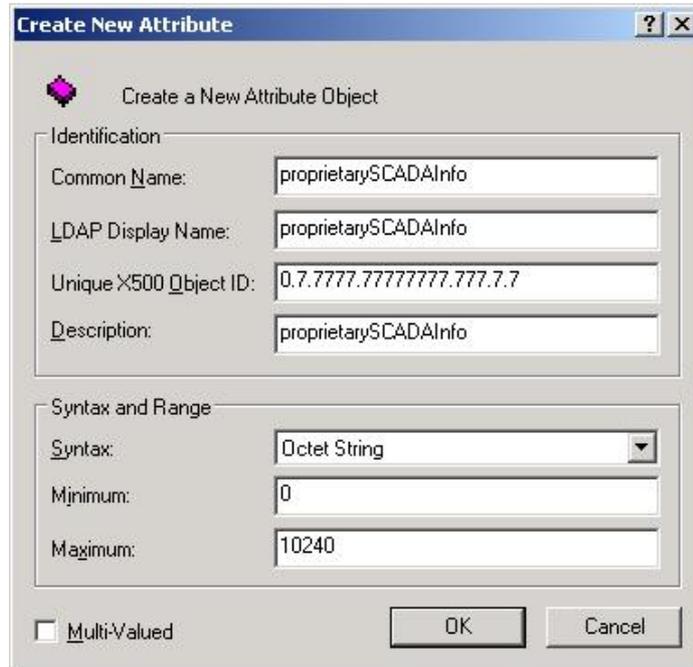


- b) Right-click **Active Directory Schema > Attributes**, and then click **Create Attribute** on the shortcut menu. A message is displayed explaining that your schema will be permanent changed.
- c) Click **Continue**.
A *Create New Attribute* dialog is displayed.
- d) In the dialog, complete the fields as follows.
- **Common Name:** proprietarySCADAInfo
 - **LDAP Display Name:** proprietarySCADAInfo
 - **Unique X500 Object ID:** 0.7.7777.77777777.777.7.7

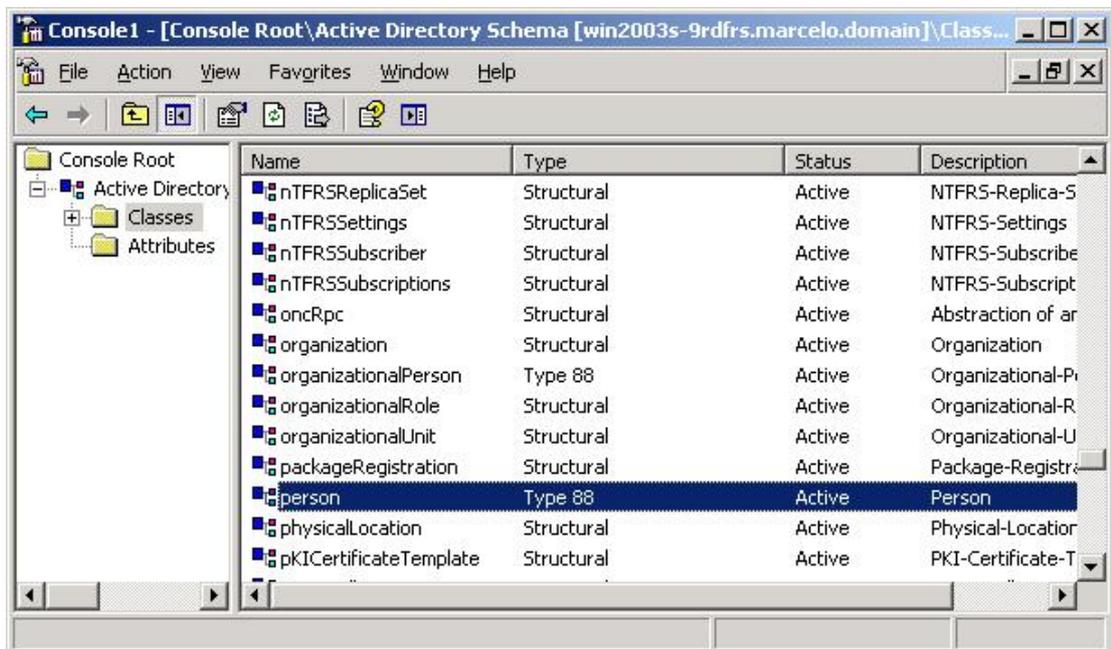
 **Note:** An unique Object ID should be used.

- **Description:** proprietarySCADAInfo
- **Syntax:** Octect String
- **Minimum:** 0

- **Maximum:** 10240

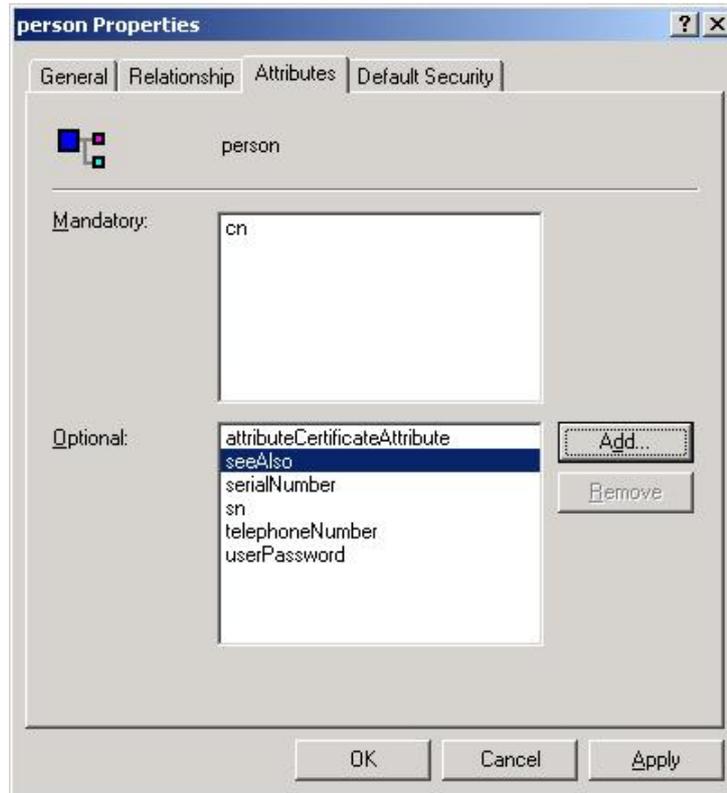


- e) Click **OK** to close the dialog.
The proprietarySCADAInfo attribute is added to the list.
4. Add the proprietarySCADAInfo attribute to the **person** and **group** classes.
 - a) In the **Console Root** tree-view, select **Active Directory Schema > Classes**

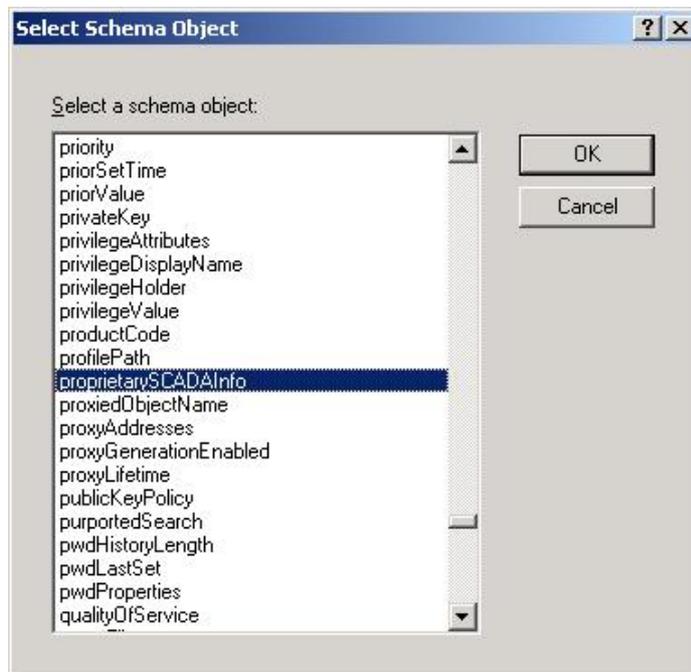


- b) In the list of classes, right-click **person**, and then click **Properties** on the shortcut menu.
The *Properties* dialog is displayed.

- c) In the dialog, click the **Attributes** tab.



- d) Click **Add**.
The *Select Schema Object* dialog is displayed.
- e) In the list of schema objects, select **proprietarySCADAInfo**, and then click **OK**.



- 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

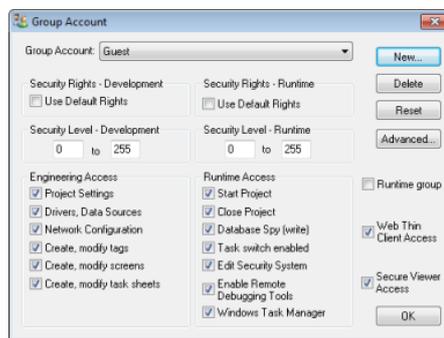
The *Group Account* dialog is used to create and delete user groups, as well as to configure the access privileges for a selected group.

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.

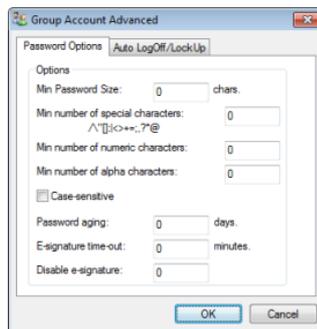
The dialog in detail



Group Account dialog

Area / Element		Description
Group Account		<p>The user group that you are currently configuring.</p> <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <p> Note: There are two default groups for all projects: Guest and (Default Rights).</p> </div> <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <p> Note: 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.</p> </div>
New		Creates a new group. In the <i>New Group Account</i> dialog, type the name of the new group and then click OK .
Delete		Deletes the currently selected group.
Reset		Resets the privileges of the currently selected group to match the (Default Rights) group. This does not lock the group to the default; you can make further changes. To lock the group, see Use Default Rights below.
Advanced		Opens the <i>Group Account Advanced</i> dialog (see below).
Security Rights – Development	Use Default Rights	Locks the development privileges of the currently selected group to those configured for the (Default Rights) group. If changes are made to the (Default Rights) group, then they also apply to this group.
Security Level – Development	Range	The range of access levels that this group may access in the development application.
Engineering Access	Project Settings	Members of the group may modify the project settings .
	Drivers, Data Sources	Members of the group may create, modify device drivers and external data sources .
	Network Configuration	Members of the group may create, modify TCP/IP Client worksheets .

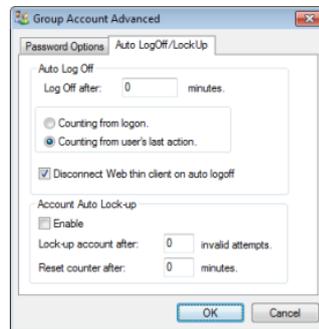
Area / Element		Description
	Create, modify tags	Members of the group may create, modify project tags .
	Create, modify screens	Members of the group may create, modify project screens .
	Create, modify task sheets	Members of the group may create, modify task worksheets .
Security Rights – Runtime	Use Default Rights	Locks the runtime privileges of the currently selected group to those configured for the (Default Rights) group. If changes are made to the (Default Rights) group, then they also apply to this group.
Security Level – Runtime	Range	The range of access levels that this group may access in the runtime project.
Runtime Access	Start Project	Members of the group may run the project.
	Close Project	Members of the group may stop the project.
	Database Spy (write)	Members of the group may write values to the project database using the Database Spy window . <div style="border: 1px solid black; padding: 5px;"> <p> Note: This only applies to projects running locally. For projects running remotely, see Enable Remote Debugging Tools below.</p> </div>
	Task switch enabled	Members of the group may switch away from the runtime project to another Windows task. (This option does not apply to projects running on Windows Embedded target systems; the user may always switch away from the runtime project.)
	Edit Security System	Members of the group may make changes to the project security system during runtime. <div style="border: 1px solid black; padding: 5px;"> <p> Note: Be careful not to clear this option for your own group, or you may not be able to undo your own changes.</p> </div>
	Enable Remote Debugging Tools	Members of the group may use Remote Database Spy and Remote LogWin .
	Windows Task Manager	Members of the group may open the Windows Task Manager. (This option does not apply to projects running on Windows Embedded target systems; the user may always open the Windows Task Manager.) <div style="border: 1px solid black; padding: 5px;"> <p> Note: Clearing this option means disabling the Task Manager during runtime, which requires Administrator privileges. You will need to run the project with elevated privileges.</p> </div>
Runtime group	A user created during runtime by calling the CreateUser function may be assigned to this group.	
Web Thin Client Access	Members of the group may access the runtime project by using a Web Thin Client.	
Secure Viewer Access	Members of the group may access the runtime project by using a Secure Viewer.	



Advanced dialog – Password Options

Area / Element	Description
Min password size	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

Area / Element	Description
Min number of special characters	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.
Min number of numeric characters	
Min number of alpha characters	
Case-sensitive	<p>If this option is selected, then passwords are case sensitive — that is, passwords created with both upper and lowercase characters <i>must</i> be entered the same way by the user.</p> <div style="border: 1px solid black; padding: 5px;"> <p> Note: In projects created with InduSoft Web Studio v6.1+SP2 through v6.1+SP6, all passwords were case sensitive.</p> </div>
Password aging	<p>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 <i>Change Password</i> dialog is automatically displayed and the user cannot complete the logon process until he provides a new password.</p> <p>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 (<i>project_name . app</i>) to include the following setting:</p> <p>[Security] ChangePasswordMode=1</p> <p>To make passwords never expire, set Password aging to 0.</p>
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	<p>When the value in this box is TRUE (non-zero), users in this group cannot use any project features that require an e-signature.</p> <p>You can configure a project tag in this box, so that e-signature is dynamically enabled/disabled during runtime.</p>



Advanced dialog – Auto LogOff/LockUp

Area / Element	Description	
Auto Log Off	Log Off after	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.
	Counting from logon	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.
	Counting from user's last action	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.
	Disconnect Web Thin Client on auto logoff	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.
Auto Lock-up	Enable	Enables the auto lock-up features described below.

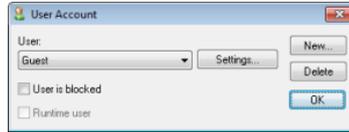
Area / Element		Description
	Lock up account after	Maximum number of times a user can try to log on to an account. If the user exceeds the specified maximum number of attempts (provides an invalid password) within the period of time specified in the Reset counter after field, the project will automatically block the user.
	Reset counter after	Defines how long after an invalid log-on attempt the project will wait (in minutes) until it resets the log-on attempts counter.

-  **Note:** 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 above 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**.)

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.



User Account dialog

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.

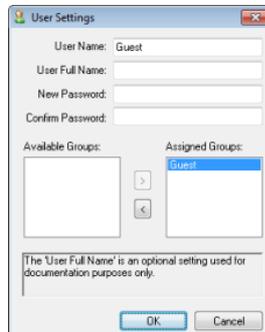
To delete a user, click the **User** combo-box button, select the user name from list, and then click **Delete**.

To configure a user, use the following procedure:

1. Click the **User** combo-box button and select a user from the list.
2. If necessary, click the **User is blocked** checkbox to block the selected user.

 **Note:** The **Runtime user** option indicates that a user was created during runtime using the [CreateUser](#) function. It cannot be selected for a user created using the procedure described here.

3. Click the new **Settings** button to open the *User Settings* dialog:



User Settings dialog

4. 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.
5. 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.
6. 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



Security System dialog

Area / Element		Description
Enable Security System		Indicates whether the project security system is currently enabled. If it is, then the users and groups' specified access privileges are enforced.
Main Password		Opens a dialog where you can specify a main administrative password for the entire project.
Security Mode	Mode	The current security mode of the project.
	Server Settings	Opens the Server Settings dialog , where you can configure the server settings for Distributed – Client or Domain (LDAP) .
Run Wizard		Opens the security system configuration wizard .
Backup		Opens the Import/Export dialog , where you can export or import the security system configuration.
Accounts Management	Groups	Opens the Group Account dialog , where you can create and configure groups.
	Users	Opens the User Account dialog , where you can create and configure users.
	Display list of users at logon	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.
	Log On on E-Signature	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.
	Default User	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.

Area / Element		Description
		 Note: This user's privileges should be heavily restricted, to prevent your project from being left vulnerable.
Virtual Keyboard		The type of virtual keyboard that is displayed on the client when the user is prompted to log on.

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.

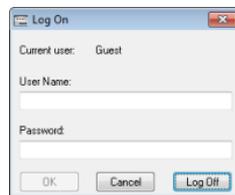
Note: The project security system must be enabled before you can use this feature.

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

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

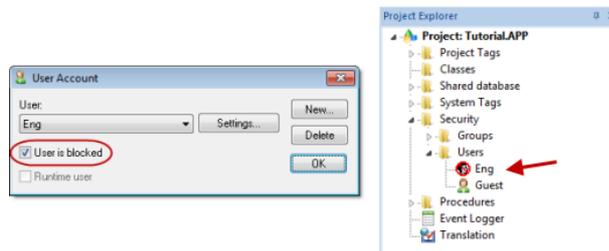
An individual user in the project security system may be completely blocked from accessing the project, and a blocked may subsequently be unblocked.

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.



User is blocked

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 re-use 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.

Automatic Translation

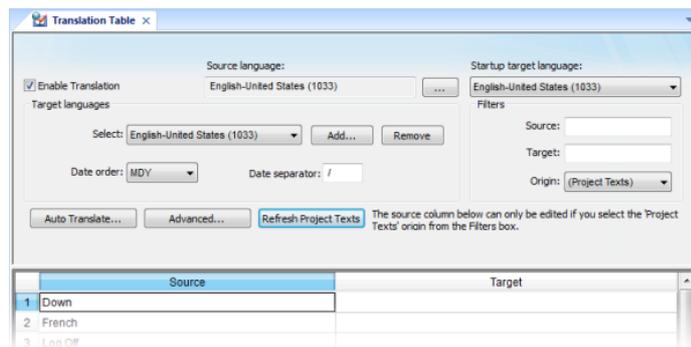
You can quickly translate your project's user interface to multiple languages, leveraging the power of automatic translation through Google Translate, and you can switch your project's language during runtime with a simple function call.

Adding a language to the Translation Table

The Translation Table is used to manage the languages to 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.

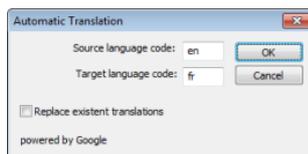
1. Open the Translation Table worksheet by doing one of the following.
 - On the Insert tab of the ribbon, in the Global group, click **Translation**; or
 - In the Global tab of the Project Explorer, double-click **Translation**.

The Translation Table worksheet is opened for editing, with the Source column already populated with all of the translatable text strings in your project.



Translation Table worksheet

2. Make sure the **Enable Translation** option is selected.
3. In the **Target languages** area, click **Add**.
The *Languages* dialog is displayed.
4. In the *Languages* dialog, select the language to which you want to translate your project and then click **OK**.
The language is added to the **Select** list.
5. Configure **Date order** and **Date separator** as desired 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**.
6. Click **Auto Translate**.
The *Automatic Translation* dialog is displayed.



Automatic Translation dialog

7. Confirm the source and target language codes.
Language codes are defined by [ISO 639-1](#).
8. If you want to overwrite any previous translations, select **Replace existent translations**.
Please note that this will overwrite both automatic translations and manual edits.
9. Click **OK**.
The application processes the **Source** column of the worksheet through [Google Translate](#) and then populates the **Target** column with the results.
10. Review and manually edit the translation results as needed.
Use the **Filters** to search the worksheet for a text string; as you type a few characters, the list is dynamically filtered to show only the strings that match.

11. Save and close the worksheet.

Changes made to the Translation Table will not take effect until you either call the [SetLanguage function](#) or restart the runtime project.

 **Tip:** The Translation Table is saved as a tab-delimited text file in your project folder, at [...] \My Documents \InduSoft Web Studio v7.0 Projects \project_name \Web \Translation.trn. You can open and directly edit this file with Microsoft Excel, if you choose to do so.

Setting 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 [Adding a language](#).

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've added.
3. Save and close the worksheet.

Setting the project's language during runtime

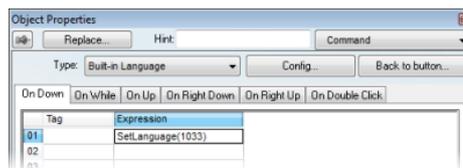
You can set your project's language during runtime by using the `SetLanguage` function anywhere that an expression can be configured.

This procedure assumes that you have already added at least one target language to the Translation Table. For more information, see [Adding a language](#).

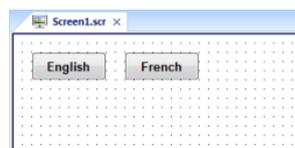
The `SetLanguage` function takes one parameter: a numeric code representing the target language. The codes are shown in parentheses in the list of languages — for example, `English-United States` is 1033.

The following very basic 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)`.



8. Close the *Object Properties* dialog.
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 runtime, clicking each button will set the language of the entire project to that language, using the translated text from the Translation Table.

Disabling 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.

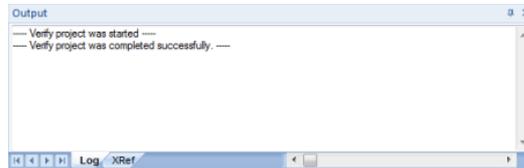
1. Double-click a screen object to open its *Object Properties* dialog.
2. In the dialog, look for the **Enable translation** option.
If the option is not available in the object's basic properties, then click **Advanced** to access the advanced properties
3. Clear the **Enable translation** option.
4. Close the *Object Properties* dialog.

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.

Testing and Debugging

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

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 (☒): 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 (⌵): 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.

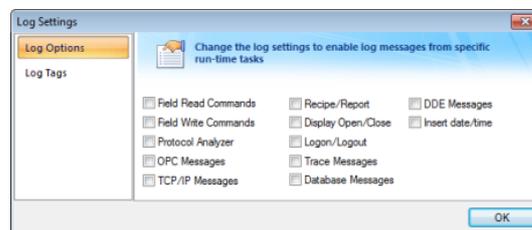
The *Output* window is dockable, which means you can move it to another location in the development environment. Click on the title bar and drag the window to a new location. Release the mouse button to attach or *dock* the window to its new location.

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.

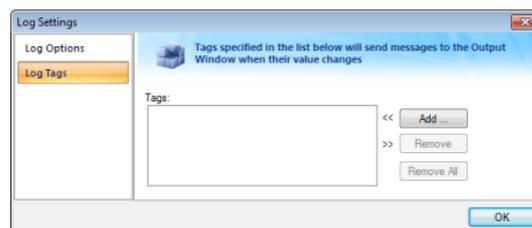


Log Settings — Options Tab

Option	Description
Field Read Commands and Field Write Commands	Show any read and/or write commands that are sent to connected devices.

Option	Description
Protocol Analyzer	Show messages generated by configured device drivers .
OPC Messages	Show messages generated by OPC communications.
TCP/IP Messages	Show messages generated by TCP/IP communications.
Recipe/Report	Show messages generated by the Recipe and Report tasks.
Display Open/Close	<p>Display detailed information whenever a screen is opened or closed:</p> <ul style="list-style-type: none"> • Disk Load Time: Time to load the screen file from the disk into memory. • Open Time: Time to open the screen, including initializing tags used in the screen and running any "OnOpen" scripts or functions. • Total Load Time: Total time to load the screen (includes Disk Load Time and Open Time above). • First Draw Time: Time to first drawing of screen objects. • First OnWhile Time: Time to first running of any "OnWhile" scripts or functions. • Total Open Time: Total time to open the screen (includes First Draw Time and First OnWhile Time above). • Close Time: Time to close the screen, including finalizing tags used in the screen and running any "OnClose" scripts or functions. • Total Close Time: Total time to close the screen, including the time to close the screen file on the disk. <p>This information can be used to analyze runtime performance on low-end target systems. If a particular step of opening or closing takes an unusually long time, then it can be identified and redesigned.</p>
Logon/Logout	Display a message whenever a user logs on or logs out. (For more information, see Security .)
Trace Messages	Show messages generated by the Trace () function. This function is used to generate customized messages from within your project.
Database Messages	Show messages generated by the ODBC and ADO.NET database interfaces.
DDE Messages	Show messages generated by DDE communications.
Insert date/time	Timestamp each message.

- In the *Log Tags* tab of the dialog, click **Add** to browse for [project tags](#).



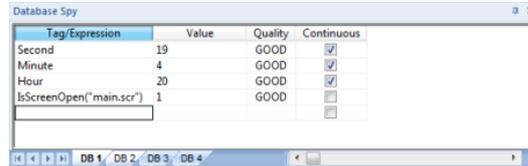
Log Settings — Tags Tab

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

- 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.



Tag/Expression	Value	Quality	Continuous
Second	19	GOOD	<input checked="" type="checkbox"/>
Minute	4	GOOD	<input checked="" type="checkbox"/>
Hour	20	GOOD	<input checked="" type="checkbox"/>
IsScreenOpen("main.scr")	1	GOOD	<input type="checkbox"/>

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 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.

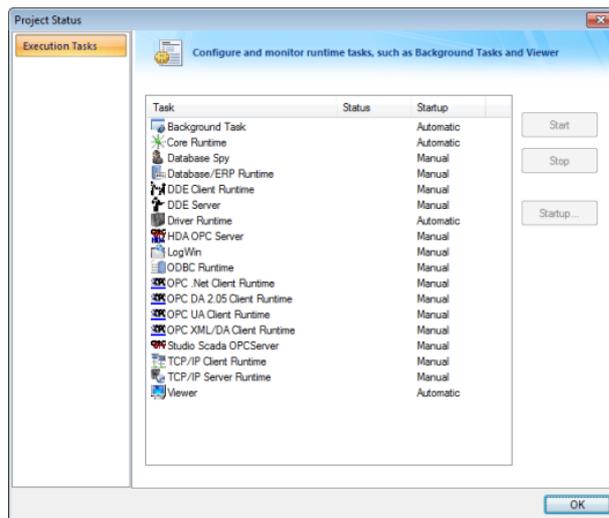
 **Tip:** The Database Spy is dockable, which means you can move it to another location in the development environment. Click on the titlebar and drag it to a new location. Release the mouse button to attach or dock the window to its new location.

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 DDE, OPC, and TCP/IP transactions, activate modules, trace tags, and so forth.

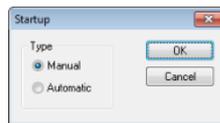
To enable the LogWin module as a part of your project:

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 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 (located in the [Tools menu](#)) to monitor the **Database Spy** of an IWS 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

 **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

 **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 (located in the [Tools menu](#)) to monitor the **Output log (LogWin)** of an IWS 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.

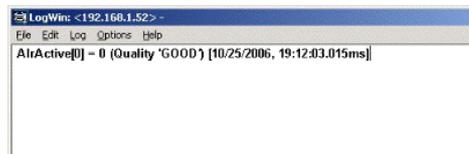
2. Enter the IP address of the remote computer, as shown below.



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.



Remote LogWin

When you are done, just close the window to disconnect.

Deploying as a Web Application

Introduction to Thin Clients

IWS is built on a Client/Server architecture, and you can support both Thin Clients and Thick Client. The choice of the type of Client architecture depends upon your system requirements.

Thick Client

A Thick (Fat) Client is a computer that performs most, if not all, of the processing activity. The Thick Client has sufficient processing power, memory, graphics, etc. to run the project. The Thick Client can exchange data with the Server as required (e.g., archival of data, program synchronization). A Server can also be a Thick Client to another Server.

Thin Client

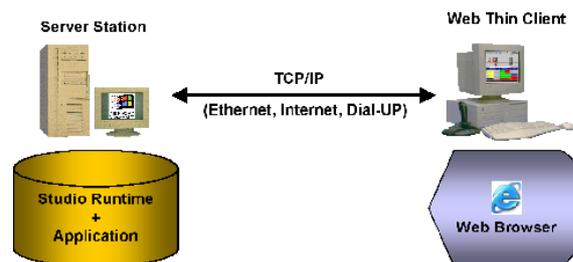
A Thin (Lean) Client is a computer that depends primarily on the central Server for processing activities. The Thin Client needs to have a processor with minimal memory, a browser (Internet Explorer) and a network connection (e.g., to the Internet or Intranet). We use the term "Web Client" to refer to a Thin Client solution. IWS lets you choose the type of browser that is best for your runtime environment:

- Microsoft Internet Explorer. This type of browser is best for desktop or remote access environments.
- InduSoft's Secure Viewer, a dedicated viewer for IWS projects. Secure Viewer does not let the user browse to other websites, and will only connect to a specified IWS Server. Secure Viewer is intended for plant floor environments where browsing to other web sites is not required or desired.

In this chapter, we will explore using Thin Clients with IWS. Thick Clients will be covered in a subsequent chapter.

Thin Clients in IWS

IWS allows you to create screens that can be visualized in a remote station a regular web browser (e.g., Internet Explorer). The station where the user can visualize the graphical interface (screens) on the web browser is called the Thin Client. The Thin Client (Browser) is the host.



Typical Thin Client Architecture

IWS is installed on the Server station only. Also, the project (screen files, tags database, configuration worksheets, etc.) is stored on the Server only. You do not need to install the development application or the runtime project on the Thin Client. This solution provides a high level of flexibility because any computer physically linked to the Server station (via a TCP/IP link) can access the graphical screen and online/history data from the Server without installing the development application or the runtime project on the Thin Client stations.

All background tasks (Math, Scheduler etc) and communication tasks (e.g., Driver, OPC, DDE TCP/IP) are executed only on the Server station. The Thin Client is able to load the graphical interface configured on the Server (screens with objects and animations) and display the online values from the tags configured in the server, as well as history data (Alarm, Events and Trend history data).

Note: Since VBScript and the IWS Scripting Language can be associated with a screen as well as a Command animation for an Object, these Scripts can execute on a Thin Client.

Any computer or device (e.g., a PDA powered by Windows Mobile 5.0) running Microsoft Internet Explorer 6.0 (or later) or our Secure Viewer program can be a Thin Client for your project. Moreover, IWS provides a sophisticated **Security System** to prohibit unauthorized access to the project.

From the Thin Client, you are able to visualize data from the Server and capable of changing set points, acknowledging alarms and/or sending commands to the Server. When configuring the project, you can optionally enable or disable all commands from the Thin Client to the Server. Even if the Server has

disabled all command from the Thin Client, the Thin Client can still read data from the Server but cannot send any data to the Server.

Thin Client Competitive Advantages

IWS is built on a Client/Server architecture that supports **true** Thin Clients. This capability is built into IWS and is not an add-on. This means that:

- The IWS Server supports a large number of concurrent Thin Clients (up to 128 IE-based and 128 Secure Viewer-based). Each Thin Client can view the same or different screens as another Thin Client.
- The IWS Server knows which display each Thin Client viewing and automatically "pushes" any updated Tag values to the Thin Client, keeping the Thin Client display current and eliminating the need for screen refreshes
- The IWS Server can support runtime language switching for each Thin Client. This means that one Thin Client can be viewing a screen in English while another Thin Client can display the same screen in Spanish.

Many competitive products offer either a static display on a Thin Client (i.e., it must be manually "refreshed" to get the latest data), a Terminal Server solution (requires the Server to build multiple instances of the project to support each Thin Client), or offer a Thin Client solution similar to IWS but with expensive "add-on" software products.

Other Thin Client advantages include:

- Optional Secure Viewer that does not allow navigation outside the IWS project. Secure Viewer is not based on Microsoft's Internet Explorer.
- Ability to run VBScript and the Built-in Scripting Language on the Thin Client
- Can build Thin Client projects using Windows Embedded devices as the Server.
- Ability to support redundant Web Servers and Data Servers, with automatic switchover.

Thin Client Licensing

The maximum number of Thin Client stations connected simultaneously to the Server depends on the settings of the license installed on the Server. The user does not have to install any license on the Thin Client. For more information, see [License Settings](#).

Building a Simple Thin client program

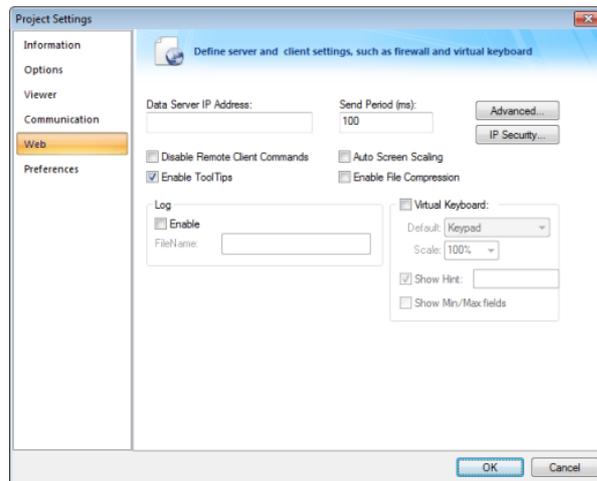
As we will see shortly, there is no one way to build a Thin client program. The following procedures allow you to develop a simple, unsecured Thin client program.

Procedure A: Thin Client using NTWebServer, Local Loopback

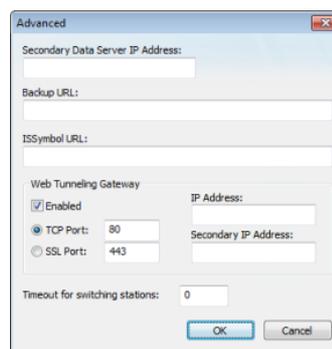
An easy way to initially develop and test a Thin client program is to use the light-weight Web server **NTWebServer** on your development PC, initiate the runtime project and use Microsoft Internet Explorer on the same PC to display the Web pages.

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 v7.0\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

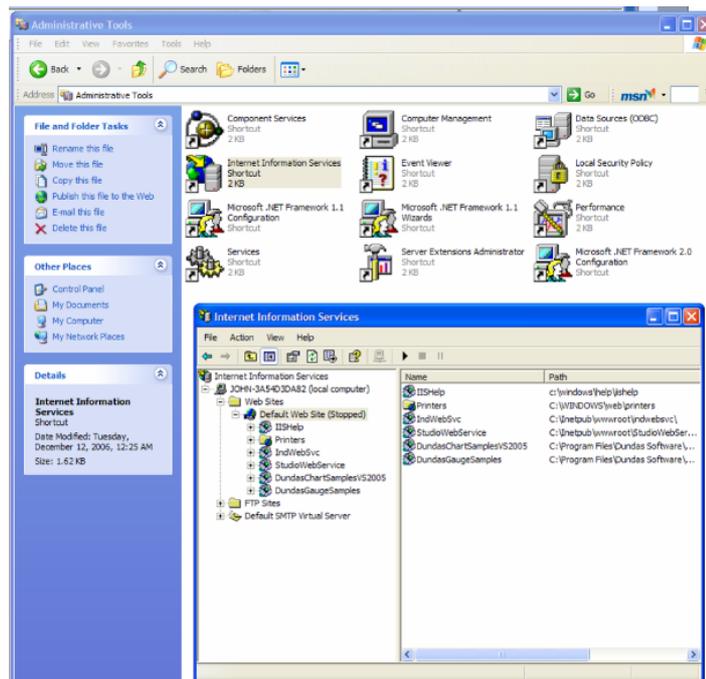
1. Click on the **Start** button (or Alt Tab) to access the Microsoft Internet Explorer program.
2. Start Internet Explorer, and type the address of the starting (home webpage). E.g. <http://192.168.1.100/startup.html>
3. Sign on as **Guest** with no password, assuming no security has been enabled.

Procedure C: Thin Client using IIS, Network IP

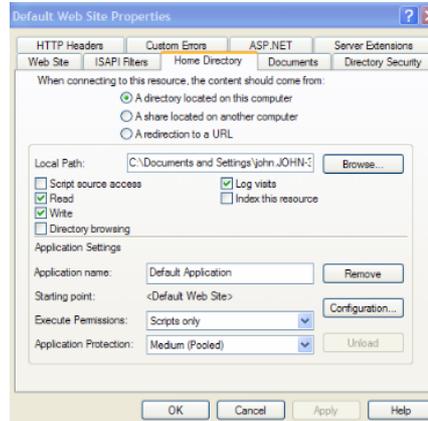
Once Procedure B has been completed, the next step is to enable IIS to become the Web Server for your Thin Clients instead of NTWebServer. The following are the basic steps to configuring IIS, although depending on your network and Operating System environment, additional settings may need to be configured such as user security.

Step 1: Configure IWS Settings

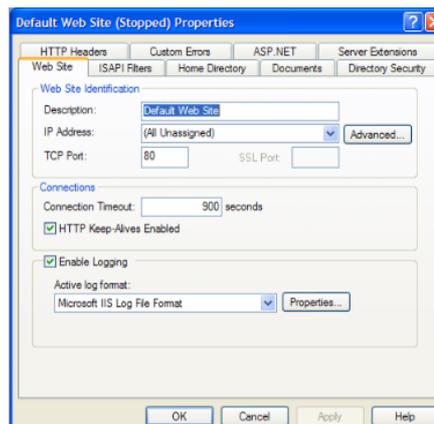
1. Be sure NTWebServer is terminated.
2. Click the Start button, then **Control Panel > Administrative Tools**.
3. Select (click on) **Internet Information Services**.
4. Expand the **Web Sites** tree structure to see the **Default Web Site**.



5. Right click on the Default Web Site and select **Properties** to get the Default Web Site Properties dialog.



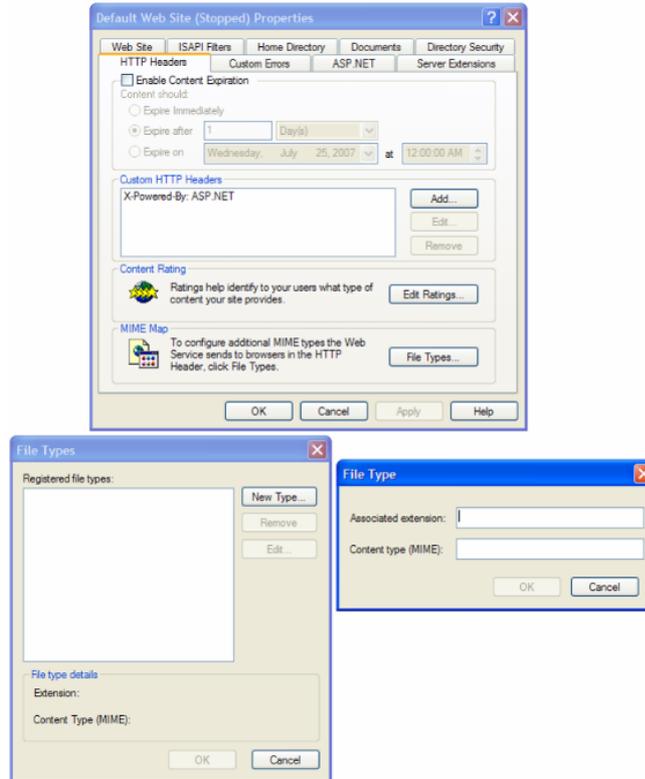
6. Click on the **Home Directory** tab. Click on the **Local Path Browse** button and point to your project's web subfolder.
7. Click on the **Web Site** tab. Make sure the TCP port is set to 80. You can click the **Advanced** button to enable the IIS Web Server to respond to specific IP addresses and 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.



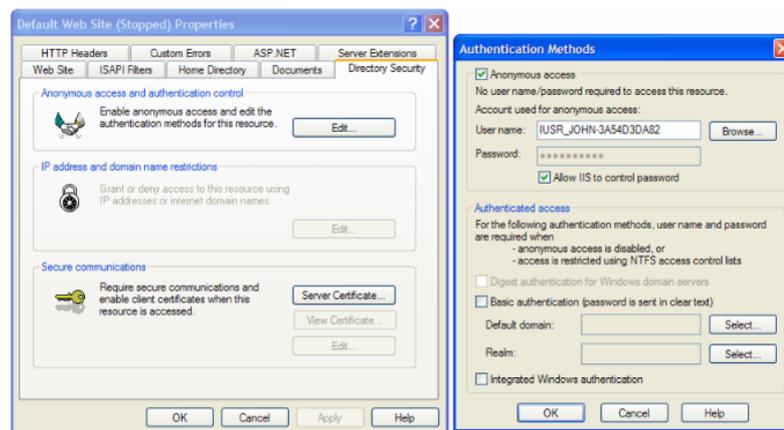
9. Click on the **HTTP Headers** tab. Click on the **MIME Map File Types** button to open the **File Types** dialog. Next, click on the **New Type** button to add a new MIME type. Put the file extension in the **Associated Extension** field. In the **Content type (MIME)** field, put MIME type followed 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.

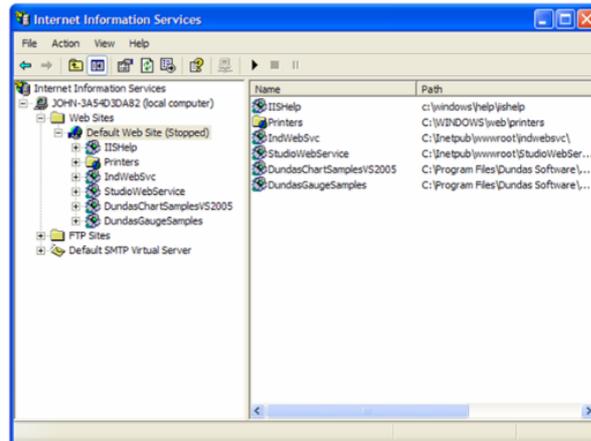
IIS is preconfigured to recognize a default set of global MIME types. These MIME types are recognized by all Web sites you create in IIS. MIME types can also be defined at the Web site and directory levels, independent of one another or the types defined globally. When you view MIME types at the Web 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
 - .sc
 - .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 you 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 v7.0\Bin\ISSymbolReg.exe
- [...] \InduSoft Web Studio v7.0\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:

```
[...] \InduSoft Web Studio v7.0\Redist\CE_version\processor_type\Bin\
```

For example, for a MIPS IV processor running Windows CE 5.0 or later:

```
[...] \InduSoft Web Studio v7.0\Redist\WinCE 5.0\MIPSIV\Bin\
```

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:

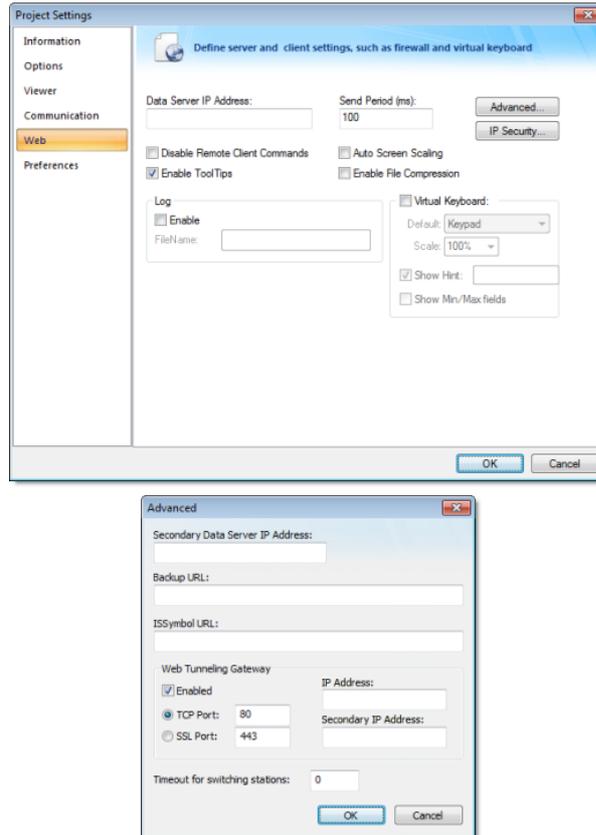
- **Web server** (HTTP Server): Provides the files from the server to the Thin Client via HTTP protocol over TCP/IP.
- **Data server** (TCP/IP Server module from IWS): Provides tag values and/or history data from the project running on the server to the Thin Client station(s).

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 dialogs:



The following table describes the meaning of the main Web settings illustrated in the above dialogs:

Setting	Description
Data Server IP Address	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.
Secondary Data Server IP Address	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.
Web Tunneling Gateway IP Address	The Thin Client Control (ISSymbol) uses the Web Tunneling Gateway IP Address to connect to the Web Tunneling Gateway.
Web Tunneling Gateway Secondary IP Address	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.

The Secondary addresses can be used in the following scenarios:

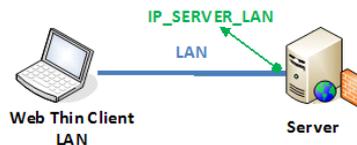
- When the Thin Clients can connect to either one of two redundant Servers (Web or Data); or

- 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:

Term	Description
LAN	Local Area Network (for example, Intranet)
WAN	Wide Area Network (for example, Internet)
Server	Station where the following components are running: <ul style="list-style-type: none"> • IWS (TCP/IP Server task) • Web Server (for example, Internet Information Services from Microsoft – IIS) • Web Tunneling Gateway for IIS (if enabled) Although IWS does not need to run in the same station where the other components are running, the following examples assume that it is.
Thin Client LAN	Thin Client station (Web Browser + ISSymbol control) that connects the Server via the LAN.
Thin Client WAN	Thin Client station (Web Browser + ISSymbol control) that connects the Server via the WAN.
IP_SERVER_LAN	IP Address of the Server on the LAN.
IP_SERVER_WAN	IP Address of the Server on the WAN.
IP_ROUTER_LAN	IP Address of the Router on the LAN.
IP_ROUTER_WAN	IP Address of the Router on the WAN.
ScreenName	Name of the project screen, saved as HTML, that is open on the Thin Client station.

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).

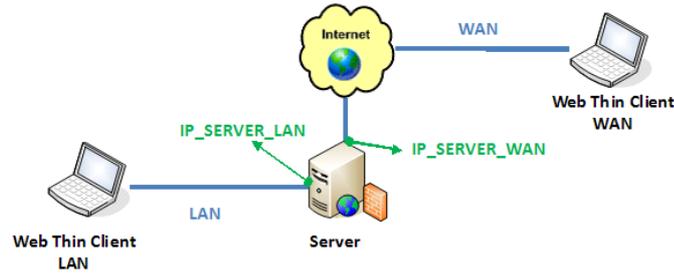
Configuration:

Setting	WTG Enabled	Web Gateway Disabled
Data Server IP Address	IP_SERVER_LAN	IP_SERVER_LAN
Secondary Data Server IP Address	-	-
Web Tunneling Gateway IP Address	IP_SERVER_LAN	-
Web Tunneling Gateway Secondary IP Address	-	-

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).

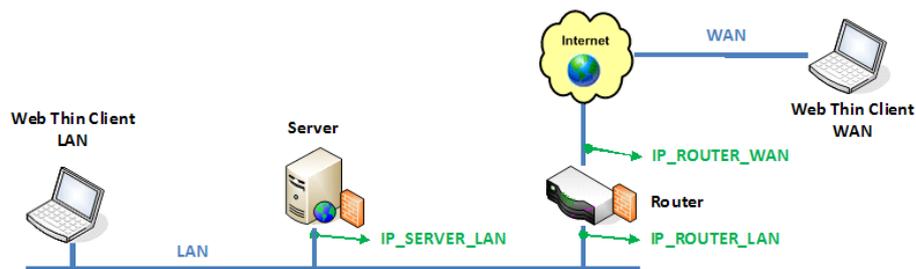
Configuration:

Setting	Web Gateway Enabled	Web Gateway Disabled
Data Server IP Address	IP_SERVER_LAN	IP_SERVER_LAN
Secondary Data Server IP Address	IP_SERVER_LAN	IP_SERVER_WAN
Web Tunneling Gateway IP Address	IP_SERVER_LAN	-
Web Tunneling Gateway Secondary IP Address	IP_SERVER_WAN	-

Note:

- URL From Thin Client LAN: http://IP_SERVER_LAN/ScreenName.html
- URL From Thin Client WAN: http://IP_SERVER_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:

Setting	Web Gateway Enabled	Web Gateway Disabled
Data Server IP Address	IP_SERVER_LAN	IP_SERVER_LAN
Secondary Data Server IP Address	IP_SERVER_LAN	IP_ROUTER_WAN

Setting	Web Gateway Enabled	Web Gateway Disabled
Web Tunneling Gateway IP Address	IP_SERVER_LAN	-
Web Tunneling Gateway Secondary IP Address	IP_ROUTER_WAN	-

 **Note:**

- URL From Thin Client LAN: **http://IP_SERVER_LAN/ScreenName.html**
- URL From Thin Client WAN: **http://IP_ROUTERR_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.

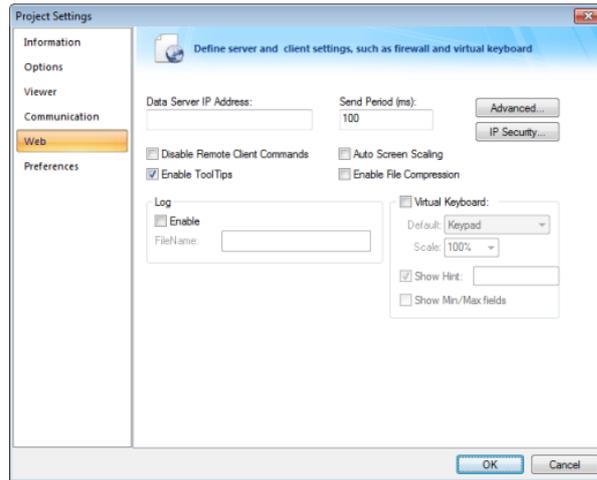
- Home directory of the Web Server (HTTP server) on the Server station: Web sub-folder of your project folder
- You must assign a Fixed IP address to the Router on the Internet (WAN), and the project must be running in this Server. Consult your ISP provider or IT department for further information about how to get a Fixed IP address for your Server.

Configuring the Data Server

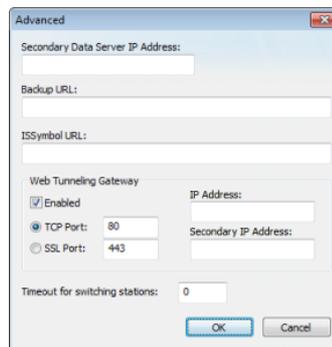
IWS has a couple dialogues that are used for configuration of the Data Server and the Web Server configuration to be used. This information is embedded in the HTML web pages that correspond the screens. The Data Server is part of the IWS runtime project and uses the TCP/IP Server module. There are three (3) basic steps to this configuration:

Step 1: Web Settings dialog configuration

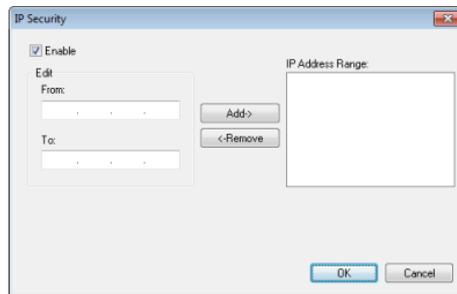
1. On the Project tab of the ribbon, in the Web group, click **Thin Client**.



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.

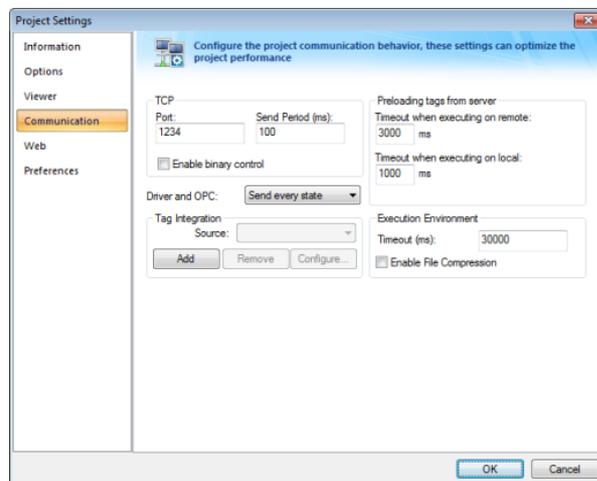


- By clicking on the **IP Security** button, you can define a range of IP addresses that will be accepted by the Data Server. If a request for data comes from an IP address outside of the defined ranges, the request will not be acknowledged. This is an Embedded Firewall function.



Step 2: Communication Settings dialog configuration

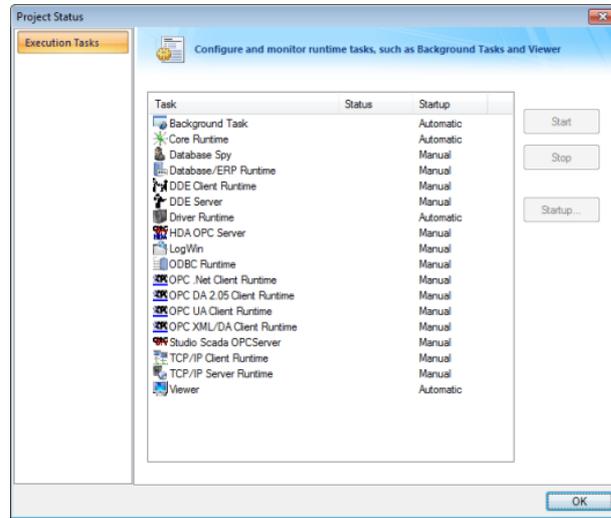
- On the Project tab of the ribbon, in the settings group, click **Communication**.



- 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).
- 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 Internet 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 IWS project over the Web, you must configure a Web server to host your project screens.

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](#) (SMA), the server must be able to process Collaborative Data Objects (CDO) and Active Server Pages (ASP); 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 v7.0 \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 Web site.

CEWebServer can be run on any [supported](#) Windows Embedded device, and it can be found in the IWS program directory at [...] \InduSoft Web Studio v7.0 \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 Web site.

 **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:

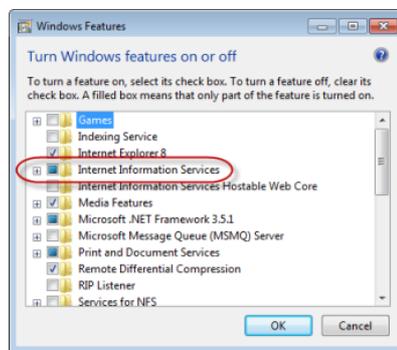
Versions of IIS

Version of Windows	Version of IIS	Notes
Windows XP Professional	IIS 5.1	Maximum of 10 simultaneous connections.
Windows Server 2003	IIS 6.0	No limit on connections.

Version of Windows	Version of IIS	Notes
Windows XP Professional x64 Edition		
Windows Server 2008 Windows Vista (most versions)	IIS 7.0	No limit on connections.
Windows Server 2008 R2 Windows 7 (all versions)	IIS 7.5	No limit on connections.
Windows CE and Windows Mobile 5.0 and later	IIS for CE .NET	Must be included in Platform Builder. Default is maximum of 10 simultaneous connections.

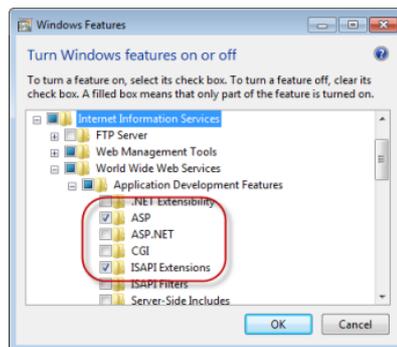
IIS supports all features of InduSoft Web Studio and it is robust enough to serve almost any IWS project in a production environment. It's the Web server that we recommend for most users. However, to properly install and configure it, you should be experienced with administering Windows on a network.

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**):



Turning on IIS in the Windows Features dialog

If you want to use SMA and/or WTG in your IWS project, then make sure ASP and ISAPI Extensions are also turned on:



Turning on ASP and ISAPI Extensions in the Windows Features dialog

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 (x86)\InduSoft Web Studio v7.0\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 be 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 v7.0\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:

```
regsvrce.exe "\ISSymbolPath\ISSymbolCE.ocx"
```

Example:

```
regsvrce.exe "\Storage Card\ISSymbolCE.ocx"
```

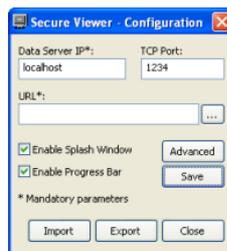
5. Save the registry settings to keep **ISSymbolCE.ocx** registered when you reboot the Windows Embedded device.

 **Tip:** Check the device manufacturer's documentation for how to save the registry settings.

6. Start the Secure Viewer by running **Viewer.exe**.

### 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.



 **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:



Secure Viewer Configuration Utility — Advanced Settings

- **Secondary Data Server IP** field: Type the IP address (or host name) of the secondary data server station. If the primary data server fails, the Secure Viewer will attempt to connect to the secondary data server automatically.
- **Backup URL** field: Type the URL of the backup location of the project file.
- **Date Format**: Adjust the date format on the Secure Viewer station to match the Data Server. This is to maintain compatibility with any Alarm/Event Control or Trend Control objects that you have in your project.
- **Timeout for switching stations** field: Enter the number of seconds that the Secure Viewer should wait before attempting to connect to the secondary Server, in the event that the primary Server fails. (For more information, see [Web tab](#) on page 85.)
- **Log on as Guest** option: Check (enable) to have the Secure Viewer automatically log on as Guest, eliminating the need to enter a Username or Password.
- **Disable Commands** option: Check (enable) to prevent the Secure Viewer from sending user commands back to the Server. It will only display current information from the Server.
- **Web** button: If you have configured a Web Tunneling Gateway to bridge your intranet to the Internet, then enter the addresses for the gateway here.

The second way to change the configuration of the Secure Viewer is to manually edit the **viewer.ini** file with a text editor. The structure of the file is as follows:

```
[Options]
nosplash= // Enable (0) or disable (1) the splash window
noprogressbar= // Enable (0) or disable (1) the progress bar
ds1= // 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) CEView 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
```

```
noprogressbar=1
dsl=192.168.1.106
nosplash=1
ds2=
dsp=1234
wtg1=
wtg2=
user=Guest
```

```
[OEM]
Splash=Splash.bmp //this is a splash screen
```

 **Note:**

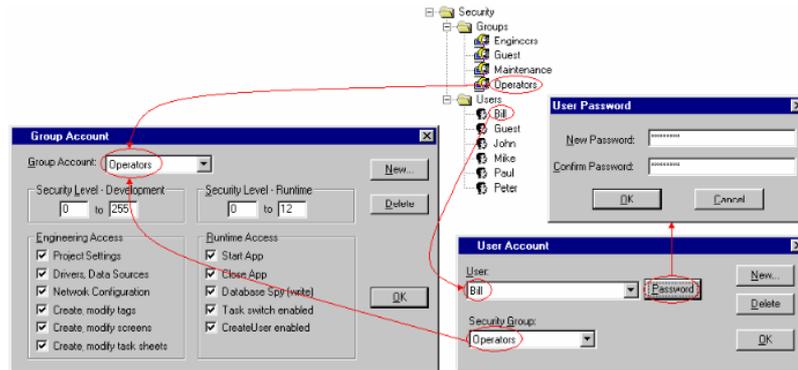
- Your InduSoft Web Studio runtime license must support a Secure Viewer (Thin Client) otherwise the connection to the Data Server will be refused.
- Be sure to put NTWebServer in your project folder (not web sub-folder) or point IIS to your project folder.

## 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.

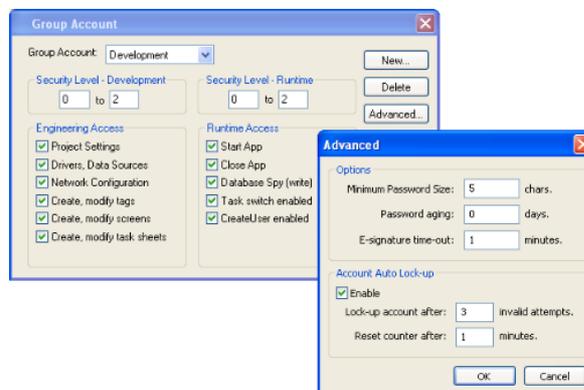
### Method 1: Password Protection

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.



Security Groups and Users

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).

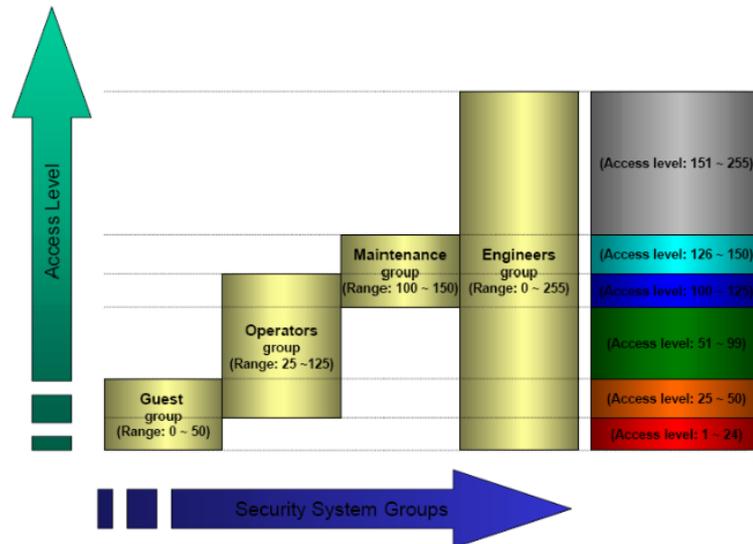


If System Security is enabled, these Password Protection features are also available at the Thin Client station. When a User at a Thin Client station attempts to connect to the Web Server, they will be prompted for a User Name and a Password. If either is invalid, the User will not be let on to the system.



Log On dialog

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.

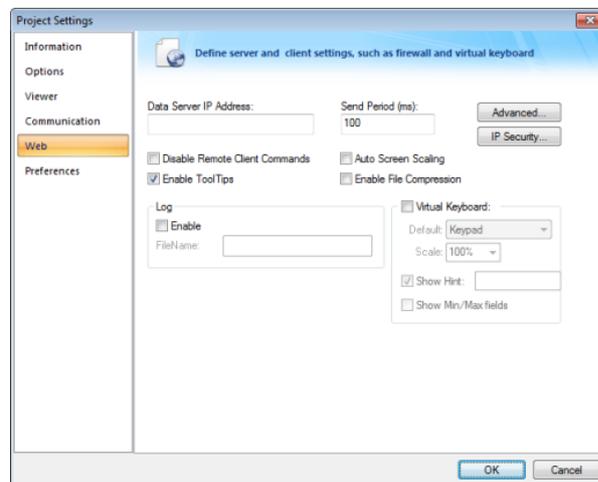


For more information, see [Security](#).

### 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):

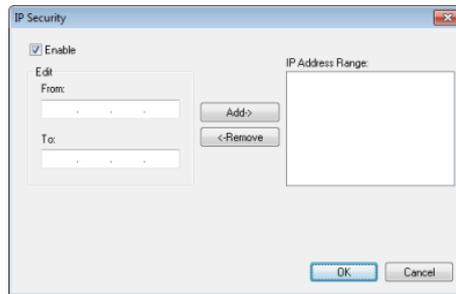


*Project Settings — Web tab*

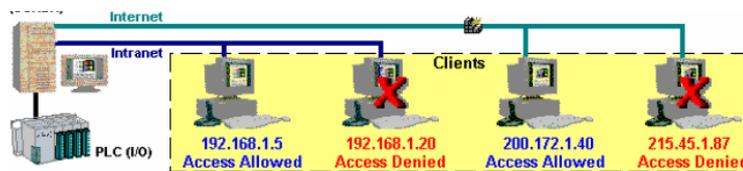
### Method 3: Embedded Firewall

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*

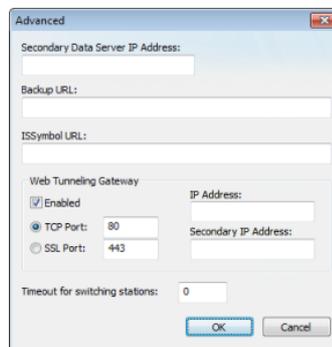


*Access allowed by IP address*

#### 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 than a simple public Internet connection.

## Port Usage

There are various ports that are used by IWS and/or related software. These are:

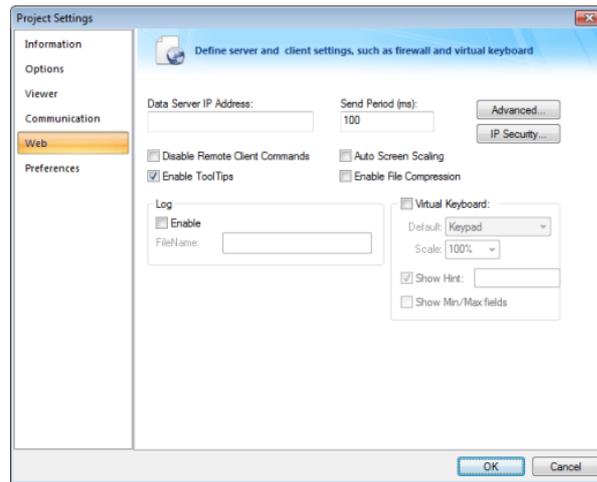
| Port # | Program                                      |
|--------|----------------------------------------------|
| 20     | FTP Server (Data)                            |
| 21     | FTP Server (Command)                         |
| 25     | SMTP Server                                  |
| 80     | Microsoft IIS Server for HTTP packets        |
| 110    | POP3                                         |
| 118    | Microsoft SQL Server Services                |
| 161    | SNMP                                         |
| 162    | SNMP Trap                                    |
| 389    | LDAP                                         |
| 443    | Microsoft IIS Server for HTTPS packets (SSL) |
| 502    | Modbus TCP/IP protocol                       |
| 663    | LDAP over SSL                                |
| 1028   | FTP Client (Command)                         |
| 1029   | FTP Client (Data)                            |
| 1234   | Project TCP/IP Server                        |
| 1443   | Microsoft SQL Server                         |
| 1444   | Microsoft SQL Server default port (Monitor)  |
| 1521   | Oracle                                       |
| 1526   | Oracle                                       |
| 2030   | Oracle                                       |
| 3001   | A-B Ethernet TCP/IP Protocol (default)       |
| 3306   | MySQL (can be configured to use 3306-3309)   |
| 3872   | Oracle Management Remote Agent               |
| 3997   | Studio ADO Gateway                           |
| 4322   | Remote Agent (CEServer)                      |
| 5432   | PostgreSQL                                   |
| 47808  | BACNet UDP Protocol (default)                |

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

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.

 **Note:** Microsoft provides Web Servers for any Microsoft operating system. Consult your Microsoft documentation about installing and configuring a Web Server.

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*

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.

## Downloading to a Remote Device

---

## Configuring the Target System

After configuring a project and testing it locally (on your development workstation), you can download the project to a remote runtime workstation that is running IWS on Windows XP/Vista/7 or running CEView on a Windows Embedded device.

1. Before you begin, verify that the Remote Agent (CEServer.exe) is running on the target (remote) workstation.
  - On a Windows platform, the **CEServer.exe** file is located in the [...]InduSoft Web Studio v7.0\Redist\CE Version\Processor Type\Bin folder.
  - On a Windows Embedded device, the file is located in the **\non-volatile** folder.
2. Run the **CEServer.exe** on the target workstation, and when the *Remote Agent* dialog is displayed, click **Setup**.



**Remote Agent dialog**

The *Setup* dialog opens:

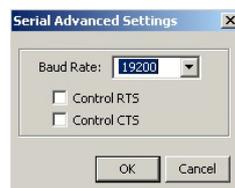


**Setup dialog**

Use the parameters on this dialog to configure communication between the development and target systems:

- **Serial Port:** To establish a serial connection to the project server, select Serial Port and then select the specific port from the drop-down combo box.

If you click **Advanced**, the *Serial Advanced Settings* dialog is displayed.



**Serial Advanced Settings dialog**

You can use the parameters on this dialog to control the flow of data between your target and development stations:

- **Baud Rate:** Select a baud rate from this drop-down combo box.
- **Control RTS:** Select (check) this to use a "Request to Send" control, where IWS sends an RS-232 signal from the transmitting station to the receiving station requesting permission to transmit.

- **Control CTS:** Select (check) this to use a "Clear to Send" control, where IWS sends an RS-232 signal from the receiving station to the transmitting station to indicate the receiving station is ready to accept data.

When you finish setting these parameters, click **OK** to close the *Serial Advanced Settings* dialog.

- **TCP/IP:** Enable this button to establish a TCP/IP connection to the development station.

 **Tip:** For better performance, we recommend using a TCP/IP connection instead of a Serial Link connection.

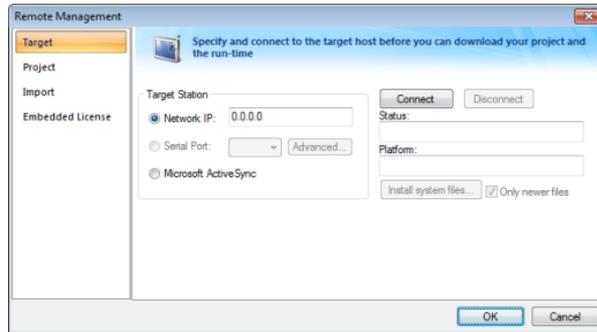
3. When you are finished, click **OK** to close the *Setup* dialog, but leave the Remote Agent program running in the remote workstation.

Continue to "[Configuring the Development Station.](#)"

## Configuring the Development Station

After configuring the target system to receive data, use the following steps to configure the development station to send the project data:

1. On the Home tab of the ribbon, in the Remote Management group, click **Connect**. The *Remote Management* dialog is displayed:



*Remote Management dialog*

2. Select the **Target** tab and use the following options to specify the target system to which you are going to connect.
  - **Network IP:** Select this button and type the IP address of the target system into the text box if you specified a TCP/IP connection when you configured the target system.
  - **Serial Port:** Select this button and select a port from the combo-box list if you specified a Serial Port connection when you configured the target system.
  - **Microsoft ActiveSync:** Select this button to connect to devices on which ActiveSync is enabled.
3. Click the **Connect** button to connect to the target system.  
If the target system is a Windows Embedded device, then you can click **Install System Files** to install the CEView runtime engine on the station.
4. Select the **Project** tab.
5. In the **Target** box, verify the directory for the project files on the target system. If the path is incorrect, then click the browse button to the right of the **Target** field to select a new directory. (Assuming you are properly connected to the target system, you should be able to browse it like any network drive.)

**Note:** You can download the project files only to folders that are below where the CEMServer is installed and running. For example, if CEMServer is installed at \Storage Card\CEView\, then the project files should be installed at \Storage Card\CEView\project\_name\.

6. Click **Send to Target** to download the entire project to the target system. (Or click **Send File** to select an individual file to send.)

**Caution:** When you send a project to the target system, the new project replaces the old one automatically and immediately, even if the old project is still running. Also, if you uncheck (disable) 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 send the new one. That way, you can make sure it stops gracefully and doesn't disrupt any other processes.

It's 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 **Send project to target** (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.

7. After the download is complete, click **Run** to start the project on the target system.

## Automatically Running a Project

By default, you must manually run your finished project on the Windows Embedded device, either from your PC by using the **Project** tab of the *Remote Management* dialog (see above) or on the device itself by clicking the **Start** button in the *Remote Agent* dialog.

However, you can configure the Windows Embedded device to automatically run a specified project. To do this, edit the file `CEServer.ini` on the Windows CE device to include the following setting:

```
[Setup]
AppName=Applicaion Path
```

Where `project_name` is the location of the IWS project files on the Windows Embedded device. For example:

```
[Setup]
AppName=\Harddisk\Test\CEserverTest
```

The next time the Windows Embedded 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 Windows Embedded 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 Windows Embedded device as a shared volume on your PC and edit the file there.
- Edit the file in the `[...]\InduSoft Web Studio v7.0\Redist` directory *before* you install the system files on the Windows Embedded 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 Windows Embedded devices; and
- You already know the location (file path) of the IWS project files on the device (perhaps by using the normal installation method on a test device).

## Database Interface

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

| Database                  | Database Version          | ADO.NET Provider          | Assembly Version |
|---------------------------|---------------------------|---------------------------|------------------|
| Microsoft SQL Server 2000 | 8.0                       | System.Data.SqlClient     | 1.0.5000.0       |
| Microsoft Access 2000     | 9.0.3821 SR-1             | System.Data.OleDbClient   | 1.0.5000.0       |
| Microsoft Excel 2000      | 9.0.3821 SR-1             | System.Data.OleDbClient   | 1.0.5000.0       |
| Oracle                    | 10g Release 1 for Windows | System.Data.OracleClient  | 1.0.5000.0       |
| Sybase                    | Anywhere 9.0.1.1751       | iAnywhere.Data.AsacClient | 9.0.1.1751       |
| MySQL                     | 4.0.20a                   | ByteFX.MySqlClient        | 0.7.6.15073      |

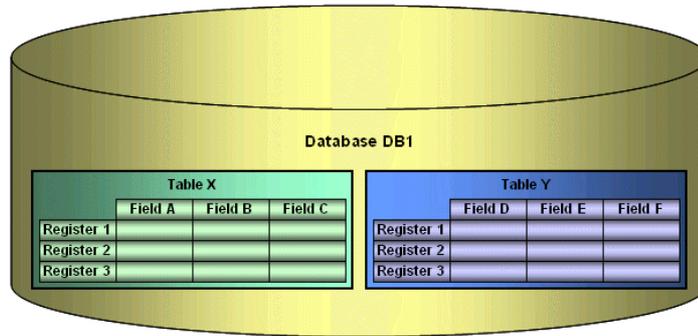
 **Note:** For information about how to configure a specific database, please refer to the following:

- [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

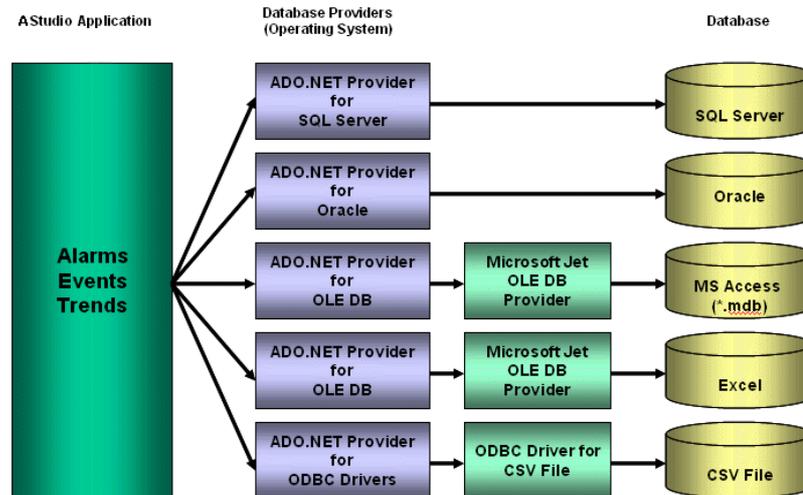
A SQL Relational Database is a set of information stored in tables with fields and registers, which support SQL commands.

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:



The project is running on the Server station (where InduSoft Web Studio 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 of the remote computer, and then execute the `StADOSvr.exe`.

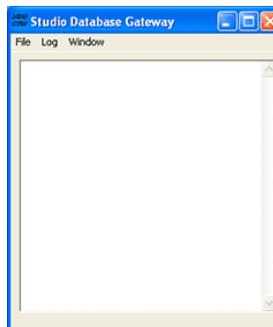
The gateway can be started multiple times for different TCP/IP port numbers. The default port number is 3997, and it is changed by specifying the desired port number in the command prompt (e.g., `StADOSvr 1111`). When running the `StADOSvr`, it will add the following icon to the system tray:



When you right-click on the icon, the following shortcut menu is displayed:



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*

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.



**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.

### 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:

| Section of .INI | Parameter             | Accepted Values                                    | Description                                                                                                                                                                                                                                      |
|-----------------|-----------------------|----------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Providers       | SaveMSec              | 1 : Disable 2 : Enable 3 : Separate Column         | 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.                                                                |
|                 | Assembly              | Any string that contains a .Net Framework assembly | 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.                                                                          |
|                 | ConnectionClass       | Any connection class inside the assembly           | The Connection Class is the one that implements the System.Data.IDbConnection interface.                                                                                                                                                         |
|                 | DateAdapterClass      | Any data adapter class inside the assembly         | 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.                                                      |
|                 | CommandBuilderClass   | Any command builder class inside the assembly      | The Command Builder class is also responsible for updates on databases. It must be compatible with the connection class.                                                                                                                         |
|                 | Provider              | Name of the provider                               | One of the parameters in the connection string is the "Provider". 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.                   |
|                 | ColumnDelimiterPrefix | Any character or group of characters               | Specify a character that will be placed before column names on SQL statements                                                                                                                                                                    |
|                 | ColumnDelimiterSuffix | Any character or group of characters               | Specify a character that will be placed after column names on SQL statements                                                                                                                                                                     |
|                 | TableDelimiterPrefix  | Any character or group of characters               | Specify a character that will be placed before table names on SQL statements                                                                                                                                                                     |
|                 | TableDelimiterSuffix  | Any character or group of characters               | Specify a character that will be placed after table names on SQL statements                                                                                                                                                                      |
|                 | ValueString           | Any string                                         | 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 (?)                                                                      |
|                 | ValueStringPrefix     | Any string                                         | 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. |

| Section of .IN | Parameter            | Accepted Values                                            | Description                                                                                                                                                                                                                                                                                                                 |
|----------------|----------------------|------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | ValueAddNumber       | 0 or 1                                                     | 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.                             |
|                | BoolType             | Any string representing a valid data type for the database | 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.                                                                                                                           |
|                | IntegerType          | Any string representing a valid data type for the database | 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.                                                                                                                      |
|                | RealType             | Any string representing a valid data type for the database | 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.                                                                                                                        |
|                | StringType           | Any string representing a valid data type for the database | 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.                                                                     |
|                | TimeStampType        | Any string representing a valid data type for the database | When trying to create columns to store TimeStamp values, the data type specified on this parameter will be used.                                                                                                                                                                                                            |
|                | EnableTop            | 0 or 1                                                     | When this field is set to 1, the ADO will place the TOP in the SQL statement to limit the amount of registers required.                                                                                                                                                                                                     |
|                | SingleConnection     | 0 or 1                                                     | 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. |
| Communication  | TimeOut              | 2                                                          | Time out to perform insert and update operations                                                                                                                                                                                                                                                                            |
|                | LongTimeOut          | 5                                                          | Time out to perform connection and query updates                                                                                                                                                                                                                                                                            |
|                | SyncTimeOut          | 60                                                         | Time out to perform synchronization                                                                                                                                                                                                                                                                                         |
| Connection     | RegBufSize           | 128                                                        | Size of the internal buffer used by the database API.                                                                                                                                                                                                                                                                       |
| Options        | DisableCloseQuestion | 0 or 1                                                     | 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.                                                                                                                                 |

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:

```
[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
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
SaveMSecl=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
CommandBuilderClass3=System.Data.Odbc.OdbcCommandBuilder
ValueString3=?
ValueAddNumber3=0
StringType3=VARCHAR(128)
EnableTop3=0
```

```
Provider4=ORAOLEDB
Assembly4=System.Data.OracleClient
ConnectionClass4=System.Data.OracleClient.OracleConnection
DataAdapterClass4=System.Data.OracleClient.OracleDataAdapter
CommandBuilderClass4=System.Data.OracleClient.OracleCommandBuilder
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
CommandBuilderClass5=iAnywhere.Data.AsaClient.AsaCommandBuilder
ValueString5=?
ValueAddNumber5=0
ColumnDelimiterPrefix5=[
ColumnDelimiterSuffix5=]
TableDelimiterPrefix5=[
TableDelimiterSuffix5=]
```

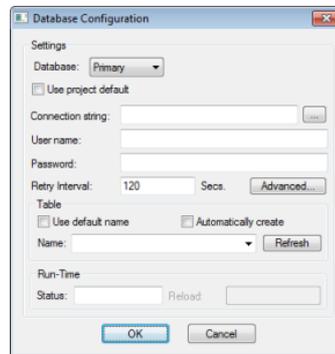
```
Provider6=MYSQLPROV
Assembly6=ByteFX.MySqlClient
ConnectionClass6=ByteFX.Data.MySqlClient.MySqlConnection
DataAdapterClass6=ByteFX.Data.MySqlClient.MySqlDataAdapter
CommandBuilderClass6=ByteFX.Data.MySqlClient.MySqlCommandBuilder
ValueString6=@Value
ValueAddNumber6=1
StringType6=VARCHAR(128)
EnableTop6=0
```

```
Provider7=MSDAORA
Assembly7=System.Data.OracleClient
ConnectionClass7=System.Data.OracleClient.OracleConnection
DataAdapterClass7=System.Data.OracleClient.OracleDataAdapter
CommandBuilderClass7=System.Data.OracleClient.OracleCommandBuilder ValueString7=Value
ValueAddNumber7=1
```

```
ValueStringPrefix7=:
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*

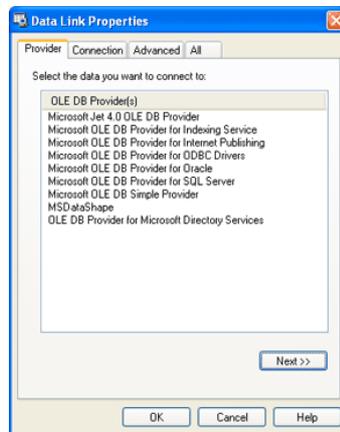
- **Database** combo-box: Allows you to select either Primary or Secondary. With Primary, all settings displayed in the Database Configuration window apply to the Primary Database interface. Otherwise, they apply to the Secondary Database interface. You can configure the Secondary database in the following 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

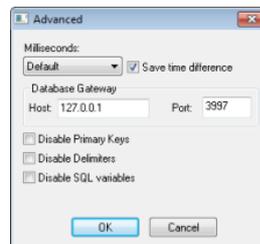
manually, you can press the browse button (...) and select the database type from the **Data Link Properties** window.



*Data Link Properties dialog*

**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*

- **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.
- **Save time difference** checkbox: When this option is checked (default), IWS saves the Time Zone configured in the computer where the project is running in each register of the database. This option must be enabled to avoid problems with daylight savings time.
- **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.

### 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:

| Value | Description                                                                                                                   |
|-------|-------------------------------------------------------------------------------------------------------------------------------|
| 0     | Disconnected from the database. The database is not available; your configuration is incorrect or it is an illegal operation. |
| 1     | The database is connected successfully.                                                                                       |
| 2     | The database is being synchronized.                                                                                           |

- **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:**

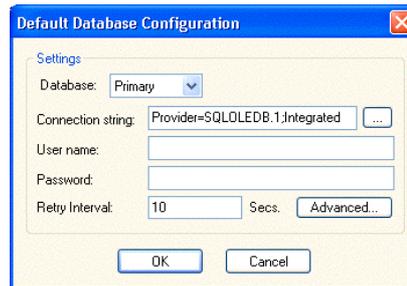
[Configuring a Default Database for All Task History.](#)

## 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.



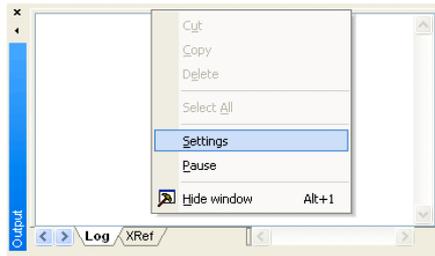
**Default Database Configuration dialog**

Please refer to [Database Configuration dialog](#) for help completing the fields in this window.

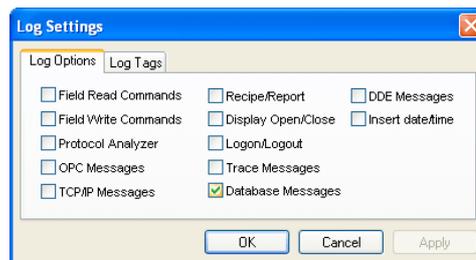
## Database Troubleshooting

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:



### enabling Database Messages

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 x.x 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 *AssemblyCache*?

**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 v7.0\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))
```

## ORACLE

**Q:** When I lose the connection with an ORACLE database, it does not recover. I receive the following message in the logwin: **Database: Error: ORA-03114: not connected to ORACLE**. Is that a problem with the IWS Database Interface?

**A:** The Oracle .Net Provider has a problem managing the connection pool. You need to install a QFE 830173. At the time that this document was written, more information about this problem could be found at [this site](#) (external link).

**Q:** When I try to access the database, I get the following error message: **ORA-00162: external dbid length 19 is greater than maximum (16)**. What should I do?

**A:** At the time that this document was written, there was a problem on the Oracle .NET Provider; the **Server Name** (SERVER/TNS) could not exceed 16 characters. In order to fix this problem, you should try to reduce your **Server Name** field. One way of doing this is to edit the file `\WINDOWS\system32\drivers\etc\hosts` to add an entry with a smaller server name. For instance, the server name specified by 192.168.89.98, has 13 characters, it could be reduced to 3 by adding the following line in the file:

```
192.168.89.98 ora
```

Now you can configure the **Server Name** configuration using ORA/TNS instead of 192.168.89.98/TNS.

## 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.

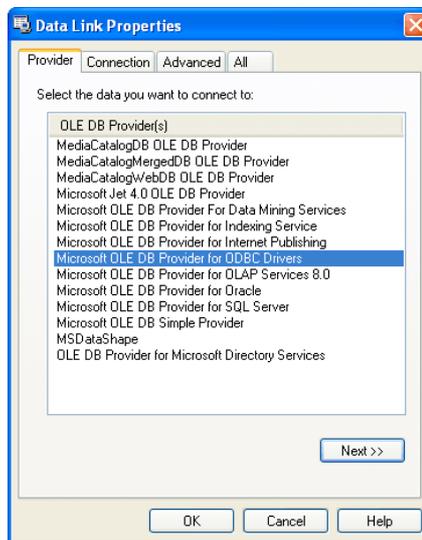
## Appendices

### 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.

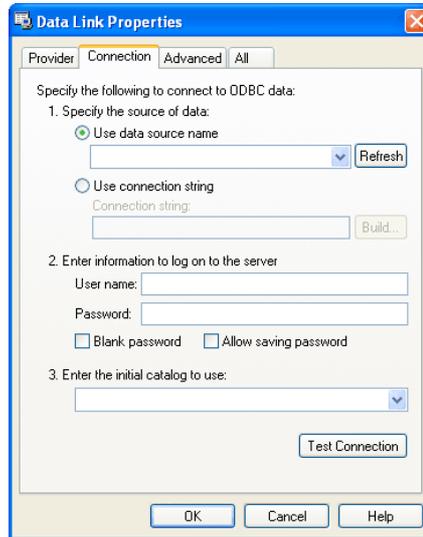
**Note:** Microsoft .NET Framework 2.0 is automatically installed with InduSoft Web Studio v6.1+SP5 and later.

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:



*Data Link Properties, Provider - ODBC*

When you click **Next**, the following window will display:



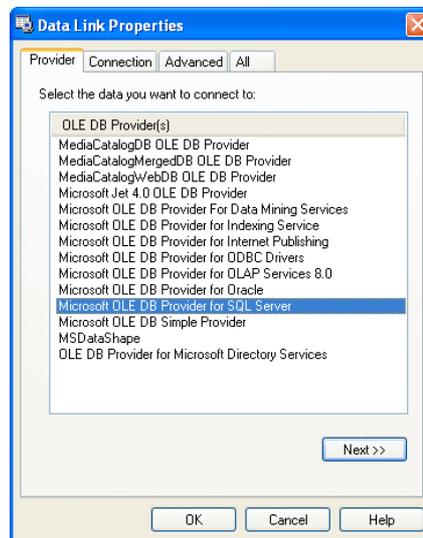
**Data Link Properties, Provider - Connection**

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

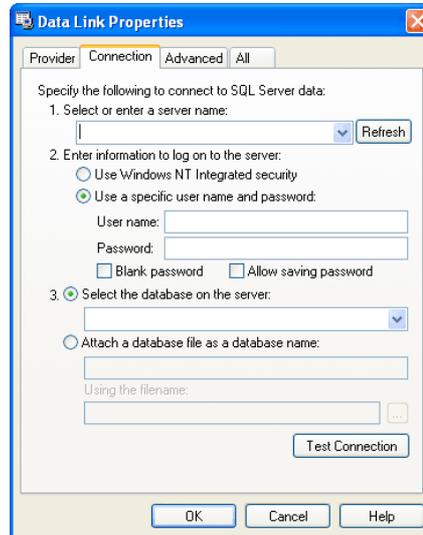
IWS Database Interface allows you to retrieve and store information on Microsoft SQL Server relational databases. You should follow the steps below in order to configure the SQL Server database:

1. Click on the **Browse** button in the [Database Configuration Dialog window](#). The following window will display:



**Data Link Properties, Provider - SQL Server**

2. Select the Microsoft OLE Provider for SQL Server and click **Next**. The following window will display:



**Data Link Properties, Connection - SQL Server**

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:

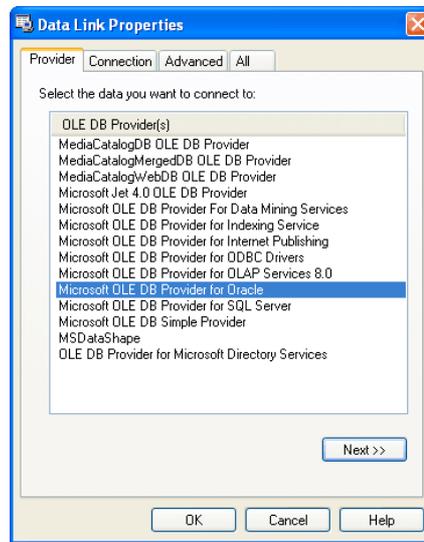
```
Provider=SQLOLEDB.1; Integrated Security=SSPI; Initial Catalog=MyDatabase; Data Source=192.168.23.200
```

 **Note:** These procedures were tested using Microsoft SQL Server 2000.

## Using ORACLE Databases

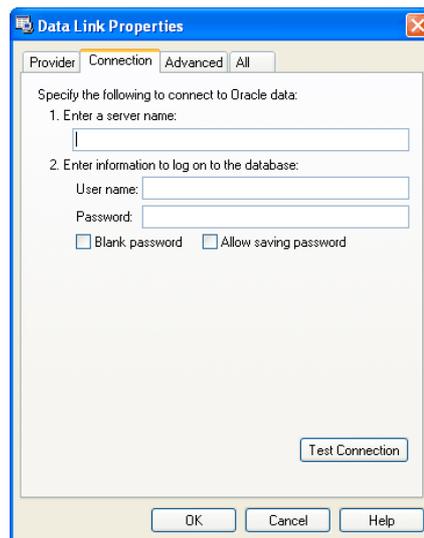
IWS Database Interface allows you to retrieve and store information on ORACLE relational databases. You should follow the steps below in order to configure the ORACLE database:

1. Click on the **Browse** button in the [Database Configuration Dialog window](#). The following window will display:



**Data Link Properties, Provider - Oracle**

2. Select the Microsoft OLE Provider for Oracle and click **Next**. The following window will display:



**Data Link Properties, Connection - Oracle**

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.

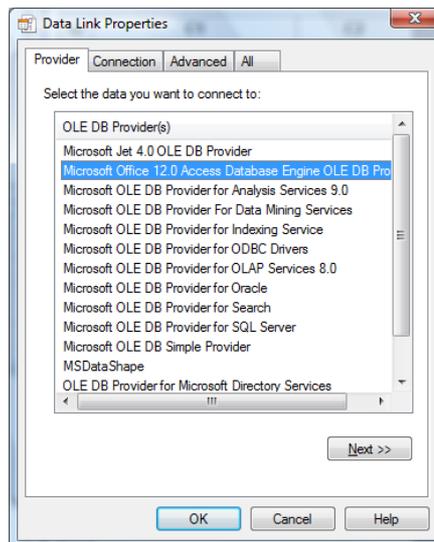
You do not need to have Access installed on the same computer as IWS; IWS can directly read from and write to an existing ACCDB file. However, you do need to use Access to create the initial file — either a blank database for new data or a populated database for reference data. Once you've created the 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 (...).

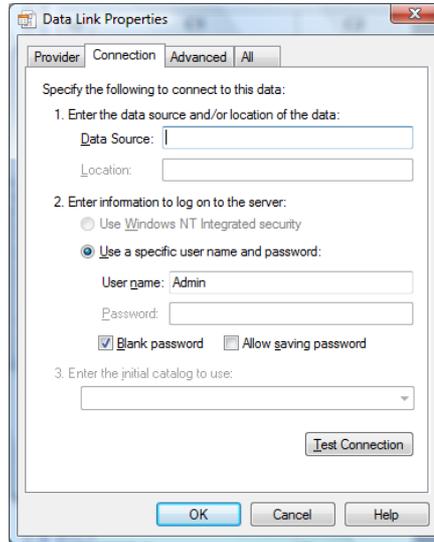
The *Data Link Properties* property sheet is displayed:



**Selecting the OLE DB Provider**

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:



**Specifying the location of the file**

- 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\My Documents\InduSoft Web Studio v7.0 Projects
\project_name\Database1.accdb
```

If you do this, however, then you must update the file path whenever you move the project folder.

- Specify the **User name** and **Password**, if any, for the ACCDB file.
- Click **Test Connection**.  
If a connection can be successfully established, then an appropriate message is displayed.
- 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 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:

| Version    | Connection String                         |
|------------|-------------------------------------------|
| SQL CE 2.0 | Provider=SQLCE; Data Source=database_path |

| Version    | Connection String                             |
|------------|-----------------------------------------------|
| SQL CE 3.0 | Provider=SQL_CE3.0; Data Source=database_path |
| SQL CE 3.5 | Provider=SQL_CE3.5; Data Source=database_path |

 **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`:

`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`:

`Provider=SQLCE; Data Source={DatabaseFile}`

The curly brackets (`{ }`) indicate that it's a tag reference.

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:

| IWS                                                                                         | SQL CE                |
|---------------------------------------------------------------------------------------------|-----------------------|
|  Boolean | <code>int</code>      |
|  Integer | <code>int</code>      |
|  Real    | <code>real</code>     |
|  String  | <code>nvarchar</code> |

## INSTALLING SQL SERVER ON A WINDOWS EMBEDDED DEVICE

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**.

1. On your desktop, download Microsoft SQL Server Compact Edition from the following website: <http://www.microsoft.com/sqlserver/en/us/editions/compact.aspx>

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:

```
wcload /noui /delete 0 "filepath\sqlce.dev.ENU.wce5.x86.CAB"
wcload /noui /delete 0 "filepath\sqlce.repl.wce5.x86.CAB"
wcload /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 Wcload 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.

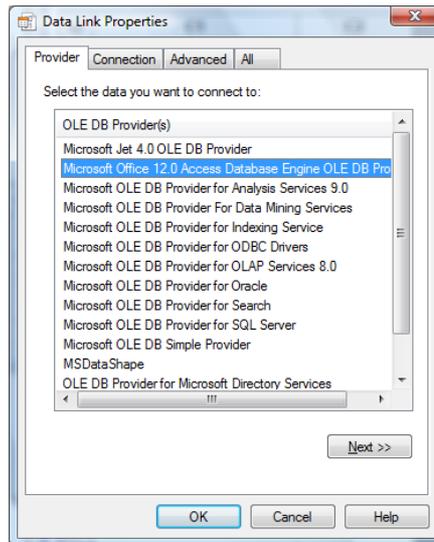
You do not need to have Excel installed on the same computer as IWS; IWS can directly read from and write to an existing XLS/XLSX file. However, you do need to use Excel to create the initial file — either a blank spreadsheet for new data or a populated spreadsheet for reference data. Once you've created 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 (...).

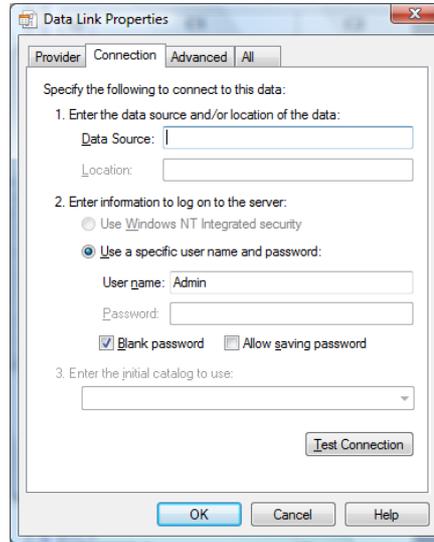
The *Data Link Properties* property sheet is displayed:



**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:



**Specifying the location of the file**

4. In the **Data Source** text box, type the complete file path for your XLS/XLSX file.

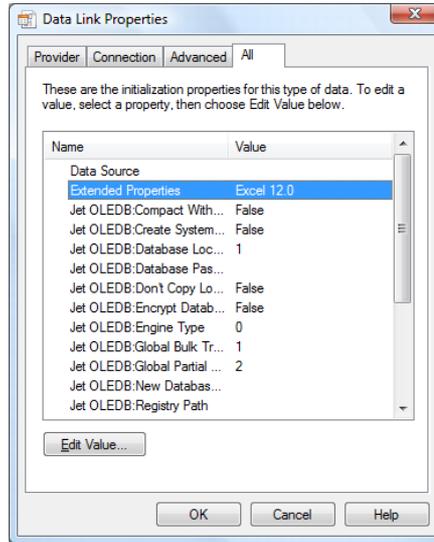
Although the file can be located anywhere on your computer or network, it may be useful to keep it in your project folder. For example:

```
C:\Users\username\My Documents\InduSoft Web Studio v7.0 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.

- Click **OK** to close the *Edit Property Value* dialog.



Editing the value of Extended Properties

- Click the **Connection** tab.
- Click **Test Connection**.

If a connection can be successfully established, then an appropriate message is displayed.

- 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=MYSQLCLIENT; 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

| Parameter    | Description                                                                                                                                            |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| Port=number; | 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. |

| Parameter                                                                                     | Description                                                                                                                                                                                                                   |
|-----------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                               | If a port of -1 is specified, then the connection will use the named pipes network protocol (see <code>Protocol</code> below).                                                                                                |
| <code>Server=myServerAddress1 &amp; myServerAddress2 &amp; ... &amp; myServerAddressN;</code> | Use any server in a replicated server configuration.                                                                                                                                                                          |
| <code>Encryption=true;</code>                                                                 | Enables SSL encryption for all data sent between the client and the server. The server must have a valid certificate installed.                                                                                               |
| <code>Encrypt=true;</code>                                                                    | An alternative to <code>Encryption</code> above, in case there are errors.                                                                                                                                                    |
| <code>Default Command Timeout=milliseconds;</code>                                            | Specifies a default command timeout for the connection. This does not supersede any timeout properties on individual commands.                                                                                                |
| <code>Connection Timeout=seconds;</code>                                                      | Specifies how long the client will wait for a server connection before terminating the attempt.                                                                                                                               |
| <code>Ignore Prepare=true;</code>                                                             | Instructs the database provider to ignore <b>Command.Prepare</b> statements, to prevent corruption from server-side prepared commands.                                                                                        |
| <code>Protocol=myProtocol;</code>                                                             | Specifies which network protocol to use. The default is <code>socket</code> or <code>tcp</code> , but you can specify <code>pipe</code> to use a named pipes connection or <code>memory</code> to use a shared memory object. |
| <code>Shared Memory Name=MySQL;</code>                                                        | Specifies the name of the shared memory object to be used for communication. (This parameter applies only if the <code>Protocol</code> parameter above is set to <code>memory</code> .)                                       |
| <code>CharSet=UTF8;</code>                                                                    | Specifies which character set to use to encode queries to the server.<br>Please note that query results are encoded in the same character set that the data itself is recorded.                                               |

 **Note:** These procedures were tested using MySQL v5.1.11 and MySQL Connector/Net v6.2.0.

Troubleshooting

---

## Troubleshooting

---

## General Troubleshooting

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 can not 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.

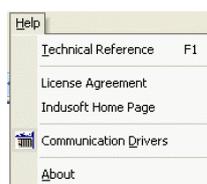
### Verifying Your Project

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 [...] \InduSoft Web Studio v7.0 \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**.)



*Help Menu*

### 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:

```
[...] \My Documents\InduSoft Web Studio v7.0 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, you may choose to email Customer Support at [support@indusoft.com](mailto:support@indusoft.com). Email is answered daily.

## Frequently Asked Questions

### Database & Security System

**Q.** What does the **Shared Tags** folder store?

**A.** 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.

**Q.** How do I count how many tags are configured in the project database?

**A.** The number of tags currently used in the project is displayed in the status bar at the bottom of the development environment. Each array position and each class member of the tags configured in the IWS tag database are counted.

**Q.** How do I see the list of "Users" I've added during runtime in my project that I have created with the `CreateUser()` function?

**A.** Execute the following command: "`Studio Path\BIN\Studio Manager.exe`" "`Studio Path\BIN\ExtUser.dll`" (for example: "`E:\Program Files\Studio\BIN\Studio Manager.exe`" "`E:\Program Files\Studio\BIN\ExtUser.dll`"). This command will launch a dialog. You can see the users created by the `CreateUser()` function, and you can then create or delete users.

### Graphics

**Q.** How do I insert and configure an ActiveX object in a project?

**A.** 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).

**Q.** How do I designate one screen that will open each time I start the project?

**A.** 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.

**Q.** How do I insert a background picture on the screen?

**A.** 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.

### Tasks

**Q.** How do I convert the History Trend to an ASCII file?

**A.** To convert a History Trend file to an ASCII format, copy the file "**StudioPath\bin\hst2txt.exe**" to the path "**\project\_name\hst\**". Alternatively, you can use the **HST2TXT** function in a *Math* worksheet to convert binary files into text format automatically without having to use a DOS window.

**Q.** How do I exchange data with FOX Pro by an ODBC protocol?

**A.** When exchanging data with FOX Pro database, it's necessary to set the parameter **UseQuote=0** from the [ODBC] section in the **project\_name.app** file.

**Q.** How do I set a **DATE** field for an ODBC interface with an Oracle package?

**A.** Configure the "Column" cells in the ODBC worksheet with the syntax **ColumnName.ts** (for example: **MyDate.ts**).

**Q.** How do I execute a *Math* worksheet during the startup and another *Math* worksheet during the project shutdown?

**A.**

- **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

**Q.** How do I set a "communication error" alarm?

**A.** 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).

**Q.** How do I communicate with a Siemens S7-200 PLC without using Prosave software?

**A.** Siemens S7-200 PLC has a Freeport that can implement any protocol via PLC programming. There is PLC free software distributed by Siemens that implements Modbus protocol in the PLC Freeport (for further details contact Siemens support). Using this software in the PLC with IWS's Modbus driver (MODBU), you can exchange information between them.

**Q.** How do I start and stop communication drivers during runtime?

**A.** There are three functions available to handle the execution of the communication drivers during runtime:

- Start all drivers configured in the project: **StartTask("Driver")**  
For example: **StartTask("Driver")**
- Start a specific driver configured in the project:

```
WinExec("StudioPath\bin\Studio Manager.exe" + " " + "StudioPath\bin\Driver.dll" + " " + "DriverName")
```

For example:

```
WinExec(Asc2Str(34) + "Bin\Studio Manager.exe" + Asc2Str(34) + " " + Asc2Str(34) + "Bin\Driver.dll" + Asc2Str(34) + " " + Asc2Str(34) + "MODBU" + Asc2Str(34))
```

 **Note:** The **Asc2Str(34)** function is used to concatenate quotation marks for paths where there are space chars.

- Stop a specific driver configured in the project: **EndTask("DriverDriverName")**  
For example: **EndTask("DriverMODBU")**

 **Tip:** You can start or stop other tasks using the `StartTask(TaskName)` and `EndTask(TaskName)` functions. For example, `StartTaks("Viewer")`, `Endtask("Viewer")`.

**Q.** What are the parameters of the IWS DDE Server?

**A.** The IWS DDE Server and NetDDE Server parameters are shown in the table below:

| Comm. Type  | Application              | Topic     | Item      |
|-------------|--------------------------|-----------|-----------|
| Network DDE | \\<Computer Name>\NDDE\$ | UNISOFT\$ | <TagName> |
| Local DDE   | UNIDDE                   | DB        | <TagName> |
| Network DDE | \\<Computer Name>\NDDE\$ | UNISOFT\$ | <TagName> |
| Local DDE   | UNIDDE                   | DB        | <TagName> |

**Q.** How do I exchange data with Excel by using NetDDE?

**A.** 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:

```
='\computer name\NDDE$' | 'UNISOFT$' !tag name
```

For example:

```
='\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`).

**Q.** Is the IWS OPC interface compliant with OPC specification v1.0a or v2.0?

**A.** The IWS OPC Client and OPC Server modules are compliant with both OPC specification v1.0a and v2.0.

**Q.** How do I get errors from Intellution / GE Fanuc iFIX applications?

**A.** 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

**Q.** What operating systems are compatible with InduSoft Web Studio and CEView?

**A.** See table below. "N" means that the operating system is NOT supported, and "Y" means that the operating system is supported.

| Operating System |         | InduSoft Web Studio |                      |                      |                   | CEView |      |      |      |
|------------------|---------|---------------------|----------------------|----------------------|-------------------|--------|------|------|------|
| Name             | Version | v2.x                | v3.x<br>thru<br>v4.1 | v4.2<br>thru<br>v6.0 | v6.1 or<br>higher | v3.x   | v4.x | v5.x | v6.x |
| Windows Vista    | Any     | N                   | N                    | N                    | Y                 | N      | N    | N    | N    |
| Windows XP       | Any     | N                   | Y                    | Y                    | Y                 | N      | N    | N    | N    |

| Operating System |                    | InduSoft Web Studio |                |                |                | CEView |      |      |      |
|------------------|--------------------|---------------------|----------------|----------------|----------------|--------|------|------|------|
| Name             | Version            | v2.x                | v3.x thru v4.1 | v4.2 thru v6.0 | v6.1 or higher | v3.x   | v4.x | v5.x | v6.x |
| Windows 2000     | Any                | N                   | Y              | Y              | Y              | N      | N    | N    | N    |
| Windows NT       | v4.0+SP4 or higher | Y                   | Y              | Y              | N              | N      | N    | N    | N    |
| Windows ME       | Any                | N                   | Y              | N              | N              | N      | N    | N    | N    |
| Windows 98       | Any                | N                   | Y              | N              | N              | N      | N    | N    | N    |
| Windows 95       | Any                | Y                   | Y              | N              | N              | N      | N    | N    | N    |
| Windows CE       | v2.12              | N                   | N              | N              | N              | Y      | N    | N    | N    |
|                  | v3.x               | N                   | N              | N              | N              | Y      | Y    | Y    | Y    |
| Windows CE.Net   | v4.0               | N                   | N              | N              | N              | N      | N    | Y    | Y    |
|                  | v4.1               | N                   | N              | N              | N              | N      | N    | Y    | Y    |
|                  | v4.2               | N                   | N              | N              | N              | N      | N    | Y    | Y    |
|                  | v5.0               | N                   | N              | N              | N              | N      | N    | N    | Y    |
|                  | v6.0               | N                   | N              | N              | N              | N      | N    | N    | Y    |

**Q.** How do I start IWS automatically when the computer is powered on?

**A.** Create a shortcut to [...] \InduSoft Web Studio v7.0 \Bin \RunStudio.exe and then place it in your Startup folder (C:\WINDOWS\Documents and Settings\All Users\Start Menu\Programs\Startup\).

**Q.** How do I disable Dr. Watson?

**A.** The step-by-step procedure to disable Dr. Watson under Windows NT is described below:

1. Execute the program *WindowsPath\RegEdit.exe* (for example, C:\WinNT\Regedit.exe)
2. Select the path `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AeDebug`
3. Set the value 0 (zero) to the parameter "Auto" from the path selected.
4. Close the *Registry Editor* window.



**Caution:** Special cautions must be taken when editing parameters in the *Registry Editor* program because some of them can modify the overall behavior of the operating system.

There are two ways to modify the data format in IWS:

- **Off-Line:** Set the parameters `Order=<DateFormat>` (for example, `DMY`) and `Separator=Separator` (for example, `/` or `.`) from the [International] section from the [...] \InduSoft Web Studio v7.0 \Bin \Program Settings.ini file.



**Note:** You will need to verify your project to apply date settings to previously saved Web pages.

- **On-Line:** Use the function `SetDateFormat( strSeparator, strDateFormat )`. For example, `SetDateFormat( "/", "MDY" )`.

**Q.** What features are not supported by CEView?

**A.** The main features not supported by CEView are: DDE, NetDDE, ODBC, PasteLink, and a number of functions.

**Q.** How do I show a splashwindow when starting a project in CEView?

**A.** 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
```

**Q.** How do I see runtime messages and errors on a "blind" Windows Embedded device — that is, a device that has no display?

**A.** 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
```

**Q.** What are the main steps to create a Web-based application?

**A.** 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 (**Save** in the Application menu), save the screens that must be available for the Thin Client in HTML format (**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.

**Q.** How do I maintain communication between a Thin Client connecting via proxy and a Web Gateway application running on Microsoft IIS?

**A.** 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.

**Q.** How do I send an email from the IWS project?

**A.** 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:

| Characteristic          | SendEmail   | SendEmailExt |
|-------------------------|-------------|--------------|
| Execution               | Synchronous | Asynchronous |
| Supports Subject text   | Y           | Y            |
| Supports Message text   | Y           | Y            |
| Supports TO addresses   | Y           | Y            |
| Supports CC addresses   | N           | Y            |
| Supports BCC addresses  | N           | Y            |
| Supports attached files | N           | Y            |

**Q.** The runtime task (TCP/IP, OPC, DDE, ODBC, etc) does not work.

**A.** 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**.

**Q.** The browser from the Thin Client does not display the screen and launches a warning message regarding ISSymbol.ocx.

**A.** 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**.

**Q.** The browser of the Thin Client launches an error message missing the `ISSymbol.ocx` and does not display the screens from the Server.

**A.** `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:** Use the command `regsvr32 ISSymbol32.ocx` to register the ActiveX component in the Thin Client.

**Q.** The screens are shown on the Thin Client (Browser); however, the data (tags values) are not read from the Server.

**A.** 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:** 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.

**Q.** The "On Up" expressions configured in the Command animation are not executed.

**A.** 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.

**Q.** The Trend History does not work after adding or removing tags in the *Trend* worksheet.

**A.** 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.

**Q.** The value of indirect tags ( `@TagName` ) is not shown in the Thin client program.

**A.** When a screen is saved as HTML, IWS saves a `ScreenName.tag1` file in the `\WEB` subfolder. This file has the list of all tags configured in the screen (objects and animations). When a screen is opened in the Thin Client browser, the tags listed in the `ScreenName.tag1` are "enabled" for TCP/IP communication

with the server station. It provides an optimized communication between the server station and the Thin Client stations.

When using indirect tags in this way (*@IndirectTag*), the tags pointed will not exchange data with the Server, unless they had been configured in the screen. In other words, the tags that will be pointed in the screen MUST be configured in any object of the screen to enable the TCP/IP communication for these tags with the server station.

 **Tip:** Add a transparent rectangle (no fill and no line) in the screen corner. Apply the Command animation to this rectangle and configure the tags (which can be pointed by indirect tags during runtime in the Thin Client station) in the **Expression** fields (keep the **Tag Name** fields blank). These tags will be added to the **ScreenName.TAGL** file, and they will be available for TCP/IP communication with the Server station.

**Q.** Which functionalities are not supported by PocketPC platforms (for example, IPaq, Cassiopeia, Jornada)?

**A.** Windows CE devices powered PocketPC do not support some functionality which are supported by Windows CE devices powered by the "standard" Windows CE version:

#### Functionality not supported by PocketPC devices

DCOM (Distributed Component Object Model): It means that all features based on DCOM (for example, remote OPC communication) are not supported by PocketPC devices.

[DialGetClientIP\(\)](#) function does not work for PocketPC devices

**Q.** How do I enable the "Hibernate" options from the operating system after installing IWS on a notebook?

**A.** Follow the procedure below:

1. Run the *Registry Editor* ( <Start button>\Run `egedit` ).
2. Select the following path from the *Registry Editor*: **HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Services\ Proteq\Parameters**
3. The **IoPortAddress** parameter from the path mentioned above is set with the hexadecimal value, **0x00000111**. Set this parameter with the hexadecimal address of the LPT1 parallel port of your notebook (for example, **0x00000378** ).
4. Close the *Registry Editor* and reboot the computer.

 **Tip:** The Hexadecimal address of the LPT1 parallel port of the notebook can be gotten from the Control Panel ( **System\Hardware\Device Manager\Ports (COM & LPT)\Printer Port (LPT1)\Properties\Resources** ). Pick the initial address of the **I/O Range**. Usually it is the hexadecimal address **0x00000378**.

## 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.

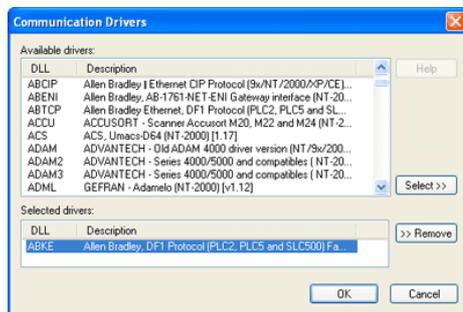
### Technical Reference

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.

### Communication Drivers

To see the available documentation for the communication drivers, click **Communication Drivers** on the Help tab of the ribbon.



*Communication Drivers dialog*

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

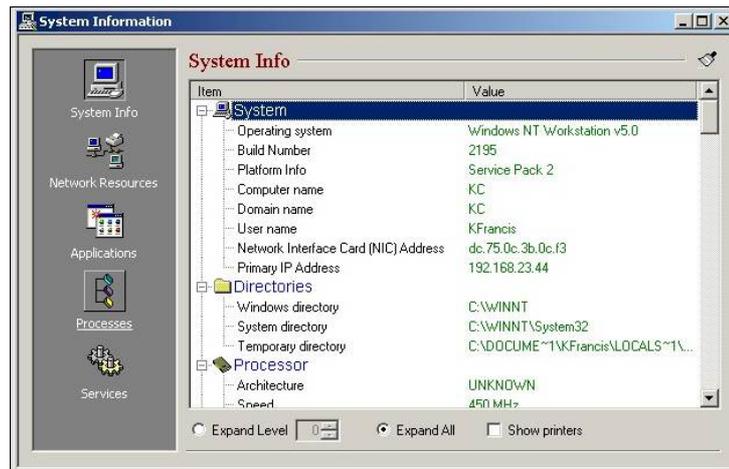
To view the release notes for InduSoft Web Studio, click **Release Notes** on the Help tab of the ribbon.

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).



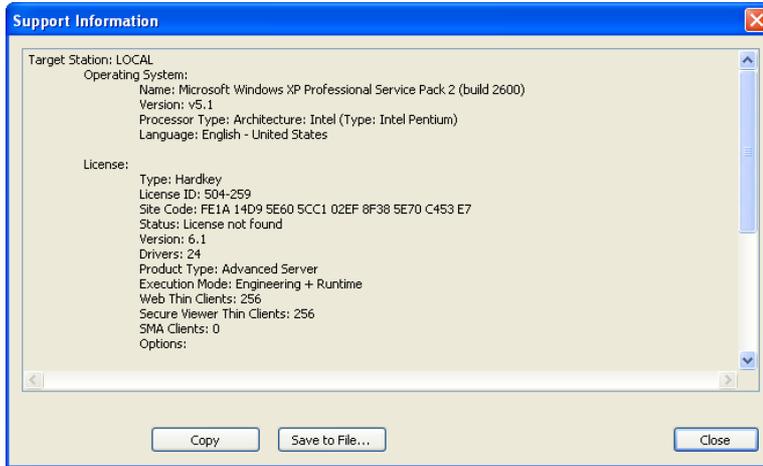
**System Information Window**

**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*

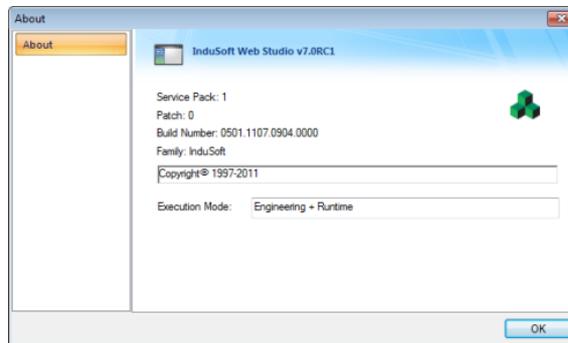
To copy the information to the Clipboard, click **Copy**. You can then paste the information into another window or text field, such as the body of an email message.

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*

## Appendix: Built-in Scripting Language

---

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

| Operator | Usage   | Description                                                                           |
|----------|---------|---------------------------------------------------------------------------------------|
| AND      | A AND B | TRUE if A and B are both TRUE                                                         |
| OR       | A OR B  | TRUE if A is TRUE, or B is TRUE, or both                                              |
| XOR      | A XOR B | TRUE if A is TRUE, or B is TRUE, but not both                                         |
| NOT      | NOT A   | TRUE if A is FALSE                                                                    |
| =        | X = Y   | TRUE if X is equal to Y                                                               |
| >        | X > Y   | TRUE if X is greater than Y                                                           |
| >=       | X >= Y  | TRUE if X is greater than or equal to Y                                               |
| <        | X < Y   | TRUE if X is less than Y                                                              |
| <=       | X <= Y  | TRUE if X is less than or equal to Y                                                  |
| <>       | X <> Y  | TRUE if X is not equal to Y                                                           |
| &        | X & Y   | Bitwise AND:<br><br>0101 (decimal 5)<br>AND 0011 (decimal 3)<br>= 0001 (decimal 1)    |
|          | X   Y   | Bitwise OR:<br><br>0101 (decimal 5)<br>OR 0011 (decimal 3)<br>= 0111 (decimal 7)      |
| ^        | X ^ Y   | Bitwise XOR:<br><br>0101 (decimal 5)<br>XOR 0011 (decimal 3)<br>= 0110 (decimal 6)    |
| ~        | ~ X     | Bitwise NOT:<br><br>NOT 0101 (decimal 5)<br>= 1010 (decimal 10)                       |
| >> n     | X >> Y  | Rotate n bits to right:<br><br>0110 (decimal 6)<br>ROTATE RIGHT<br>= 0011 (decimal 3) |
| << n     | X << Y  | Rotate n bits to left:<br><br>0110 (decimal 6)<br>ROTATE LEFT<br>= 1100 (decimal 12)  |

 **Tip:** For more complex logic, try the [Logical](#) and [Loop](#) functions.

### Arithmetic operators

| Operator | Usage | Description |
|----------|-------|-------------|
| +        | X + Y | Add (plus)  |

| Operator | Usage   | Description      |
|----------|---------|------------------|
| -        | $X - Y$ | Subtract (minus) |
| *        | $X * Y$ | Multiply by      |
| /        | $X / Y$ | Divide by        |

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:

$5+2*3$

In contrast, if you use parentheses to change the syntax, 5 and 2 are added together and then multiplied by 3 to produce 21:

$(5+2)*3$



**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:

| Function             | Group             | Execution                          | Windows                           | Embedded                          | Thin Client                       |
|----------------------|-------------------|------------------------------------|-----------------------------------|-----------------------------------|-----------------------------------|
| <i>function name</i> | <i>group name</i> | <i>synchronous or asynchronous</i> | <i>supported or not supported</i> | <i>supported or not supported</i> | <i>supported or not supported</i> |

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 means that 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, then your project may suffer decreased performance.

- Asynchronous means that 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 includes Server and Client stations running on a full desktop or server version of Microsoft Windows.
- Embedded includes Server and Client stations running on some version of Windows Embedded.
- Thin Client includes Client stations running the Secure Viewer program or in a Web browser.

For more information about these system types, see [System Requirements](#).

### 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` means the parameter can take either a literal Boolean value *or* the name of a Boolean tag. For example, either `0` or `boolTag`.
- `num` means the parameter can take either a literal numerical value *or* the name of an Integer or Real tag. For example, either `45.6543` or `numTag`.
- `str` means 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 `strTag`.

The additional prefix `opt` indicates that a parameter is optional. If you do not specify a value for the parameter, then 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.

| Function | Group       | Execution   | Windows   | Embedded  | Thin Client   |
|----------|-------------|-------------|-----------|-----------|---------------|
| Trace    | Log Message | Synchronous | Supported | Supported | Not Supported |

### Syntax

`Trace (strOutputMessage)`

#### **strOutputMessage**

The text of the message to be displayed.

### 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.

| Function | Group      | Execution   | Windows   | Embedded  | Thin Client |
|----------|------------|-------------|-----------|-----------|-------------|
| Abs      | Arithmetic | Synchronous | Supported | Supported | Supported   |

### Syntax

Abs ( *numValue* )

#### **numValue**

The numeric value from which the function takes the absolute value.

### Returned value

The absolute value of the specified numeric value.

### Examples

| Tag Name | Expression                                                                                                  |
|----------|-------------------------------------------------------------------------------------------------------------|
| Tag      | <b>ABS</b> ( "-54.9788" ) // Returned value = 54.9788                                                       |
| Tag      | <b>ABS</b> ( <b>numValue</b> ) // Returned value = absolute value of the number in the <b>numValue</b> tag. |

### Div

Div is a built-in scripting function that gets the dividend of two specified numeric values.

| Function | Group      | Execution   | Windows   | Embedded  | Thin Client |
|----------|------------|-------------|-----------|-----------|-------------|
| Div      | Arithmetic | Synchronous | Supported | Supported | Supported   |

### Syntax

Div( *numNumerator*, *numDenominator* )

#### **numNumerator**

The numerator of the division operation.

#### **numDenominator**

The denominator of the division operation.

### Returned value

This function returns the dividend only as a whole number. The remainder is omitted.

 **Tip:** Use the [Mod](#) function to get the remainder instead of the dividend.

### Examples

| Tag Name        | Expression                                      |
|-----------------|-------------------------------------------------|
| <b>numValue</b> | <b>Div</b> ( 100, 8 ) // Returns the value 12.5 |
| <b>numValue</b> | <b>Div</b> ( 16, 4 ) // Returns the value 4     |
| <b>numValue</b> | <b>Div</b> ( 100, 12.5 ) // Returns the value 8 |

## Format

Format is a built-in scripting function that formats a numerical value and returns it as a string.

| Function | Group      | Execution   | Windows   | Embedded  | Thin Client |
|----------|------------|-------------|-----------|-----------|-------------|
| Format   | Arithmetic | Synchronous | Supported | Supported | Supported   |

## Syntax

Format(*strFlag*, *numValue*)

### strFlag

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 flags **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 flags **f**, **e**, **E**, **g** and **G**.
- F* is the specific format:

| Format | Description                                                                           |
|--------|---------------------------------------------------------------------------------------|
| d      | Decimal                                                                               |
| x      | Hexadecimal (alphabetic characters in lowercase)                                      |
| X      | Hexadecimal (alphabetic characters in uppercase)                                      |
| o      | Octal                                                                                 |
| b      | Binary                                                                                |
| f      | Floating-point                                                                        |
| e      | Scientific notation (e in lowercase)                                                  |
| E      | Scientific notation (E in uppercase)                                                  |
| g      | Rounded (e in lowercase, when applicable)                                             |
| G      | Rounded (E in uppercase, when applicable)                                             |
| s      | String                                                                                |
| c      | ASCII character (i.e., the numerical value is interpreted as an ASCII character code) |
| h      | Hour (hh : mm : ss)                                                                   |

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.

### Returned value

This function returns a string that contains the formatted numerical value.

### 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

| Tag Name | Expression                                                      |
|----------|-----------------------------------------------------------------|
| Tag      | <code>Format( "%d", 12.34 )</code> // Returned value = "12"     |
| Tag      | <code>Format( "%04d", 12.34 )</code> // Returned value = "0012" |
| Tag      | <code>Format( "%4d", 12.34 )</code> // Returned value = "12"    |

| Tag Name | Expression                                                   |
|----------|--------------------------------------------------------------|
| Tag      | <code>Format( "%x", 26 )</code> // Returned value = "1a"     |
| Tag      | <code>Format( "%04x", 26 )</code> // Returned value = "001a" |
| Tag      | <code>Format( "%4x", 26 )</code> // Returned value = "1a"    |

| Tag Name | Expression                                                   |
|----------|--------------------------------------------------------------|
| Tag      | <code>Format( "%X", 26 )</code> // Returned value = "1A"     |
| Tag      | <code>Format( "%04X", 26 )</code> // Returned value = "001A" |
| Tag      | <code>Format( "%4X", 26 )</code> // Returned value = "1A"    |

| Tag Name | Expression                                                   |
|----------|--------------------------------------------------------------|
| Tag      | <code>Format( "%o", 16 )</code> // Returned value = "20"     |
| Tag      | <code>Format( "%04o", 16 )</code> // Returned value = "0020" |
| Tag      | <code>Format( "%4o", 16 )</code> // Returned value = "20"    |

| Tag Name | Expression                                                  |
|----------|-------------------------------------------------------------|
| Tag      | <code>Format( "%b", 2 )</code> // Returned value = "10"     |
| Tag      | <code>Format( "%4b", 2 )</code> // Returned value = "0010"  |
| Tag      | <code>Format( "%04b", 2 )</code> // Returned value = "0010" |

| Tag Name | Expression                                                          |
|----------|---------------------------------------------------------------------|
| Tag      | <code>Format( "%0.1f", 12.34 )</code> // Returned value = "12.3"    |
| Tag      | <code>Format( "%06.1f", 12.34 )</code> // Returned value = "0012.3" |
| Tag      | <code>Format( "%6.1f", 12.34 )</code> // Returned value = "12.3"    |

| Tag Name | Expression                                                             |
|----------|------------------------------------------------------------------------|
| Tag      | <code>Format( "%e", 12.34 )</code> // Returned value = "1.234000e+001" |
| Tag      | <code>Format( "%0.1e", 12.34 )</code> // Returned value = "1.2e+001"   |
| Tag      | <code>Format( "%09.1e", 12.34 )</code> // Returned value = "01.2e+001" |
| Tag      | <code>Format( "%9.1e", 12.34 )</code> // Returned value = "1.2e+001"   |

| Tag Name | Expression                                                             |
|----------|------------------------------------------------------------------------|
| Tag      | <code>Format( "%E", 12.34 )</code> // Returned value = "1.234000E+001" |
| Tag      | <code>Format( "%0.1E", 12.34 )</code> // Returned value = "1.2E+001"   |
| Tag      | <code>Format( "%09.1E", 12.34 )</code> // Returned value = "01.2E+001" |
| Tag      | <code>Format( "%9.1E", 12.34 )</code> // Returned value = "1.2E+001"   |

| Tag Name | Expression                                                         |
|----------|--------------------------------------------------------------------|
| Tag      | <code>Format( "%0.1g", 12.34 )</code> // Returned value = "1e+001" |
| Tag      | <code>Format( "%0.2g", 12.34 )</code> // Returned value = "12"     |
| Tag      | <code>Format( "%0.3g", 12.34 )</code> // Returned value = "12.3"   |
| Tag      | <code>Format( "%05.3g", 12.34 )</code> // Returned value = "012.3" |

| Tag Name | Expression                                                           |
|----------|----------------------------------------------------------------------|
| Tag      | <code>Format( "%5.3g", 12.34 )</code> // Returned value = "12.3"     |
| Tag Name | Expression                                                           |
| Tag      | <code>Format( "%0.1G", 12.34 )</code> // Returned value = "1E+001"   |
| Tag      | <code>Format( "%0.2G", 12.34 )</code> // Returned value = "12"       |
| Tag      | <code>Format( "%0.3G", 12.34 )</code> // Returned value = "12.3"     |
| Tag      | <code>Format( "%05.3G", 12.34 )</code> // Returned value = "012.3"   |
| Tag      | <code>Format( "%5.3G", 12.34 )</code> // Returned value = "12.3"     |
| Tag Name | Expression                                                           |
| Tag      | <code>Format( "%s", 12.34 )</code> // Returned value = "12"          |
| Tag      | <code>Format( "%04s", 12.34 )</code> // Returned value = "0012"      |
| Tag      | <code>Format( "%4s", 12.34 )</code> // Returned value = "12"         |
| Tag Name | Expression                                                           |
| Tag      | <code>Format( "%c", 97 )</code> // Returned value = "a"              |
| Tag      | <code>Format( "%4c", 97 )</code> // Returned value = "a"             |
| Tag      | <code>Format( "%04c", 97 )</code> // Returned value = "000a"         |
| Tag Name | Expression                                                           |
| Tag      | <code>Format( "%h", 30 )</code> // Returned value = "00:00:30"       |
| Tag      | <code>Format( "%h", 60 )</code> // Returned value = "00:01:00"       |
| Tag      | <code>Format( "%h", 90 )</code> // Returned value = "00:01:30"       |
| Tag      | <code>Format( "%h", 3600 )</code> // Returned value = "01:00:00"     |
| Tag Name | Expression                                                           |
| Tag      | <code>Format( "##.#", 26.56789 )</code> // Returned value = "26.6"   |
| Tag      | <code>Format( "#.##", 26.56789 )</code> // Returned value = "26.57"  |
| Tag      | <code>Format( "##.##", 26.56789 )</code> // Returned value = "26.57" |

## GetBit

GetBit is a built-in scripting function that gets the value of a single bit in a numeric value.

| Function | Group      | Execution   | Windows   | Embedded  | Thin Client |
|----------|------------|-------------|-----------|-----------|-------------|
| GetBit   | Arithmetic | Synchronous | Supported | Supported | Supported   |

## 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

| Tag Name | Expression                                                                                                              |
|----------|-------------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>GetBit( "numSource", 4 )</code> // If the tag <b>numSource</b> holds a value of 15, then this function returns 0. |
| Tag      | <code>GetBit( "numSource", 1 )</code> // If the tag <b>numSource</b> holds a value of 19, then this function returns 1. |

## Mod

Mod is a built-in scripting function that gets the remainder from a division operation.

| Function | Group      | Execution   | Windows   | Embedded  | Thin Client |
|----------|------------|-------------|-----------|-----------|-------------|
| Mod      | Arithmetic | Synchronous | Supported | Supported | Supported   |

| Group      | Execution   | Windows PC | Windows CE | Thin Client |
|------------|-------------|------------|------------|-------------|
| Arithmetic | Synchronous | Supported  | Supported  | Supported   |

## 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

| Tag Name | Expression                                             |
|----------|--------------------------------------------------------|
| Tag      | <code>Mod( 50, 4 )</code> // Returned value = 2        |
| Tag      | <code>Mod( 16, 4 )</code> // Returned value = 0        |
| Tag      | <code>Mod( 100, 8.2 )</code> // Returned value = 1.600 |

## Pow

Pow is a built-in scripting function that gets the result of raising a numeric value to a specified exponent.

| Function | Group      | Execution   | Windows   | Embedded  | Thin Client |
|----------|------------|-------------|-----------|-----------|-------------|
| Pow      | Arithmetic | Synchronous | Supported | Supported | Supported   |

## 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

| Tag Name | Expression                                          |
|----------|-----------------------------------------------------|
| Tag      | <code>Pow( 2, 3 )</code> // Returned value = 8      |
| Tag      | <code>Pow( 10, 4 )</code> // Returned value = 10000 |

## ResetBit

Resets a single bit in an Integer tag to 0.

| Function | Group      | Execution   | Windows   | Embedded  | Thin Client |
|----------|------------|-------------|-----------|-----------|-------------|
| ResetBit | Arithmetic | Synchronous | Supported | Supported | Supported   |

## Syntax

`ResetBit( tagName, numBitNumber )`

### tagName

The name of an Integer tag where the bit value will be reset.

 **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

|   |                    |
|---|--------------------|
| 0 | No error           |
| 1 | Invalid parameter  |
| 2 | Tag does not exist |

## 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

| Tag Name | Expression                                                                                                                                                                      |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>ResetBit( "numSource", 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. |
| Tag      | <code>ResetBit( "numSource", 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. |

## Round

Rounds `numValue` to the nearest integer.

| Function | Group      | Execution   | Windows   | Embedded  | Thin Client |
|----------|------------|-------------|-----------|-----------|-------------|
| Round    | Arithmetic | Synchronous | Supported | Supported | Supported   |

## 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

| Tag Name | Expression                                             |
|----------|--------------------------------------------------------|
| Tag      | <code>Round( "345.87" )</code> // Returned value = 346 |
| Tag      | <code>Round( "65.323" )</code> // Returned value = 65  |

## SetBit

Sets a single bit in an Integer tag to 1.

| Function | Group      | Execution   | Windows   | Embedded  | Thin Client |
|----------|------------|-------------|-----------|-----------|-------------|
| SetBit   | Arithmetic | Synchronous | Supported | Supported | Supported   |

### 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

|   |                    |
|---|--------------------|
| 0 | No error           |
| 1 | Invalid parameter  |
| 2 | Tag does not exist |

### 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

| Tag Name | Expression                                                                                                                                                                     |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>GetBit( "numSource", 4 )</code> // If the tag <code>numSource</code> held a value of 0, then this function returns 0 and <code>numSource</code> holds a new value of 16. |
| Tag      | <code>GetBit( "numSource", 1 )</code> // If the tag <code>numSource</code> held the value 17, then this function returns 0 and <code>numSource</code> holds a new value of 19. |

## Sqrt

Takes the square root of `numValue`.

| Function | Group      | Execution   | Windows   | Embedded  | Thin Client |
|----------|------------|-------------|-----------|-----------|-------------|
| Sqrt     | Arithmetic | Synchronous | Supported | Supported | Supported   |

## Syntax

`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

| Tag Name | Expression                                            |
|----------|-------------------------------------------------------|
| Tag      | <code>SQRT( 25 )</code> // Returns the value 5        |
| Tag      | <code>SQRT( 67 )</code> // Returns the value 8.185353 |

## Swap16

Swaps the two lower bytes of a tag.

| Function | Group      | Execution   | Windows   | Embedded  | Thin Client |
|----------|------------|-------------|-----------|-----------|-------------|
| Swap16   | Arithmetic | Synchronous | Supported | Supported | Supported   |

## Syntax

`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

| Tag Name | Expression                                                                                                               |
|----------|--------------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>Swap16( 16 )</code> // 16 = 000000000010000 in binary. Returned value = 4096 = 0001000000000000 in binary.         |
| Tag      | <code>Swap16( 43760 )</code> // 43760 = 1010010111110000 in binary. Returned value = 61610 = 1111000010100101 in binary. |

## Swap32

Swaps two words in a tag.

| Function | Group      | Execution   | Windows   | Embedded  | Thin Client |
|----------|------------|-------------|-----------|-----------|-------------|
| Swap32   | Arithmetic | Synchronous | Supported | Supported | Supported   |

## Syntax

`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.



## 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.

| Function | Group       | Execution   | Windows   | Embedded  | Thin Client |
|----------|-------------|-------------|-----------|-----------|-------------|
| Avg      | Statistical | Synchronous | Supported | Supported | Supported   |

### 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

| Tag Name | Expression                                                                                                               |
|----------|--------------------------------------------------------------------------------------------------------------------------|
| Tag      | <b>Avg( 1, 2.34, 5, 7, 4, 8, 9.4 )</b> // Returned value = 5.248571                                                      |
| Tag      | <b>Avg( 1, 5, -9, 0, 5, 3 )</b> // Returned value = 0.833333                                                             |
| Tag Name | Expression                                                                                                               |
| Tag      | <b>Avg( "tagArray[1]", 3 )</b> // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 30     |
| Tag      | <b>Avg( "tagArray[1]", 3, 10 )</b> // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 40 |

### Max

Returns the maximum value of a set of numbers.

| Function | Group       | Execution   | Windows   | Embedded  | Thin Client |
|----------|-------------|-------------|-----------|-----------|-------------|
| Max      | Statistical | Synchronous | Supported | Supported | Supported   |

### 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**

| Tag Name | Expression                                                                                                                     |
|----------|--------------------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>Max( 1, 2.34, 5, 7, 4, 8, 9.4 )</code> // Returned value = 9.4                                                           |
| Tag      | <code>Max( 1, 5, -9, 0, 5, 3 )</code> // Returned value = 5                                                                    |
| Tag      | <code>Max( "tagArray[1]", 3 )</code> // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 60     |
| Tag      | <code>Max( "tagArray[1]", 3, 10 )</code> // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 60 |

**Min**

Returns the minimum value of a set of numbers.

| Function | Group       | Execution   | Windows   | Embedded  | Thin Client |
|----------|-------------|-------------|-----------|-----------|-------------|
| Min      | Statistical | Synchronous | Supported | Supported | Supported   |

**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 `Min( numValue1, numValue2, ... , numValueN)` format.
  - If the first parameter is an array tag in double-quotes or a string tag, you must use the `Min( "tagArray", numSample, optNumIgnore )` format.

**numValue (1...N)**

Integer or Real tags containing the numbers to be analyzed.

**tagArray**

Name of an 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 a value to be ignored in the analysis.

## Returned value

Returns the minimum value of the set.

## Examples

| Tag Name | Expression                                                                                                                     |
|----------|--------------------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>Min( 1, 2.34, 5, 7, 4, 8, 9.4 )</code> // Returned value = 1                                                             |
| Tag      | <code>Min( 1, 5, -9, 0, 5, 3 )</code> // Returned value = -9                                                                   |
| Tag      | <code>Min( "tagArray[1]", 3 )</code> // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 10     |
| Tag      | <code>Min( "tagArray[1]", 3, 10 )</code> // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 20 |

## Rand

Generates a random number between 0 and 1.

| Function | Group       | Execution   | Windows   | Embedded  | Thin Client |
|----------|-------------|-------------|-----------|-----------|-------------|
| Rand     | Statistical | Synchronous | Supported | Supported | Supported   |

## Syntax

`Rand ( )`

This function has no parameters.

## Returned value

Returns a real number between 0 and 1.

## Examples

| Tag Name | Expression                                                |
|----------|-----------------------------------------------------------|
| Tag      | <code>Rand ( )</code> // Returned value = ?, Where: 0<?<1 |

## 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**.

| Function | Group       | Execution   | Windows   | Embedded  | Thin Client |
|----------|-------------|-------------|-----------|-----------|-------------|
| Exp      | Logarithmic | Synchronous | Supported | Supported | Supported   |

### Syntax

Exp ( numValue )

**numValue**

Integer or Real tag containing the exponent of **e**.

### Returned value

Returns the value of  $e^{numValue}$ .

### Examples

| Tag Name | Expression                                    |
|----------|-----------------------------------------------|
| Tag      | Exp( 1 ) // Returned value = 2.718282         |
| Tag      | Exp( 5.25896 ) // Returned value = 192.281415 |

### Log

Calculates the natural log of **numValue**.

| Function | Group       | Execution   | Windows   | Embedded  | Thin Client |
|----------|-------------|-------------|-----------|-----------|-------------|
| Log      | Logarithmic | Synchronous | Supported | Supported | Supported   |

### 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

| Tag Name | Expression                              |
|----------|-----------------------------------------|
| Tag      | Log( 2.718282 ) // Returned value = 1   |
| Tag      | Log( 100 ) // Returned value = 4.605170 |

### Log10

Calculates the log base 10 of **numValue**.

| Function | Group       | Execution   | Windows   | Embedded  | Thin Client |
|----------|-------------|-------------|-----------|-----------|-------------|
| Log10    | Logarithmic | Synchronous | Supported | Supported | Supported   |

## Syntax

`Log10( numValue )`

### **numValue**

Integer or Real tag, from which the log base 10 is taken.

## Returned value

Returns the value of `log10( 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

| Tag Name | Expression                                               |
|----------|----------------------------------------------------------|
| Tag      | <code>Log10( 1000 )</code> // Returned value = 3         |
| Tag      | <code>Log10( 43.05 )</code> // Returned value = 1.633973 |

## 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.

| Function | Group   | Execution   | Windows   | Embedded  | Thin Client |
|----------|---------|-------------|-----------|-----------|-------------|
| False    | Logical | Synchronous | Supported | Supported | Supported   |

### Syntax

False ( *TagOrExpression* )

#### TagOrExpression

Tag or expression to be used in the function.

### Returned value

|   |                                                         |
|---|---------------------------------------------------------|
| 0 | If the tag or expression is <i>not</i> logically false. |
| 1 | If the tag or expression is logically false.            |

 **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

| Tag Name | Expression                                   |
|----------|----------------------------------------------|
| Tag      | <b>False</b> ( 1 ) // Returned value = 0     |
| Tag      | <b>False</b> ( 5 < 2 ) // Returned value = 1 |

### If

Determines whether the contents of **numExpression** are logically true, then returns the value of **numThen** or **optNumElse** accordingly.

| Function | Group   | Execution   | Windows   | Embedded  | Thin Client |
|----------|---------|-------------|-----------|-----------|-------------|
| If       | Logical | Synchronous | Supported | Supported | Supported   |

### Syntax

If( *numExpression*, *numThen*, *optNumElse* )

#### numExpression

Tag or expression used as the condition in the function.

#### numThen

Tag or expression used if the condition is logically true.

#### optNumElse

Optional tag or expression used if the condition is logically false.

### Returned value

|                   |                                                                                                   |
|-------------------|---------------------------------------------------------------------------------------------------|
| <b>numThen</b>    | If the <b>numExpression</b> is logically true.                                                    |
| <b>optNumElse</b> | If the <b>numExpression</b> is logically false.                                                   |
| No value returned | If the <b>numExpression</b> is logically false and there is no <b>optNumElse</b> in the function. |

## Notes

The numExpression parameter can be a combination of logic statements (**AND**, **OR**, and **NOT**). For example, `If(TagA>TagB AND TagA=10,1,0)`.

The numThen parameter can be another function, including the `If()` function. Therefore, you can use `If()` functions in cascade. For example, `if(TagA>TagB,If(TagA<TagC,1, 2),3)`.

## Examples

| Tag Name | Expression                                                                       |
|----------|----------------------------------------------------------------------------------|
| Tag      | <code>If( 5 &gt; 4,10, 6 )</code> // Returned value = 10                         |
| Tag      | <code>If( 5 &lt; 2, 0, 2 )</code> // Returned value = 0                          |
| Tag      | <code>If( 3 = 9, 67 )</code> // No Returned value. (Tag retains previous value.) |

## Toggle

Returns the toggled value from the contents of numValue tag.

| Function | Group   | Execution   | Windows   | Embedded  | Thin Client |
|----------|---------|-------------|-----------|-----------|-------------|
| Toggle   | Logical | Synchronous | Supported | Supported | Supported   |

## Syntax

`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

| Tag Name | Expression                                                                                                           |
|----------|----------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>Toggle( MyBoolTag )</code> // Returned value = 1 if MyBoolTag value equals 0, or 0 if MyBoolTag value equals 1 |
| Tag      | <code>Toggle( numValue )</code> // Returned value = toggled value of the number in the numValue tag                  |

## True

Determines whether the specified tag or expression is logically true.

| Function | Group   | Execution   | Windows   | Embedded  | Thin Client |
|----------|---------|-------------|-----------|-----------|-------------|
| True     | Logical | Synchronous | Supported | Supported | Supported   |

## Syntax

`True( TagOrExpression )`

### TagOrExpression

Tag or expression to be used in the function.

## Returned value

|   |                                                 |
|---|-------------------------------------------------|
| 0 | If the tag or expression is not logically true. |
| 1 | If the tag or expression is logically true.     |

 **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

| Tag Name | Expression                                          |
|----------|-----------------------------------------------------|
| Tag      | <code>True( 1 )</code> // Returned value = 1        |
| Tag      | <code>True( 5 &lt; 2 )</code> // Returned value = 0 |

## 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.

| Function | Group  | Execution   | Windows   | Embedded  | Thin Client |
|----------|--------|-------------|-----------|-----------|-------------|
| Asc2Str  | String | Synchronous | Supported | Supported | Supported   |

### 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

| Tag Name | Expression                                                                     |
|----------|--------------------------------------------------------------------------------|
| Tag      | <code>Asc2Str( 65 )</code> // Returned value = "A"                             |
| Tag      | <code>Asc2Str( 65, 66, 67 )</code> // Returned value = "ABC"                   |
| Tag      | <code>Asc2Str( Array[0], Array[1], Array[2] )</code> // Returned value = "ABC" |

### CharToValue

This function converts a string to Unicode character codes and then stores those values in an integer array.

| Function    | Group  | Execution   | Windows   | Embedded  | Thin Client |
|-------------|--------|-------------|-----------|-----------|-------------|
| CharToValue | String | Synchronous | Supported | Supported | Supported   |

### Syntax

```
CharToValue("tagString" , "tagArray")
```

#### tagString

The name of the string tag, whose value will be converted.

 **Note:** The tag name must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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")
```

If StrTag = "ABC", then Array[10] = 65, Array[11] = 66, and Array[12] = 67:

```
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.

| Function     | Group  | Execution   | Windows   | Embedded  | Thin Client |
|--------------|--------|-------------|-----------|-----------|-------------|
| CharToValueW | String | Synchronous | Supported | Supported | Supported   |

## Syntax

```
CharToValueW("tagString" , "tagArray")
```

### tagString

The name of the string tag, whose value will be converted.

 **Note:** The tag name must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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")
```

If StrTag = "Studio", then Array[10] = 29779 ("St"), Array[11] = 25717 ("ud"), and Array[12] = 28521 ("io"):

```
CharToValue("StrTag", "Array[10]")
```

## ClassMembersToStrVector

Transfers values from a Class tag to an Array tag.

| Function                | Group  | Execution   | Windows   | Embedded  | Thin Client |
|-------------------------|--------|-------------|-----------|-----------|-------------|
| ClassMembersToStrVector | String | Synchronous | Supported | Supported | Supported   |

## 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

|    |                                                                           |
|----|---------------------------------------------------------------------------|
| -6 | Array size of <i>strClassTag</i> is not big enough for <i>numStartPos</i> |
| -5 | <i>strClassTag</i> is not a Class tag                                     |
| -4 | <i>strClassTag</i> is not found                                           |
| -3 | <i>strArrayTag</i> is not found                                           |
| -2 | Invalid data type of the parameters                                       |
| -1 | Invalid number of parameters                                              |
| 0  | Transferred successfully                                                  |

## Notes

If strClassTag has more than one member, the value of each member will be transferred to strArrayTag. Therefore, it is important to make sure that the array size of strArrayTag is big enough to receive all values from strClassTag.

## Examples

| Tag Name | Expression                                                 |
|----------|------------------------------------------------------------|
| Tag      | ClassMembersToStrVector( "Classtag", 5, 3, "Arraytag" )    |
| Tag      | ClassMembersToStrVector( "Classtag", 5, 3, "Arraytag", 0 ) |
| Tag      | ClassMembersToStrVector ( TagName, 0, 1, ArrayName )       |

## NCopy

Copies a defined section of a larger string.

| Function | Group  | Execution   | Windows   | Embedded  | Thin Client |
|----------|--------|-------------|-----------|-----------|-------------|
| NCopy    | String | Synchronous | Supported | Supported | Supported   |

## Syntax

```
NCopy(strSource, numStartChar, 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

| Tag Name | Expression                                                                        |
|----------|-----------------------------------------------------------------------------------|
| Tag      | <code>NCopy( "Studio version 7.0", 7, 7 )</code> // Returned value = "ersion"     |
| Tag      | <code>NCopy( "Technical Reference", 0, 9 )</code> // Returned value = "Technical" |

 **Note:** The first character in the string will be assigned the value 0.

### Num

Converts a string into a float.

| Function | Group  | Execution   | Windows   | Embedded  | Thin Client |
|----------|--------|-------------|-----------|-----------|-------------|
| Num      | String | Synchronous | Supported | Supported | Supported   |

### Syntax

`Num( 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

| Tag Name | Expression                                                          |
|----------|---------------------------------------------------------------------|
| Tag      | <code>Num( "321654.987" )</code> // Returned value = 321654.987     |
| Tag      | <code>Num( "5.6589626246" )</code> // Returned value = 5.6589626246 |

 **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.

| Function | Group  | Execution   | Windows   | Embedded  | Thin Client |
|----------|--------|-------------|-----------|-----------|-------------|
| Str      | String | Synchronous | Supported | Supported | Supported   |

### 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

| Tag Name | Expression                                                      |
|----------|-----------------------------------------------------------------|
| Tag      | <code>Str( 321654.987 )</code> // Returned value = "321654.987" |
| Tag      | <code>Str( 5.65896246 )</code> // Returned value = "5.658962"   |

## Str2Asc

This function converts a character to its corresponding Unicode character code.

| Function | Group  | Execution   | Windows   | Embedded  | Thin Client |
|----------|--------|-------------|-----------|-----------|-------------|
| Str2Asc  | String | Synchronous | Supported | Supported | Supported   |

## 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

| Tag Name | Expression                                          |
|----------|-----------------------------------------------------|
| Tag      | <code>Str2Asc( "C" )</code> // Returned value = 67  |
| Tag      | <code>Str2Asc( "o" )</code> // Returned value = 111 |

## StrCompare

Compares two strings to see if they are identical.

| Function   | Group  | Execution   | Windows   | Embedded  | Thin Client |
|------------|--------|-------------|-----------|-----------|-------------|
| StrCompare | String | Synchronous | Supported | Supported | Supported   |

## 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

|    |                                                                                           |
|----|-------------------------------------------------------------------------------------------|
| -1 | The value of <code>strValue1</code> is less than the value of <code>strValue2</code> .    |
| 0  | <code>strValue1</code> and <code>strValue2</code> are identical.                          |
| 1  | The value of <code>strValue1</code> is greater than the value of <code>strValue2</code> . |

## Examples

| Tag Name | Expression                                                         |
|----------|--------------------------------------------------------------------|
| Tag      | <code>StrCompare( "Text1", "Text2" )</code> // Returned value = #1 |

| Tag Name                             | Expression                                                   |
|--------------------------------------|--------------------------------------------------------------|
| Tag<br>Tag1 = "Text" Tag2 = "Text"   | <code>StrCompare( Tag1, Tag2 )</code> // Returned value = 0  |
| Tag Tag1 = "Text1"<br>Tag2 = "Text2" | <code>StrCompare( Tag1, Tag2 )</code> // Returned value = #1 |

## 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").

| Function         | Group  | Execution   | Windows   | Embedded  | Thin Client |
|------------------|--------|-------------|-----------|-----------|-------------|
| StrCompareNoCase | String | Synchronous | Supported | Supported | Supported   |

### 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

|    |                                                                               |
|----|-------------------------------------------------------------------------------|
| -1 | The value of <i>strValue1</i> is less than the value of <i>strValue2</i> .    |
| 0  | <i>strValue1</i> and <i>strValue2</i> are identical.                          |
| 1  | The value of <i>strValue1</i> is greater than the value of <i>strValue2</i> . |

### Examples

| Tag Name                             | Expression                                                              |
|--------------------------------------|-------------------------------------------------------------------------|
| Tag                                  | <code>StrCompareNoCase( "Text1", "TEXT1" )</code> // Returned value = 0 |
| Tag Tag1 = "Text1"<br>Tag2 = "TEXT1" | <code>StrCompareNoCase( Tag1, Tag2 )</code> // Returned value = 0       |

## StrFromInt

Converts an integer into its string representation in another base number system, such as binary (base-2) or octal (base-8).

| Function   | Group  | Execution   | Windows   | Embedded  | Thin Client |
|------------|--------|-------------|-----------|-----------|-------------|
| StrFromInt | String | Synchronous | Supported | Supported | Supported   |

### 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

| Tag Name | Expression                                                   |
|----------|--------------------------------------------------------------|
| Tag      | <code>StrFromInt( 26, 2 )</code> // Returned value = "11010" |
| Tag      | <code>StrFromInt( 26, 8 )</code> // Returned value = "32"    |

## 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.

| Function    | Group  | Execution   | Windows   | Embedded  | Thin Client |
|-------------|--------|-------------|-----------|-----------|-------------|
| StrFromReal | String | Synchronous | Supported | Supported | Supported   |

## 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:

| Value of strType | Description                                               |
|------------------|-----------------------------------------------------------|
| f                | Formatted in floating-point notation.                     |
| e                | Formatted in exponential notation with a lower-case "e".  |
| E                | Formatted in exponential notation with an upper-case "E". |

## 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.

| Function    | Group  | Execution   | Windows   | Embedded  | Thin Client |
|-------------|--------|-------------|-----------|-----------|-------------|
| StrFromTime | String | Synchronous | Supported | Supported | Supported   |

### 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:

| Value of numType | Description                                                                                              |
|------------------|----------------------------------------------------------------------------------------------------------|
| 1                | Displays the <i>date</i> in the same format that is selected in the Control Panel on the local computer. |
| 2                | Displays the <i>time</i> in the same format that is selected in the Control Panel on the local computer. |
| 3                | Displays a standard 24-character string that shows both date and time.                                   |
| 4                | Displays the abbreviated name of the day of the week.                                                    |
| 5                | Displays the full name of the day of the week.                                                           |

### 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

 **Note:** The examples below are for a computer set to Eastern Standard Time (or UTC -05:00).

| Tag Name | Expression                                                                          |
|----------|-------------------------------------------------------------------------------------|
| Tag      | <code>StrFromTime( 86400, 1 ) // Returned value = "1/1/70"</code>                   |
| Tag      | <code>StrFromTime( 86400, 2 ) // Returned value = "07:00:00 PM"</code>              |
| Tag      | <code>StrFromTime( 86400, 3 ) // Returned value = "Thu Jan 01 19:00:00 1970"</code> |
| Tag      | <code>StrFromTime( 86400, 4 ) // Returned value = "Thu"</code>                      |
| Tag      | <code>StrFromTime( 86400, 5 ) // Returned value = "Thursday"</code>                 |

## StrGetElement

Gets a specific element from a string source.

| Function      | Group  | Execution   | Windows   | Embedded  | Thin Client |
|---------------|--------|-------------|-----------|-----------|-------------|
| StrGetElement | String | Synchronous | Supported | Supported | Supported   |

### Syntax

`StrGetElement( strSource, strDelimiter, 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**

| Tag Name | Expression                                                            |
|----------|-----------------------------------------------------------------------|
| Tag      | <code>StrGetElement( "a b c", " ", 2 )</code> // returned value = "b" |
| Tag      | <code>StrGetElement( "a,b,c", ",", 3 )</code> // returned value = "c" |

**StrLeft**

Copies the first characters of a larger string.

| Function | Group  | Execution   | Windows   | Embedded  | Thin Client |
|----------|--------|-------------|-----------|-----------|-------------|
| StrLeft  | String | Synchronous | Supported | Supported | Supported   |

**Syntax**

`StrLeft( strSource, 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**

| Tag Name | Expression                                                                      |
|----------|---------------------------------------------------------------------------------|
| Tag      | <code>StrLeft( "Studio version 7.0", 8 )</code> // Returned value = Studio v    |
| Tag      | <code>StrLeft ( "Technical Reference", 9 )</code> // Returned value = Technical |

**StrLen**

Determines the length of a string.

| Function | Group  | Execution   | Windows   | Embedded  | Thin Client |
|----------|--------|-------------|-----------|-----------|-------------|
| StrLen   | String | Synchronous | Supported | Supported | Supported   |

**Syntax**

`StrLen( strSource )`

**strSource**

The string.

**Returned value**

Returns an integer that is the number of characters in the string.

**Examples**

| Tag Name | Expression                                                          |
|----------|---------------------------------------------------------------------|
| Tag      | <code>StrLen( "Studio version 7.0" )</code> // Returned value = 18  |
| Tag      | <code>StrLen( "Technical Reference" )</code> // Returned value = 19 |

**StrLower**

Converts a string to all lower case characters.

| Function | Group  | Execution   | Windows   | Embedded  | Thin Client |
|----------|--------|-------------|-----------|-----------|-------------|
| StrLower | String | Synchronous | Supported | Supported | Supported   |

**Syntax**

```
StrLower(strSource)
```

**strSource**

The string to be converted.

**Returned value**

Returns the string, where all the characters are in lowercase.

**Examples**

| Tag Name | Expression                                                                               |
|----------|------------------------------------------------------------------------------------------|
| Tag      | <code>StrLower( "Studio version 7.0" )</code> // Returned value = "studio version 7.0"   |
| Tag      | <code>StrLower( "Technical Reference" )</code> // Returned value = "technical reference" |

**StrRChr**

Isolates the final occurrence of a character sequence within a string.

| Function | Group  | Execution   | Windows   | Embedded  | Thin Client |
|----------|--------|-------------|-----------|-----------|-------------|
| StrRChr  | String | Synchronous | Supported | Supported | Supported   |

**Syntax**

```
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**

| Tag Name | Expression                                                                      |
|----------|---------------------------------------------------------------------------------|
| Tag      | <code>StrRChr( "Studio version 7.0", "i" )</code> // Returned value = "ion 7.0" |
| Tag      | <code>StrRChr( "Technical Reference", "n" )</code> // Returned value = "nce"    |

**StrRight**

Copies the last characters in a larger string.

| Function | Group  | Execution   | Windows   | Embedded  | Thin Client |
|----------|--------|-------------|-----------|-----------|-------------|
| StrRight | String | Synchronous | Supported | Supported | Supported   |

## Syntax

`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

| Tag Name | Expression                                                                        |
|----------|-----------------------------------------------------------------------------------|
| Tag      | <code>StrRight( "Studio version 7.0", 8 )</code> // Returned value = "sion 7.0"   |
| Tag      | <code>StrRight( "Technical Reference", 9 )</code> // Returned value = "Reference" |

## StrSetElement

Sets a specific element in a string source.

| Function      | Group  | Execution   | Windows   | Embedded  | Thin Client |
|---------------|--------|-------------|-----------|-----------|-------------|
| StrSetElement | String | Synchronous | Supported | Supported | Supported   |

## Syntax

`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

| Tag Name  | Expression                                              |
|-----------|---------------------------------------------------------|
| StringTag | <code>StrSetElement( strSource, " ", 2, "abcd" )</code> |
| StringTag | <code>StrSetElement( strSource, ",", 3, "defg" )</code> |

## StrStr

Isolates the first occurrence of a character sequence within a string.

| Function | Group  | Execution   | Windows   | Embedded  | Thin Client |
|----------|--------|-------------|-----------|-----------|-------------|
| StrStr   | String | Synchronous | Supported | Supported | Supported   |

## Syntax

`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.

**Examples**

| Tag Name | Expression                                                                             |
|----------|----------------------------------------------------------------------------------------|
| Tag      | <code>StrStr( "Studio version 7.0", "i" )</code> // Returned value = "io version 7.0"  |
| Tag      | <code>StrStr( "Technical Reference", "n" )</code> // Returned value ="nical Reference" |

**StrStrPos**

Finds the first occurrence of a character within a string.

| Function  | Group  | Execution   | Windows   | Embedded  | Thin Client |
|-----------|--------|-------------|-----------|-----------|-------------|
| StrStrPos | String | Synchronous | Supported | Supported | Supported   |

**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**

| Tag Name | Expression                                                                 |
|----------|----------------------------------------------------------------------------|
| Tag      | <code>StrStrPos( "Studio version 7.0", "i" )</code> // Returned value = 4  |
| Tag      | <code>StrStrPos( "Technical Reference", "a" )</code> // Returned value = 7 |

 **Note:** The first character in the string assigned the value 0.

**StrTrim**

Removes unwanted spaces from a string.

| Function | Group  | Execution   | Windows   | Embedded  | Thin Client |
|----------|--------|-------------|-----------|-----------|-------------|
| StrTrim  | String | Synchronous | Supported | Supported | Supported   |

**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 | Description                                                           |
|---------------------|-----------------------------------------------------------------------|
| 0                   | Removes all spaces from both the beginning and the end of the string. |
| 1                   | Removes all spaces only from the beginning of the string.             |
| 2                   | Removes all spaces only from the end of the string.                   |
| 3                   | Removes all spaces except for single spaces between words.            |

 **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

| Tag Name | Expression                                                                                  |
|----------|---------------------------------------------------------------------------------------------|
| Tag      | <code>StrTrim( " Studio version 7.0 " )</code> // Returned value = "Studio version 7.0"     |
| Tag      | <code>StrTrim( " Studio version 7.0 ", 0 )</code> // Returned value = "Studio version 7.0"  |
| Tag      | <code>StrTrim( " Studio version 7.0 ", 1 )</code> // Returned value = "Studio version 7.0"  |
| Tag      | <code>StrTrim( " Studio version 7.0 ", 2 )</code> // Returned value = " Studio version 7.0" |
| Tag      | <code>StrTrim( " Studio version 7.0 ", 3 )</code> // Returned value = "Studio version 7.0"  |

### StrTrimAll

Eliminates a specific char from the whole string.

| Function   | Group  | Execution   | Windows   | Embedded  | Thin Client |
|------------|--------|-------------|-----------|-----------|-------------|
| StrTrimAll | String | Synchronous | Supported | Supported | Supported   |

### 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

| Tag Name | Expression                                                                                   |
|----------|----------------------------------------------------------------------------------------------|
| Tag      | <code>StrTrimAll( "Studio version 7.0 ", " " )</code> // Returned value = "Studioversion7.0" |

### StrUpper

Converts a string to all uppercase characters.

| Function | Group  | Execution   | Windows   | Embedded  | Thin Client |
|----------|--------|-------------|-----------|-----------|-------------|
| StrUpper | String | Synchronous | Supported | Supported | Supported   |

## Syntax

`StrUpper( strSource )`

### strSource

The string.

## Returned value

Returns the string with all characters are in uppercase.

## Examples

| Tag Name | Expression                                                                               |
|----------|------------------------------------------------------------------------------------------|
| Tag      | <code>StrUpper( "Studio version 7.0" )</code> // Returned value = "STUDIO VERSION 6.1"   |
| Tag      | <code>StrUpper( "Technical Reference" )</code> // Returned value = "TECHNICAL REFERENCE" |

## ValueToChar

This function converts an integer array of Unicode character codes to a string.

| Function    | Group  | Execution   | Windows   | Embedded  | Thin Client |
|-------------|--------|-------------|-----------|-----------|-------------|
| ValueToChar | String | Synchronous | Supported | Supported | Supported   |

## Syntax

`ValueToChar( "tagArray" , 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.

 **Note:** The tag name must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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.

## Examples

If `Array[0] = 65`, `Array[1] = 66`, and `Array[2] = 67`, then the returned value will be "ABC":

`ValueToChar( "Array", 3 )`

If `Array[10] = 65`, `Array[11] = 66`, and `Array[12] = 67`, then the returned value will be "ABC":

`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.

| Function     | Group  | Execution   | Windows   | Embedded  | Thin Client |
|--------------|--------|-------------|-----------|-----------|-------------|
| ValueWToChar | String | Synchronous | Supported | Supported | Supported   |

## 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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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)
```

If Array[10] = 29779, Array[11] = 25717, and Array[12] = 28521, then the returned value is "Studio":

```
ValueWToChar("Array[10]", 3)
```

## Date & Time functions

These functions are used to interact with the system clock or manipulate timestamps.

### **ClockGetDate**

Calculates the date, based on how many seconds have elapsed since 12:00 AM GMT on January 1, 1970 (taking into account the current time zone of the computer).

| Function     | Group       | Execution   | Windows   | Embedded  | Thin Client |
|--------------|-------------|-------------|-----------|-----------|-------------|
| ClockGetDate | Date & Time | Synchronous | Supported | Supported | Supported   |

### Syntax

`ClockGetDate( numSeconds )`

### numSeconds

The number of seconds elapsed since 12:00 AM GMT on January 1, 1970.

### Returned value

Returns the date calculated in string format.

### Examples

| Tag Name | Expression                                                                                                          |
|----------|---------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>ClockGetDate( 0 )</code> // If the computer is in the Central time zone. Returned value = 12/31/1969          |
| Tag      | <code>ClockGetDate( 1018886359 )</code> // If the computer is in the Central time zone. Returned value = 04/15/2002 |

 **Note:** This function takes into account the current Time Zone as specified in the Control Panel of the local computer.

### **ClockGetDayOfWeek**

Calculates the day of the week, based on how many seconds have elapsed since 12:00 AM GMT on January 1, 1970 (taking into account the current time zone of the local computer).

| Function          | Group       | Execution   | Windows   | Embedded  | Thin Client |
|-------------------|-------------|-------------|-----------|-----------|-------------|
| ClockGetDayOfWeek | Date & Time | Synchronous | Supported | Supported | Supported   |

### Syntax

`ClockGetDayOfWeek( numSeconds )`

### numSeconds

The number of seconds elapsed since 12:00 AM GMT on January 1, 1970

### Returned value

Returns the day of the week (calculated in integer format) as follows:

|   |           |
|---|-----------|
| 0 | Sunday    |
| 1 | Monday    |
| 2 | Tuesday   |
| 3 | Wednesday |
| 4 | Thursday  |
| 5 | Friday    |
| 6 | Saturday  |

## Examples

| Tag Name | Expression                                                                                                      |
|----------|-----------------------------------------------------------------------------------------------------------------|
| Tag      | <code>ClockGetDayOfWeek( 0 )</code> // If the computer is in the Central time zone. Returned value = 3          |
| Tag      | <code>ClockGetDayOfWeek( 1018886359 )</code> // If the computer is in the Central time zone. Returned value = 1 |

 **Note:** This function takes into account the current Time Zone, as specified in the Control Panel of the local computer.

## ClockGetTime

Calculates the time based on how many seconds have elapsed since 12:00 AM GMT on January 1, 1970 (taking into account the current time zone as specified on the local computer).

| Function     | Group       | Execution   | Windows   | Embedded  | Thin Client |
|--------------|-------------|-------------|-----------|-----------|-------------|
| ClockGetTime | Date & Time | Synchronous | Supported | Supported | Supported   |

## Syntax

`ClockGetTime( numSeconds )`

### numSeconds

The number of seconds elapsed since 12:00 AM GMT on January 1, 1970.

### Returned value

Returns the time calculated in string format.

## Examples

| Tag Name | Expression                                                                                                        |
|----------|-------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>ClockGetTime( 0 )</code> // If the computer is in the Central time zone. Returned value = 18:00:00          |
| Tag      | <code>ClockGetTime( 1018886359 )</code> // If the computer is in the Central time zone. Returned value = 10:59:19 |

 **Note:** This function takes into account the current Time Zone, as specified in the Control Panel of the local computer.

 **Tip:** To convert the number of seconds strictly into the **HH:MM:SS** format, you must use the `Format()` function instead of the `ClockGetTime()` function.

## DateTime2Clock

Calculates how many seconds have elapsed since 12:00 AM GMT on January 1, 1970 (taking into account the current time zone specified on the local computer.)

| Function       | Group       | Execution   | Windows   | Embedded  | Thin Client |
|----------------|-------------|-------------|-----------|-----------|-------------|
| DateTime2Clock | Date & Time | Synchronous | Supported | Supported | Supported   |

## Syntax

`DateTime2Clock( strDate, strTime )`

### strDate

The date to be used in the calculation.

### strTime

The time to be used in the calculation.

### Returned value

Returns the number of seconds that have elapsed since 12:00 AM GMT on January 1, 1970.

## Examples

| Tag Name | Expression                                                                                                                           |
|----------|--------------------------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>DateTime2Clock( "12/31/1969", "18:00:00" )</code> // If the computer is in the Central time zone. Returned value = 0           |
| Tag      | <code>DateTime2Clock( "04/15/2002", "10:59:19" )</code> // If the computer is in the Central time zone. Returned value = 01018886359 |

 **Note:** This function takes into account the current Time Zone, as specified in the Control Panel of the local computer.

## GetClock

Calculates how many seconds have elapsed since 12:00 AM GMT on January 1, 1970, at the moment the function was run (taking into account the current time zone, as specified on the local computer).

| Function | Group       | Execution   | Windows   | Embedded  | Thin Client |
|----------|-------------|-------------|-----------|-----------|-------------|
| GetClock | Date & Time | Synchronous | Supported | Supported | Supported   |

## Syntax

`GetClock( )`

This function takes no parameters.

## Returned value

Returns the number of seconds that have elapsed since 12:00 AM GMT on January 1, 1970, at the moment the function was run.

## Examples

| Tag Name | Expression                                                                                             |
|----------|--------------------------------------------------------------------------------------------------------|
| Tag      | <code>GetClock( )</code> // If executed at 10:59:19 AM April 15th 2002 CST. Returned value = 101886359 |
| Tag      | <code>GetClock( )</code> // If executed at 00:00:00 January 1st 1970 GMT. Returned value = 0           |

 **Note:** This function takes the current Time Zone into account, as specified in the Control Panel of the local computer.

## Hour2Clock

Converts time in the **HH:MM:SS** format into seconds.

| Function   | Group       | Execution   | Windows   | Embedded  | Thin Client |
|------------|-------------|-------------|-----------|-----------|-------------|
| Hour2Clock | Date & Time | Synchronous | 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

| Tag Name | Expression                                                     |
|----------|----------------------------------------------------------------|
| Tag      | <code>Hour2Clock( "01:00:00" )</code> // Returned value = 3600 |

| Tag Name | Expression                                                      |
|----------|-----------------------------------------------------------------|
| Tag      | <code>Hour2Clock( "10:01:01" )</code> // Returned value = 36061 |

## SetSystemDate

Sets the date in the operating system's clock.

| Function      | Group       | Execution   | Windows   | Embedded  | Thin Client |
|---------------|-------------|-------------|-----------|-----------|-------------|
| SetSystemDate | Date & Time | Synchronous | Supported | Supported | Supported   |

### Syntax

`SetSystemDate( strDate )`

#### strDate

The date in **MM/DD/YYYY** format in which to set the clock.

### Returned value

Returns no values.

### Examples

| Tag Name | Expression                                                                              |
|----------|-----------------------------------------------------------------------------------------|
|          | <code>SetSystemDate( "04/15/2002" )</code> // Sets the system clock to April 15th 2002. |

## SetSystemTime

Sets the time in the operating system's clock.

| Function      | Group       | Execution   | Windows   | Embedded  | Thin Client |
|---------------|-------------|-------------|-----------|-----------|-------------|
| SetSystemTime | Date & Time | Synchronous | Supported | Supported | Supported   |

### Syntax

`SetSystemTime( strTime )`

#### strTime

The time in **HH:MM:SS** format in which to set the clock.

### Returned value

No Returned Value.

### Examples

| Tag Name | Expression                                                                       |
|----------|----------------------------------------------------------------------------------|
|          | <code>SetSystemTime( "15:45:18" )</code> // Sets the system clock to 3:45:18 PM. |

## 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.

| Function | Group         | Execution   | Windows   | Embedded  | Thin Client |
|----------|---------------|-------------|-----------|-----------|-------------|
| ACos     | Trigonometric | Synchronous | Supported | Supported | Supported   |

### 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

| Tag Name | Expression                                     |
|----------|------------------------------------------------|
| Tag      | <b>ACos</b> ( 1 ) // Returned value = 0.000000 |
| Tag      | <b>ACos</b> ( 0 ) // Returned value = 1.570796 |

### ASin

Calculates the Arc Sine of a value.

| Function | Group         | Execution   | Windows   | Embedded  | Thin Client |
|----------|---------------|-------------|-----------|-----------|-------------|
| ASin     | Trigonometric | Synchronous | Supported | Supported | Supported   |

### 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

| Tag Name | Expression                                     |
|----------|------------------------------------------------|
| Tag      | <b>ASin</b> ( 1 ) // Returned value = 1.570796 |
| Tag      | <b>ASin</b> ( 0 ) // Returned value = 0.000000 |

### ATan

Calculates the Arc Tangent of a value.

| Function | Group         | Execution   | Windows   | Embedded  | Thin Client |
|----------|---------------|-------------|-----------|-----------|-------------|
| ATan     | Trigonometric | Synchronous | Supported | Supported | Supported   |

## Syntax

**ATan**( numValue )

### numValue

Numerical tag from which the Arc Tangent will be taken.

## Returned value

Returns the Arc Tangent of numValue in radians.

## Examples

| Tag Name | Expression                                     |
|----------|------------------------------------------------|
| Tag      | <b>ATan</b> ( 1 ) // Returned value = 0.785398 |
| Tag      | <b>ATan</b> ( 0 ) // Returned value = 1.570796 |

## Cos

Calculates the Cosine of a value.

| Function | Group         | Execution   | Windows   | Embedded  | Thin Client |
|----------|---------------|-------------|-----------|-----------|-------------|
| Cos      | Trigonometric | Synchronous | Supported | Supported | Supported   |

## Syntax

**Cos**( numAngle )

### numAngle

The Angle (in radians) from which to calculate the Cosine.

## Returned value

Returns the Cosine of numAngle.

## Examples

| Tag Name | Expression                                           |
|----------|------------------------------------------------------|
| Tag      | <b>Cos</b> ( 1.570796 ) // Returned value = 0.000000 |
| Tag      | <b>Cos</b> ( 0 ) // Returned value = 1.000000        |

## Cot

Calculates the Cotangent of a value.

| Function | Group         | Execution   | Windows   | Embedded  | Thin Client |
|----------|---------------|-------------|-----------|-----------|-------------|
| Cot      | Trigonometric | Synchronous | Supported | Supported | Supported   |

## Syntax

**Cot**( numAngle )

### numAngle

The Angle (in radians) from which to calculate the Cotangent.

## Returned value

Returns the Cotangent of numAngle.

 **Note:** Although mathematically the tangent of Pi is infinite, IWS only returns the largest number possible.

**Examples**

| Tag Name | Expression                                                |
|----------|-----------------------------------------------------------|
| Tag      | <code>Cot( 0.785398 ) // Returned value = 1.000000</code> |
| Tag      | <code>Cot( 0 ) // Returned value = 0.000000</code>        |

**Pi**

Calculates the value of *pi*.

| Function | Group         | Execution   | Windows   | Embedded  | Thin Client |
|----------|---------------|-------------|-----------|-----------|-------------|
| Pi       | Trigonometric | Synchronous | Supported | Supported | Supported   |

**Syntax**

`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**

| Tag Name | Expression                                     |
|----------|------------------------------------------------|
| Tag      | <code>Pi() // Returned value = 3.141593</code> |

**Sin**

| Function | Group         | Execution   | Windows   | Embedded  | Thin Client |
|----------|---------------|-------------|-----------|-----------|-------------|
| Sin      | Trigonometric | Synchronous | Supported | Supported | Supported   |

**Description**

Calculates the Sine of a value.

**Syntax**

`Sin( numAngle )`

**numAngle**

The Angle (in radians) from which to calculate the Sine.

**Returned value**

Returns the Sine of numAngle.

**Examples**

| Tag Name | Expression                                                |
|----------|-----------------------------------------------------------|
| Tag      | <code>Sin( 0 ) // Returned value = 0.000000</code>        |
| Tag      | <code>Sin( 1.570796 ) // Returned value = 1.000000</code> |

**Tan**

Calculates the Tangent of a value.

| Function | Group         | Execution   | Windows   | Embedded  | Thin Client |
|----------|---------------|-------------|-----------|-----------|-------------|
| Tan      | Trigonometric | Synchronous | Supported | Supported | Supported   |

## Syntax

`Tan( numAngle )`

### numAngle

The Angle (in radians) from which to calculate the Tangent.

## Returned value

Returns the Tangent of numAngle.

 **Note:** Although mathematically the tangent of `Pi()` is infinite, IWS only returns the largest number possible.

## Examples

| Tag Name | Expression                                               |
|----------|----------------------------------------------------------|
| Tag      | <code>Tan( 0 )</code> // Returned value = 0.00000        |
| Tag      | <code>Tan( 0.785398 )</code> // Returned value = 1.00000 |

## 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.

| Function | Group  | Execution    | Windows   | Embedded  | Thin Client |
|----------|--------|--------------|-----------|-----------|-------------|
| Close    | Screen | Asynchronous | Supported | Supported | Supported   |

### Syntax

```
Close(strScreen{ | , optNumID }
```

#### **strScreen**

The name of the screen (not including the .scr extension) to be closed.

 **Note:** Some Web servers are case-sensitive. If you plan to deploy your project as a Web application, then you should use only lowercase letters for the screen name.

#### **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 [Open](#) function.)

This is an optional parameter. If no value is specified, then the default ID is 0.

### Returned value

This function does not return any value.

### Notes

This function cannot be used with [Tasks](#) or in the [Global Procedures](#) script.

Also, 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":

```
Close("main")
```

Close the screen named "alarms":

```
Close("alarms")
```

Close the screen named "main" with ID 10:

```
Close("main", 10)
```

### Open

Open is a built-in scripting function that opens a project screen.

| Function | Group  | Execution    | Windows   | Embedded  | Thin Client |
|----------|--------|--------------|-----------|-----------|-------------|
| Open     | Screen | Asynchronous | Supported | Supported | Supported   |

### Syntax

```
Open(strScreen{ | , optNumX1{ | , optNumY1 , optNumX2 , optNumY2 , optNumResizeFlag{ | , optNumID{ | , optStrMnemonicList } } }
```

#### **strScreen**

The name of the screen (not including the .scr extension) to be opened.

 **Note:** Some Web servers are case-sensitive. If you plan to deploy your project as a Web application, then you should use only lowercase letters for the screen name.

**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. For more information, see [Screen Attributes](#).

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:

| Value | Description                                                                                                                                                                                                                                                                                                           |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0     | Screen objects will not be resized.                                                                                                                                                                                                                                                                                   |
| 1     | 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 <i>after</i> the screen is opened, then the objects will not be resized again. |

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 [Close](#) function.)

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 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. For more information, see "Examples" below.

**Returned value**

This function will return one of the following values:

| Value | Description |
|-------|-------------|
| 0     | Success     |
| 1     | Error       |

## Notes

This function cannot be used with [Tasks](#) or in the [Global Procedures](#) script.

## Examples

Open the screen "main" using the default screen size and location:

```
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), and assign it an ID of 10:

```
Open("main", 500, 250, 500, 250, 0, 10)
```

Open the screen using the default screen size and location, and replace the custom properties **Mne1** and **Mne2** with **Tag1** and **Tag2**, respectively:

```
Open("main", #1, #1, #1, #1, 1, 0, "#Mne1:Tag1 #Mne2:Tag2")
```

## OpenPrevious

OpenPrevious is a built-in scripting function that re-opens the last screen to be closed.

| Function     | Group  | Execution    | Windows   | Embedded  | Thin Client   |
|--------------|--------|--------------|-----------|-----------|---------------|
| OpenPrevious | Screen | Asynchronous | Supported | Supported | Not Supported |

## Syntax

```
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:

| Value | Description |
|-------|-------------|
| 0     | Success     |
| 1     | Error       |

## Notes

This function cannot be used with [Tasks](#) or in the [Global Procedures](#) script.

## 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

This function shows a simple text input dialog at a specified location in the project client/viewer.

| Function         | Group  | Execution    | Windows   | Embedded  | Thin Client   |
|------------------|--------|--------------|-----------|-----------|---------------|
| ShowInplaceInput | Screen | Asynchronous | Supported | Supported | Not supported |

### Syntax

```
ShowInplaceInput (" tagOutput" , numStartXPos , numStartYPos{ | , optNumMin , optNumMax{ | | , {
optNumEnablePasswordMode | 0 | 1 }{ | , { optNumShowOSVK | 0 | 1 } } })
```

#### tagOutput

The name of a tag that will receive the input.

 **Note:** The tag name must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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:

| Value | Description               |
|-------|---------------------------|
| 0     | Show input as plain text. |
| 1     | Obfuscate input.          |

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](#):

| Value | Description                   |
|-------|-------------------------------|
| 0     | Do not show Virtual Keyboard. |
| 1     | Show Virtual Keyboard.        |

This is an optional parameter. If no value is specified, then the default is 0.

### Returned value

This function returns the following possible values:

| Value | Description                          |
|-------|--------------------------------------|
| 0     | Success.                             |
| -1    | Invalid tag specified for tagOutput. |
| -2    | Invalid number of parameters.        |
| -3    | Viewer is not running.               |

## Examples

```
ShowInplaceInput("OperatorInput", 50, 50)
ShowInplaceInput("OperatorInput", 50, 50, 1, 100)
ShowInplaceInput("OperatorInput", 50, 50, 1, 100, 0, 1)
```

## ShowMessageBox

This function shows a simple message box with either an **OK** button or **Yes / No** buttons.

| Function       | Group  | Execution   | Windows   | Embedded  | Thin Client |
|----------------|--------|-------------|-----------|-----------|-------------|
| ShowMessageBox | Screen | Synchronous | Supported | Supported | Supported   |

## Syntax

```
ShowMessageBox(strMessage{ | , { optNumButtons | 0 | 4 } { | , optStrTitle } })
```

```
ShowMessageBox(strMessage)
ShowMessageBox(strMessage, optNumButtons)
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:

| Value | Description      |
|-------|------------------|
| 0     | OK button        |
| 4     | Yes / No buttons |

 **Tip:** To add an exclamation mark to the box — 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:

| Value | Description                   |
|-------|-------------------------------|
| 1     | Operator clicked <b>OK</b> .  |
| 6     | Operator clicked <b>Yes</b> . |
| 7     | Operator clicked <b>No</b> .  |

## Examples

```
ShowMessageBox("The action could not be completed.")
ShowMessageBox("Continue with action?", 4)
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.

| Function  | Group    | Execution   | Windows   | Embedded  | Thin Client |
|-----------|----------|-------------|-----------|-----------|-------------|
| BlockUser | Security | Synchronous | Supported | Supported | Supported   |

### Syntax

BlockUser ( *strUserName* )

#### **strUserName**

The name of the user to block.

### Returned value

This function returns the following possible values:

| Value | Description                                                                                                         |
|-------|---------------------------------------------------------------------------------------------------------------------|
| 0     | User blocked successfully.                                                                                          |
| 1     | Invalid number of parameters.                                                                                       |
| 2     | Wrong parameter type.                                                                                               |
| 3     | Specified user does not exist.                                                                                      |
| 4     | User currently logged on does not have the rights to block (i.e., user does not have <b>Edit Security System</b> ). |
| 5     | The operation on the distributed security system failed.                                                            |
| 6     | User cannot be blocked.                                                                                             |
| 7     | The current Security Mode does not allow user to be blocked/unblocked.                                              |
| 8     | Internal error.                                                                                                     |

### Examples

Block the user named Bob:

```
BlockUser("Bob")
```

Block the user named in position 3 of the array badUsers:

```
BlockUser(badUsers[3])
```

Block the user that is currently logged on:

```
BlockUser(UserName)
```

### **CheckESign**

CheckESign is a built-in scripting function that prompts the runtime user to electronically sign an event by entering their username and password. It can be called to secure specific expressions and scripts, just as the **E-Sign** option secures screen objects.

| Function   | Group    | Execution   | Windows   | Embedded  | Thin Client |
|------------|----------|-------------|-----------|-----------|-------------|
| CheckESign | Security | Synchronous | Supported | Supported | Supported   |

### Syntax

CheckESign()

This function takes no parameters. Calling the function displays a security dialog, where the user must enter their username and password.

## Returned value

This function returns the following possible values:

| Value | Description                            |
|-------|----------------------------------------|
| 0     | Username and/or password not accepted. |
| 1     | Username and password accepted.        |

## Notes

Usernames and passwords are stored in the [Security Folder](#).

## Examples

`CheckESign()`

## CreateUser

This function creates a new user in the project security system.

| Function   | Group    | Execution   | Windows   | Embedded  | Thin Client |
|------------|----------|-------------|-----------|-----------|-------------|
| CreateUser | Security | Synchronous | Supported | Supported | Supported   |

## Syntax

`CreateUser ( strUserName , strGroup , strPassw{ | , optStrUserFullName }`

### strUserName

The name of the user to be created.

### strGroup

The name of the group to which the user will belong.

### strPassw

The user's password.

### optStrUserFullName

The full name of the user.

## Returned value

This function returns the following possible values:

| Value | Description                                                                                                               |
|-------|---------------------------------------------------------------------------------------------------------------------------|
| -1    | Internal error; contact Technical Support.                                                                                |
| 0     | New user created successfully.                                                                                            |
| 1     | Invalid number of parameters.                                                                                             |
| 2     | Wrong parameter type.                                                                                                     |
| 3     | User name already exists.                                                                                                 |
| 4     | Group does not exist.                                                                                                     |
| 5     | Failed to save to configuration file.                                                                                     |
| 6     | Invalid user.                                                                                                             |
| 7     | User full name already exists.                                                                                            |
| 8     | Reentrant function call not allowed.                                                                                      |
| 9     | User clicked <b>Cancel</b> button when using the standard <i>Create User</i> dialog.                                      |
| 10    | Invalid password, check the minimum password size specified for the group.                                                |
| 11    | Invalid group. (Group may not have <b>Runtime group</b> option selected.)                                                 |
| 12    | Would open dialog.                                                                                                        |
| 13    | User currently logged on does not have the rights to create user (i.e., user does not have <b>Edit Security System</b> ). |

| Value | Description                                                  |
|-------|--------------------------------------------------------------|
| 14    | The current Security Mode does not allow user to be created. |

### Notes

Users created with this function are not shown in the project's *Security* folder because they are stored in a secondary database. To manage this database, use the **ExtUser.exe** program (located in the Bin sub-folder of the application folder).

### Examples

```
CreateUser("Bob", "Admin", "Chocolate", "Bob Smith")
```

```
CreateUser("Albert", "Engineering", "EMC2", "Albert Jones")
```

### ExportSecuritySystem

This function exports the security system configuration to an encrypted file.

| Function             | Group    | Execution   | Windows   | Embedded  | Thin Client |
|----------------------|----------|-------------|-----------|-----------|-------------|
| ExportSecuritySystem | Security | Synchronous | Supported | Supported | Supported   |

### 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:

| Value | Description                   |
|-------|-------------------------------|
| -2    | Wrong parameter type.         |
| -1    | Invalid number of parameters. |
| 0     | Couldn't write security data. |
| 1     | File exported successfully.   |

### Examples

```
ExportSecuritySystem("C:\security.txt")
```

```
ExportSecuritySystem("C:\security.txt", "mypa55w0rd")
```

### 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).

| Function                | Group    | Execution   | Windows   | Embedded  | Thin Client |
|-------------------------|----------|-------------|-----------|-----------|-------------|
| GetSecuritySystemStatus | Security | Synchronous | Supported | Supported | Supported   |

### Syntax

```
GetSecuritySystemStatus({ | { optNumType | 0 | 1 | 2 } }
```

```
GetSecuritySystemStatus()
```

```
GetSecuritySystemStatus(optNumType)
```

#### optNumType

The type of action to take to update the status.

| Value | Description                                                                                                   |
|-------|---------------------------------------------------------------------------------------------------------------|
| 0     | Perform a fast check using either Ping or Bind (depending on the server settings), but take no other actions. |
| 1     | Force reload of users and groups from the authentication server.                                              |
| 2     | Clear cached users and groups.                                                                                |

This is an optional parameter; if no value is specified, then the default is 0.

### Returned value

This function returns the following possible values:

| Value | Security Mode is Distributed-Client | Security Mode is Domain (LDAP)       |
|-------|-------------------------------------|--------------------------------------|
| 0     | No cache                            | Connection timeout                   |
| 1     | Updated cache                       | Bind timeout                         |
| 2     | Outdated local cache                | Query timeout                        |
| 3     | Outdated server cache               | Disconnected                         |
| 4     | Disconnected from server            | Connected                            |
| 5     | N/A                                 | No users or groups returned by query |
| 6     | N/A                                 | Invalid user or group                |

### 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](#).

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.

| Function        | Group    | Execution   | Windows   | Embedded  | Thin Client |
|-----------------|----------|-------------|-----------|-----------|-------------|
| GetUserFullName | Security | Synchronous | Supported | Supported | Supported   |

### 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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named tag.

### Returned value

This function returns the following possible values:

| Value | Description                     |
|-------|---------------------------------|
| 0     | Specified user does not exist.  |
| 1     | Success; specified user exists. |

### 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

| Function     | Group    | Execution   | Windows   | Embedded  | Thin Client        |
|--------------|----------|-------------|-----------|-----------|--------------------|
| GetUserNames | Security | Synchronous | Supported | Supported | Executed on Server |

### Syntax

```
GetUserNames("tagUsers", optNumUserType, "opttagGroups")
```

#### tagUsers

Name of the array tag that will receive users.

#### optNumUserType

|   |                                                      |
|---|------------------------------------------------------|
| 0 | Return all users                                     |
| 1 | Only users created during runtime                    |
| 2 | Only users created using the development environment |

#### 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:

|    |                                                     |
|----|-----------------------------------------------------|
| -1 | Invalid number of parameters                        |
| -2 | tagUsers is invalid                                 |
| -3 | optNumUserType is invalid                           |
| -4 | opttagGroups is invalid                             |
| -5 | Error, function cannot be called in the Thin Client |

### Examples

| Tag Name      | Expression                                                                                                                                |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| NumberOfUsers | GetUserNames( "UsersArray" ) // Retrieves all users, stores names in the UsersArray tag and the number of users in the NumberOfUsers tag. |

| Tag Name      | Expression                                                                                                                                                                                                                                                                              |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NumberOfUsers | <code>GetUserNames("UsersArray", 1)</code> // Retrieves all users created during runtime, stores names in the <b>UsersArray</b> tag and the number of users in the <b>NumberOfUsers</b> tag.                                                                                            |
| NumberOfUsers | <code>GetUserNames("UsersArray", 2)</code> // Retrieves all users created in the development environment, stores names in the <b>UsersArray</b> tag and the number of users in the <b>NumberOfUsers</b> tag.                                                                            |
| NumberOfUsers | <code>GetUserNames("UsersArray", 2, "Groups")</code> // Retrieves all users created in the development environment, stores names in the <b>UsersArray</b> tag and the number of users in the <b>NumberOfUsers</b> tag. The group name per each user is stored in the <b>Groups</b> tag. |

## GetUserPwdAging

Returns the time remaining before the password for a specific user expires.

| Function        | Group    | Execution   | Windows   | Embedded  | Thin Client |
|-----------------|----------|-------------|-----------|-----------|-------------|
| GetUserPwdAging | Security | Synchronous | Supported | Supported | Supported   |

### Syntax

`GetUserNames( strUser )`

#### strUser

User name whose password aging is checked.

### Returned value

|    |                                                   |
|----|---------------------------------------------------|
| ≤0 | Number of hours since password expired.           |
| 0  | Specified user is not logged on.                  |
| >0 | Number of hours remaining until password expires. |

 **Note:** If the function is not executed properly (e.g., User Name is invalid), or if the specified user is not logged on, then the function returns BAD quality.

### Examples

| Tag Name         | Expression                                                                                                                                   |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| TagHoursToExpire | <code>GetUserPwdAging("John")</code> // Returns the number of hours before the password for the User "John" expires.                         |
| TagHoursToExpire | <code>GetUserPwdAging(UserName)</code> // Returns the number of hours before the password for the current User logged on the system expires. |

## GetUserState

Use to see the current status of a selected user.

| Function     | Group    | Execution   | Windows   | Embedded  | Thin Client |
|--------------|----------|-------------|-----------|-----------|-------------|
| GetUserState | Security | Synchronous | Supported | Supported | Supported   |

### Syntax

`GetUserState( strUserName )`

#### strUserName

The name of the user

### Returned value

This function returns the following possible values:

| Value | Description                    |
|-------|--------------------------------|
| -3    | Specified user does not exist. |
| -2    | Wrong parameter type.          |

| Value | Description                   |
|-------|-------------------------------|
| -1    | Invalid number of parameters. |
| 0     | Specified user is unblocked.  |
| 1     | Specified user is blocked.    |

### Examples

| Tag Name | Expression                             |
|----------|----------------------------------------|
| Tag      | <code>GetUserState( "Bob" )</code>     |
| Tag      | <code>GetUserState ( "Albert" )</code> |

### ImportSecuritySystem

This function imports a security system configuration from an external file.

| Function             | Group    | Execution   | Windows   | Embedded  | Thin Client |
|----------------------|----------|-------------|-----------|-----------|-------------|
| ImportSecuritySystem | Security | Synchronous | Supported | Supported | Supported   |

### Syntax

`ImportSecuritySystem( strSecuritySystemPassword, strFileName, strFilePassword{ | , optNumMode } )`

#### **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 [ExportSecuritySystem](#) function.)

#### **strFilePassword**

The password for the specified configuration file.

#### **optNumMode**

A numeric flag indicating how the imported settings should be handled:

| Value | Description                                                                                                           |
|-------|-----------------------------------------------------------------------------------------------------------------------|
| 0     | Append the imported settings to the current settings. In the event of a conflict, replace with the imported settings. |
| 1     | Append the imported settings to the current settings. In the event of a conflict, keep the current settings.          |
| 2     | Completely replace the current settings with the imported settings.                                                   |

This parameter is optional; if no value is specified, then the default value is 0.

### Returned value

This function returns the following possible values:

| Value | Description                   |
|-------|-------------------------------|
| -2    | Wrong parameter type.         |
| -1    | Invalid number of parameters. |
| 0     | Couldn't read security data.  |
| 1     | File imported successfully.   |

### Examples

```
ExportSecuritySystem("curr3ntPa55w0rd", "C:\security.txt", "1mp0rtPa55w0rd")
```

```
ExportSecuritySystem("curr3ntPa55w0rd", "C:\security.txt", "1mp0rtPa55w0rd", 2)
```

## RemoveUser

Removes a user from the system.

| Function   | Group    | Execution   | Windows   | Embedded  | Thin Client |
|------------|----------|-------------|-----------|-----------|-------------|
| RemoveUser | Security | Synchronous | Supported | Supported | Supported   |

### Syntax

`RemoveUser( strUserName )`

#### strUserName

The name of the user to be removed.

### Returned value

This function returns the following possible values:

| Value | Description                                                                                                          |
|-------|----------------------------------------------------------------------------------------------------------------------|
| 0     | User removed successfully.                                                                                           |
| 1     | Invalid number of parameters.                                                                                        |
| 2     | Wrong parameter type.                                                                                                |
| 3     | User currently logged on does not have the rights to remove (i.e., user does not have <b>Edit Security System</b> ). |
| 4     | User cannot be removed.                                                                                              |
| 5     | Specified user does not exist.                                                                                       |
| 6     | Component-level failure.                                                                                             |
| 7     | Failed to save to configuration file.                                                                                |
| 8     | The current Security Mode does not allow user to be removed.                                                         |

### Examples

| Tag Name | Expression                          |
|----------|-------------------------------------|
| Tag      | <code>RemoveUser( "Bob" )</code>    |
| Tag      | <code>RemoveUser( "Albert" )</code> |

 **Note:** You can use this function to remove only those users you created using the `CreateUser( )` function.

## SetPassword

Use to specify a new user password.

| Function    | Group    | Execution   | Windows   | Embedded  | Thin Client |
|-------------|----------|-------------|-----------|-----------|-------------|
| SetPassword | Security | Synchronous | Supported | Supported | Supported   |

### Syntax

`SetPassword( strUserName, optStrNewPassword )`

#### strUserName

The name of the user.

#### optStrNewPassword

*Optional* The new password

 **Note:** If you omit the this parameter, then the `SetPassword` function will launch an *Enter a new password* dialog, so the user can enter a new password.

### Returned value

This function returns the following possible values:

| Value | Description                                                                                                                     |
|-------|---------------------------------------------------------------------------------------------------------------------------------|
| -1    | Internal error; contact Technical Support.                                                                                      |
| 0     | Password set successfully.                                                                                                      |
| 1     | Invalid number of parameters.                                                                                                   |
| 2     | Wrong parameter type.                                                                                                           |
| 3     | Specified user does not exist.                                                                                                  |
| 4     | Reentrant call not allowed.                                                                                                     |
| 5     | User clicked <b>Cancel</b> .                                                                                                    |
| 6     | Group does not exist.                                                                                                           |
| 7     | Password too weak.                                                                                                              |
| 8     | Invalid password.                                                                                                               |
| 9     | Invalid user.                                                                                                                   |
| 10    | User currently logged on does not have the rights to set user password (i.e., user does not have <b>Edit Security System</b> ). |
| 11    | Server offline.                                                                                                                 |
| 12    | Error.                                                                                                                          |
| 13    | Confirm password does not match.                                                                                                |
| 14    | Would open dialog.                                                                                                              |
| 15    | The current Security Mode does not allow user password to be changed.                                                           |

### Examples

| Tag Name | Expression                                       |
|----------|--------------------------------------------------|
| Tag      | <code>SetPassword( "Bob" )</code>                |
| Tag      | <code>SetPassword( "Albert", "anemarie" )</code> |

### UnblockUser

This function unblocks a blocked user in the security system.

| Function    | Group    | Execution   | Windows   | Embedded  | Thin Client |
|-------------|----------|-------------|-----------|-----------|-------------|
| UnblockUser | Security | Synchronous | Supported | Supported | Supported   |

### Syntax

`UnblockUser( strUserName )`

#### strUserName

The name of the user to unblock.

### Returned value

This function returns the following possible values:

| Value | Description                  |
|-------|------------------------------|
| 0     | User unblocked successfully. |

| Value | Description                                                                                                           |
|-------|-----------------------------------------------------------------------------------------------------------------------|
| 1     | Invalid number of parameters.                                                                                         |
| 2     | Wrong parameter type.                                                                                                 |
| 3     | Specified user does not exist.                                                                                        |
| 4     | User currently logged on does not have the rights to unblock (i.e., user does not have <b>Edit Security System</b> ). |
| 5     | The operation on the distributed security system failed.                                                              |
| 6     | Specified user cannot be blocked.                                                                                     |
| 7     | The current Security Mode does not allow user to be blocked/unblocked.                                                |
| 8     | Internal error.                                                                                                       |

### Examples

| Tag Name | Expression                           |
|----------|--------------------------------------|
| Tag      | <code>UnblockUser( "Bob" )</code>    |
| Tag      | <code>UnblockUser( "Albert" )</code> |

## 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    | Group           | Execution    | Windows   | Embedded  | Thin Client |
|-------------|-----------------|--------------|-----------|-----------|-------------|
| AppActivate | Module Activity | Asynchronous | Supported | Supported | Supported   |

### Syntax

```
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:

| Value | Command              | Description                                                                                                                                                                                                                  |
|-------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0     | SW_HIDE              | Hides the currently active window and then activates the specified window.                                                                                                                                                   |
| 1     | SW_SHOWNORMAL        | Activates and displays the specified window. If the window is minimized or maximized, then it is restored to its original size and position.<br><br>You should use this command when displaying a window for the first time. |
| 2     | SW_SHOWMINIMIZED     | Activates the specified window and then minimizes it.                                                                                                                                                                        |
| 3     | SW_SHOWMAXIMIZED     | Activates the specified window and then maximizes it.                                                                                                                                                                        |
| 4     | SW_SHOWNOACTIVATE    | Displays the specified window, but does not activate it. If the window is minimized or maximized, then it is restored to its original size and position.                                                                     |
| 5     | SW_SHOW              | 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.                           |
| 6     | SW_MINIMIZE          | Minimizes the specified window and then activates the next open window.                                                                                                                                                      |
| 7     | SW_SHOWMINNOACTIVATE | Displays the specified window as a minimized window, but does not activate it.                                                                                                                                               |
| 8     | SW_SHOWNA            | 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                                                                  |

| Value | Command           | Description                                                                                                                                                                                                        |
|-------|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|       |                   | minimized or maximized, then it remains in that state.                                                                                                                                                             |
| 9     | <b>SW_RESTORE</b> | Activates and displays the specified window. If the window is minimized or maximized, then it is restored to its original size and position.<br><br>You should use this command when restoring a minimized window. |

This is an optional parameter. If no value is specified, then the default command is **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:

| Value | Description                                                                                                       |
|-------|-------------------------------------------------------------------------------------------------------------------|
| 0     | ERROR: The specified application window was not activated or otherwise did not respond within the timeout period. |
| 1     | SUCCESS: The specified application window was successfully activated.                                             |

#### **Notes**

`AppActivate` is similar to the function `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 `test.doc`:

```
AppActivate("test.doc - Microsoft Word", 5)
```

#### **AppIsRunning**

`AppIsRunning` is a built-in scripting function that verifies another application window is open and running.

| Function                  | Group           | Execution   | Windows   | Embedded  | Thin Client |
|---------------------------|-----------------|-------------|-----------|-----------|-------------|
| <code>AppIsRunning</code> | Module Activity | Synchronous | Supported | Supported | Supported   |

#### **Syntax**

```
AppIsRunning(strAppTitle{ | , optNumTimeout })
```

##### **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:

| Value | Description                                                                                                 |
|-------|-------------------------------------------------------------------------------------------------------------|
| 0     | ERROR: The specified application window is not open or otherwise did not respond within the timeout period. |
| 1     | SUCCESS: The specified application window is open and running.                                              |

### 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:

```
AppIsRunning("test.doc - Microsoft Word")
```

### AppPostMessage

`AppPostMessage` is a built-in scripting function that sends a Windows system message to another application window.

| Function                    | Group           | Execution   | Windows   | Embedded  | Thin Client |
|-----------------------------|-----------------|-------------|-----------|-----------|-------------|
| <code>AppPostMessage</code> | Module Activity | Synchronous | Supported | Supported | Supported   |

### Syntax

```
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.

 **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:

| Value | Description                                                                                                             |
|-------|-------------------------------------------------------------------------------------------------------------------------|
| 0     | ERROR: The system message was not sent, or the specified application window did not respond, within the timeout period. |
| 1     | SUCCESS: The system message was successfully sent.                                                                      |

### 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:

```
AppPostMessage("Calculator", "CLOSE", 3, 1)
```

## AppSendKeys

Sends keyboard commands to the active application.

| Function    | Group           | Execution   | Windows   | Embedded      | Thin Client |
|-------------|-----------------|-------------|-----------|---------------|-------------|
| AppSendKeys | Module Activity | Synchronous | Supported | Not supported | Supported   |

## Syntax

```
AppSendKeys(strKeys1, strKeys2, ... , strKeysN)
```

### strKeys (1-N)

String tags containing the keyboard commands to be used.

## Returned value

No returned values.

## Examples

| Tag Name | Expression                                             |
|----------|--------------------------------------------------------|
|          | AppSendKeys( "S", "t", "u", "d", "i", "o", "<ENTER>" ) |
|          | AppSendKeys( "<Alt>F" )                                |

 **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.

| Function       | Group           | Execution   | Windows   | Embedded      | Thin Client        |
|----------------|-----------------|-------------|-----------|---------------|--------------------|
| CleanReadQueue | Module Activity | Synchronous | Supported | Not supported | Executed on Server |

## Syntax

```
CleanReadQueue()
```

This function takes no parameters.

## Returned value

No returned values.

## Examples

| Tag Name | Expression        |
|----------|-------------------|
|          | CleanReadQueue( ) |

 **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.

| Function          | Group           | Execution   | Windows   | Embedded      | Thin Client        |
|-------------------|-----------------|-------------|-----------|---------------|--------------------|
| CloseSplashWindow | Module Activity | Synchronous | Supported | Not supported | Executed on Server |

## Syntax

CloseSplashWindow()

This function takes no parameters.

## Returned value

No returned values.

## Examples

| Tag Name | Expression          |
|----------|---------------------|
|          | CloseSplashWindow() |

## DisableMath

DisableMath is a built-in scripting function that pauses the execution of all Math worksheets.

| Function    | Group           | Execution    | Windows   | Embedded  | Thin Client        |
|-------------|-----------------|--------------|-----------|-----------|--------------------|
| DisableMath | Module Activity | Asynchronous | Supported | Supported | Executed on Server |

## 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

| Tag Name | Expression    |
|----------|---------------|
|          | DisableMath() |

## EnableMath

EnableMath is a built-in scripting function that resumes the execution of all Math worksheets.

| Function   | Group           | Execution    | Windows   | Embedded  | Thin Client        |
|------------|-----------------|--------------|-----------|-----------|--------------------|
| EnableMath | Module Activity | Asynchronous | Supported | Supported | Executed on Server |

## 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

| Tag Name | Expression   |
|----------|--------------|
|          | EnableMath() |

## EndTask

Stops the IWS module that is currently running.

| Function | Group           | Execution    | Windows   | Embedded  | Thin Client        |
|----------|-----------------|--------------|-----------|-----------|--------------------|
| EndTask  | Module Activity | Asynchronous | Supported | Supported | Executed on Server |

## Syntax

EndTask( *strTask* )

### strTask

The name of the task to stop (must be one of the following):

|                          |                  |
|--------------------------|------------------|
| BGTASK                   | Background Tasks |
| VIEWER                   | Viewer           |
| DBSPY                    | Database Spy     |
| LOGWIN                   | LogWin           |
| DRIVER <i>DriverName</i> | Driver           |
| UNIDDECL                 | DDE client       |
| UNINDEE                  | DDE server       |
| UNIODBC                  | ODBC             |
| TCPSERVER                | TCP/IP Server    |
| TCPCLIENT                | TCP/IP Client    |
| OPCCLIENT                | OPC              |

## Returned value

No returned values.

## Examples

| Tag Name | Expression                       |
|----------|----------------------------------|
|          | <code>EndTask( "Viewer" )</code> |

 **Note:** To close a driver, you must use the following syntax:

```
EndTask("DriverDriverName")
```

Where *DriverName* is the name of the driver's .dll file. For example:

```
EndTask("DriverMODBU")
```

## ExitWindows

Exits the Windows operating system in a specified manner.

| Function    | Group           | Execution    | Windows   | Embedded      | Thin Client |
|-------------|-----------------|--------------|-----------|---------------|-------------|
| ExitWindows | Module Activity | Asynchronous | Supported | Not supported | Supported   |

## Syntax

ExitWindows( *numExitCode* )

### numExitCode

A numeric code specifying how Windows should be exited:

| Value | Description |
|-------|-------------|
| 0     | Restart     |
| 1     | Log off     |

| Value | Description |
|-------|-------------|
| 2     | Shut down   |

### Returned value

No returned values.

### Examples

| Tag Name | Expression                    |
|----------|-------------------------------|
|          | <code>ExitWindows( 1 )</code> |

### IsScreenOpen

Verifies that a project screen is open.

| Function     | Group           | Execution    | Windows   | Embedded  | Thin Client   |
|--------------|-----------------|--------------|-----------|-----------|---------------|
| IsScreenOpen | Module Activity | Asynchronous | Supported | Supported | Not supported |

 **Note:** This function cannot be used with [Tasks](#) or in the [Global Procedures](#) script.

### Syntax

`IsScreenOpen( strScreen{ | , optNumID } )`

#### strScreen

The name of the project screen to be verified.

#### optNumID

The specific instance number of the screen. (This number is assigned when the screen is opened with the [Open](#) function.)

This is an optional parameter. If no value is specified, then the default ID is 0.

### Returned value

|   |                     |
|---|---------------------|
| 0 | Screen is not open. |
| 1 | Screen is open.     |

### Examples

| Tag Name | Expression                                                                   |
|----------|------------------------------------------------------------------------------|
| Tag      | <code>IsScreenOpen( "main" ) // Is "main" screen open?</code>                |
| Tag      | <code>IsScreenOpen( "main", 10 ) // Is "main" screen with ID 10 open?</code> |

### IsTaskRunning

Verifies that a background task is running.

| Function      | Group           | Execution   | Windows   | Embedded  | Thin Client        |
|---------------|-----------------|-------------|-----------|-----------|--------------------|
| IsTaskRunning | Module Activity | Synchronous | Supported | Supported | Executed on Server |

### Syntax

`IsTaskRunning( strTask )`

#### strTask

The name of the task to be verified.

### Returned value

|   |                      |
|---|----------------------|
| 0 | Task is not running. |
| 1 | Task is running.     |

## Examples

Is the project viewer (Viewer.exe) running?

```
IsTaskRunning("Viewer")
```

## IsViewerInFocus

Verifies that the project viewer (Viewer.exe) is in focus on the display.

| Function        | Group           | Execution   | Windows   | Embedded      | Thin Client   |
|-----------------|-----------------|-------------|-----------|---------------|---------------|
| IsViewerInFocus | Module Activity | Synchronous | Supported | Not supported | Not supported |

## Syntax

```
IsViewerInFocus()
```

This function takes no parameters.

## Returned value

|   |                         |
|---|-------------------------|
| 0 | Viewer is not in focus. |
| 1 | Viewer is in focus.     |

## Examples

```
IsViewerInFocus()
```

## KeyPad

Displays a Virtual Keyboard to prompt the runtime user to enter a tag value.

| Function | Group           | Execution    | Windows   | Embedded  | Thin Client |
|----------|-----------------|--------------|-----------|-----------|-------------|
| KeyPad   | Module Activity | Asynchronous | Supported | Supported | Supported   |

## Syntax

```
KeyPad("strTagName", optStrKeyboardName, optNumIsPassword, optStrHint, optNumMin, optNumMax, optNumESign, optStrConfirm)
```

### 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

| Error | Description                                                    |
|-------|----------------------------------------------------------------|
| 0     | Success                                                        |
| 1     | Error                                                          |
| 2     | Tag does not exist                                             |
| 3     | Reentrant error, function is already executing                 |
| 4     | Invalid number of parameters                                   |
| 5     | Internal error, contact Technical Support for more information |

### Examples

| Tag Name | Expression                                           |
|----------|------------------------------------------------------|
| Tag      | KeyPad( "tagA" )                                     |
| Tag      | KeyPad( "tagA", "EnhKeypad" )                        |
| Tag      | KeyPad( "tagA", "EnhKeypad", 1 )                     |
| Tag      | KeyPad( "tagA", "EnhKeypad", 1, "My Input", 0, 100 ) |

### LogOff

This function logs off the current user and then logs on the default user (typically "guest").

| Function | Group           | Execution    | Windows   | Embedded  | Thin Client |
|----------|-----------------|--------------|-----------|-----------|-------------|
| LogOff   | Module Activity | Asynchronous | Supported | Supported | Supported   |

### Syntax

LogOff()

This function takes no parameters.

### Returned value

This function returns the following possible values:

| Value | Description |
|-------|-------------|
| 0     | Error.      |
| 1     | Success.    |

### Examples

LogOff()

## LogOn

This function either logs on a specified user or displays a *Log On* dialog.

| Function | Group           | Execution    | Windows   | Embedded  | Thin Client |
|----------|-----------------|--------------|-----------|-----------|-------------|
| LogOn    | Module Activity | Asynchronous | Supported | Supported | Supported   |

### Syntax

```
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're 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:

| Value | Description                                                    |
|-------|----------------------------------------------------------------|
| 0     | Error (e.g., username or password is invalid) or cancellation. |
| 1     | Success.                                                       |

### Examples

Display a Log On dialog:

```
LogOn()
```

Log on username Albert with password EMC2:

```
LogOn("Albert", "EMC2")
```

## Math

Executes the specified Math worksheet.

| Function | Group           | Execution   | Windows   | Embedded  | Thin Client        |
|----------|-----------------|-------------|-----------|-----------|--------------------|
| Math     | Module Activity | Synchronous | Supported | Supported | Executed on Server |

| Group                           | Execution   | Windows PC | Windows CE | Thin Client   |
|---------------------------------|-------------|------------|------------|---------------|
| <a href="#">Module Activity</a> | Synchronous | Supported  | Supported  | Not supported |

### Syntax

```
Math(numWorksheet)
```

#### numWorksheet

The number of the math worksheet to be executed.

### Returned value

No returned values.

### Examples

| Tag Name | Expression       |
|----------|------------------|
|          | <b>Math( 6 )</b> |



**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

Posts key codes to the currently displayed project screen.

| Function | Group           | Execution   | Windows   | Embedded  | Thin Client |
|----------|-----------------|-------------|-----------|-----------|-------------|
| PostKey  | Module Activity | Synchronous | Supported | Supported | Supported   |



**Note:** This function cannot be used with [Tasks](#) or in the [Global Procedures](#) script.

## Syntax

`PostKey( numKeyDownOrKeyUp, numwParam, numlParam )`

### numKeyDownOrKeyUp

Numerical tag containing a 0 (to indicate a KeyDown event) or a 1 (to indicate a KeyUp event).

### numwParam

Numerical tag containing key code to be sent.

### numlParam

Numerical tag containing message `lParam`.

## Returned value

No returned values.

## Examples

| Tag Name | Expression                         |
|----------|------------------------------------|
|          | <code>PostKey( 0, 0x24, 0 )</code> |

## Recipe

Executes the specified Recipe worksheet.

| Function | Group           | Execution   | Windows   | Embedded  | Thin Client |
|----------|-----------------|-------------|-----------|-----------|-------------|
| Recipe   | Module Activity | Synchronous | Supported | Supported | Supported   |

## 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:

|        |                                                            |
|--------|------------------------------------------------------------|
| Save   | Saves data to a data file.                                 |
| Load   | Loads data from a data file.                               |
| Delete | Deletes a data file.                                       |
| Init   | Initializes a data file with a value of 0 in all the tags. |

## Returned value

|   |          |
|---|----------|
| 0 | No error |
|---|----------|

|   |                                     |
|---|-------------------------------------|
| 1 | If the tag is numeric               |
| 2 | Expression does not contain ":"     |
| 3 | Previous command to the invalid ":" |
| 4 | Task not found by the system        |
| 5 | Disk error                          |

## Examples

| Tag Name | Expression                            |
|----------|---------------------------------------|
| Tag      | <code>Recipe( "Save:Recipe1" )</code> |
| Tag      | <code>Recipe( "Load:Recipe5" )</code> |

### 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.

| Function | Group           | Execution   | Windows   | Embedded  | Thin Client |
|----------|-----------------|-------------|-----------|-----------|-------------|
| Report   | Module Activity | Synchronous | Supported | Supported | Supported   |

## Syntax

`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

| Value | Description                                                                         |
|-------|-------------------------------------------------------------------------------------|
| 0     | Success                                                                             |
| 1     | <code>strFunction</code> is configured with a numeric value (invalid)               |
| 2     | <code>strFunction</code> does not contain ":" (invalid)                             |
| 3     | <code>strFunction</code> contains an invalid output type before the ":"             |
| 4     | Background Task is not running (see Tip below)                                      |
| 5     | Disk error (e.g., disk full, read-only file cannot be overwritten, or invalid path) |
| 6     | Specified Report worksheet does not exist                                           |

 **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

| Tag Name | Expression                                  |
|----------|---------------------------------------------|
|          | <code>Report( "Disk:Report1.rep" )</code>   |
|          | <code>Report( "Prn:Report2.rep", 0 )</code> |
|          | <code>Report( "Prn:Report3.rep", 1 )</code> |
|          | <code>Report( "Pdf:Report1.rep" )</code>    |

## 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.

| Function                | Group           | Execution    | Windows   | Embedded  | Thin Client             |
|-------------------------|-----------------|--------------|-----------|-----------|-------------------------|
| RunGlobalProcedureAsync | Module Activity | Asynchronous | Supported | Supported | Not supported; see note |

## Syntax

`RunGlobalProcedureAsync( strProcedureName{ | , optStrArgument1 , ... , optStrArgumentN } )`

`RunGlobalProcedureAsync( strProcedureName )`

`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:

| Value | Description                                                                 |
|-------|-----------------------------------------------------------------------------|
| -1    | Function is not supported on Viewer / Web Thin Client.                      |
| -2    | Invalid number of parameters. You must specify at least the procedure name. |
| -3    | Maximum number of threads exceeded. See note.                               |
| -4    | Failed to compile VBScript parameters for execution.                        |
| -5    | Failed to start the thread execution.                                       |
| -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...

```
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...

```
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.

| Function                         | Group           | Execution   | Windows   | Embedded  | Thin Client             |
|----------------------------------|-----------------|-------------|-----------|-----------|-------------------------|
| RunGlobalProcedureAsyncGetStatus | Module Activity | Synchronous | Supported | Supported | Not supported; see note |

### Syntax

```
RunGlobalProcedureAsyncGetStatus({ | optNumThreadID |
"optTagThreadIDs", "optTagStatus", "optTagParameters" })
```

```
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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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 | Description                                            |
|-------|--------------------------------------------------------|
| -1    | Function is not supported on Viewer / Web Thin Client. |
| -2    | Invalid thread ID.                                     |
| -3    | Invalid optTagThreadIDs.                               |
| -4    | Invalid optTagStatus.                                  |
| -5    | Invalid optTagParameters.                              |
| -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, 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 directly executes a global procedure when the value of a specified tag/expression becomes FALSE.

| Function                  | Group           | Execution   | Windows   | Embedded  | Thin Client |
|---------------------------|-----------------|-------------|-----------|-----------|-------------|
| RunGlobalProcedureOnFalse | Monitor/Control | Synchronous | Supported | Supported | Supported   |

**Syntax**

```
RunGlobalProcedureOnFalse("strCondition" , strProcedureOnFalse)
```

```
RunGlobalProcedureOnFalse("strCondition", strProcedureOnFalse)
```

**strCondition**

A project tag or expression.

 **Note:** The tag name must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named tag.

**strProcedureOnFalse**

The name of the procedure (i.e., a VBScript function or sub-routine defined in the [Procedures](#) folder) to run when the value of strCondition becomes FALSE (zero).

**Returned value**

This function returns the following possible values:

| Value | Description |
|-------|-------------|
| 0     | Error       |
| 1     | Success     |

**Notes**

Once this function is called, it remains active until the runtime project is stopped. That means every time the tag/expression becomes FALSE, the procedure is executed. However, the procedure is executed only once *when* the tag/expression becomes FALSE; it is not continuously executed *while* the tag/expression is FALSE.

The procedure is executed on the Client. To execute a procedure on the Server, use the [RunGlobalProcedureOnServer](#) function.

**Examples**

When the value of **TagOnFalse** becomes FALSE (zero), execute the procedure **UsingOnFalse**:

```
RunGlobalProcedureOnFalse("TagOnFalse", "UsingOnFalse")
```

**RunGlobalProcedureOnServer**

This function directly executes a global procedure with parameters. The procedure is run on the project server, but it can be called by any local or remote client.

| Function                   | Group            | Execution   | Windows   | Embedded  | Thin Client |
|----------------------------|------------------|-------------|-----------|-----------|-------------|
| RunGlobalProcedureOnServer | Module On Server | Synchronous | Supported | Supported | Supported   |

**Syntax**

```
RunGlobalProcedureOnServer(strProcedureName{ | , optStrArgument1 , ... , optStrArgumentN })
```

```
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 server.

**optStrArgument1, ..., optStrArgumentN**

Values that are passed to the procedure's parameters. Arguments must be passed as strings.

**Returned value**

This function returns whatever value is returned by the called procedure.

## Examples

Given the following procedure that is defined in the Procedures folder...

```
Function AddMe(intNumber)
 If intNumber >= 6 Then
 AddMe = 0
 Else
 AddMe = intNumber + 2
 End If
End Function
```

...the procedure is run by calling the RunGlobalProcedureOnServer function...

```
RunGlobalProcedureOnServer("AddMe", "2")
```

...and it returns a value of 4.

## RunGlobalProcedureOnTrigger

This function directly executes a global procedure when the value of a specified tag changes.

| Function                    | Group   | Execution   | Windows   | Embedded  | Thin Client |
|-----------------------------|---------|-------------|-----------|-----------|-------------|
| RunGlobalProcedureOnTrigger | Monitor | Synchronous | Supported | Supported | Supported   |

## Syntax

```
RunGlobalProcedureOnTrigger("strTagName" , strProcedureOnTrigger)
```

```
RunGlobalProcedureOnTrigger("strTagName", strProcedureOnTrigger)
```

### strTagName

The name of a project tag.



**Note:** The tag name must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named tag.

### strProcedureOnTrigger

The name of the procedure (i.e., a VBScript function or sub-routine defined in the [Procedures](#) folder) to run when the value of the specified tag changes.

## Returned value

This function returns the following possible values:

| Value | Description |
|-------|-------------|
| 0     | Error       |
| 1     | Success     |

## Notes

Once this function is called, it remains active until the runtime project is stopped. That means every time the value of the tag changes, the procedure is executed.

The procedure is executed on the Client. To execute a procedure on the Server, use the [RunGlobalProcedureOnServer](#) function.

## Examples

When the value of **TagTrigger** changes, execute the procedure **UsingTrigger**:

```
RunGlobalProcedureOnTrigger("TagTrigger", "UsingTrigger")
```

**RunGlobalProcedureOnTrue**

This function directly executes a global procedure when the value of a specified tag/expression becomes TRUE.

| Function                 | Group           | Execution   | Windows   | Embedded  | Thin Client |
|--------------------------|-----------------|-------------|-----------|-----------|-------------|
| RunGlobalProcedureOnTrue | Module Activity | Synchronous | Supported | Supported | Supported   |

**Syntax**

```
RunGlobalProcedureOnTrue("strCondition" , strProcedureOnTrue)
```

```
RunGlobalProcedureOnTrue("strCondition" , strProcedureOnTrue)
```

**strCondition**

A project tag or expression.

 **Note:** The tag name must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named tag.

**strProcedureOnTrue**

The name of the procedure (i.e., a VBScript function or sub-routine defined in the [Procedures](#) folder) to run when the value of strCondition becomes TRUE (non-zero).

**Returned value**

This function returns the following possible values:

| Value | Description |
|-------|-------------|
| 0     | Error       |
| 1     | Success     |

**Notes**

Once this function is called, it remains active until the runtime project is stopped. That means every time the tag/expression becomes TRUE, the procedure is executed. However, the procedure is executed only once *when* the tag/expression becomes TRUE; it is not continuously executed *while* the tag/expression is TRUE.

The procedure is executed on the Client. To execute a procedure on the Server, use the [RunGlobalProcedureOnServer](#) function.

**Examples**

When the value of **TagOnTrue** becomes TRUE (non-zero), execute the procedure **UsingOnTrue**:

```
RunGlobalProcedureOnTrue("TagOnTrue" , "UsingOnTrue")
```

**RunVBScript**

Executes a statement in VBScript language.

| Function    | Group           | Execution   | Windows   | Embedded  | Thin Client |
|-------------|-----------------|-------------|-----------|-----------|-------------|
| RunVBScript | Module Activity | Synchronous | Supported | Supported | Supported   |

**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

|   |         |
|---|---------|
| 0 | Error   |
| 1 | Success |

## Examples

| Tag Name         | Expression                                                                                                                                                                                                                                                          |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>TagResult</b> | <code>RunVBScript ( "MsgBox (Time) " )</code> // Executes the <a href="#">MsgBox function</a> from VBScript and displays the current time.                                                                                                                          |
|                  | <code>RunVBScript ( TagStatement )</code> // Executes the statement configured in the value of the string tag <b>TagStatement</b> .                                                                                                                                 |
|                  | <code>RunVBScript ( "\$TagC=\$TagA/\$TagB" , "TagError" )</code> // Writes in <b>TagC</b> the result of <b>TagA</b> divided by <b>TagB</b> . The error generated by the operation (if any) is written to the string tag <b>TagError</b> (e.g., "Division by zero"). |

 **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.

| Function           | Group           | Execution   | Windows       | Embedded      | Thin Client        |
|--------------------|-----------------|-------------|---------------|---------------|--------------------|
| SecureViewerReload | Module Activity | Synchronous | Not Supported | Not Supported | Secure Viewer only |

## 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

Sends key event codes to objects in the currently displayed project screen. You can use this function to trigger Command animations on these objects.

| Function      | Group           | Execution   | Windows   | Embedded      | Thin Client   |
|---------------|-----------------|-------------|-----------|---------------|---------------|
| SendKeyObject | Module Activity | Synchronous | Supported | Not supported | Not supported |

 **Note:** This function cannot be used with [Tasks](#) or in the [Global Procedures](#) script.

## Syntax

```
SendKeyObject(numEvent, strMainKey, numShift, numCtrl, numAlt, strTargetScreen, optNumID)
```

### numEvent

The event code for "On Down", "On While" or "On Up" of the [Command](#) animation, as follows:

|   |            |
|---|------------|
| 0 | On Down    |
| 1 | While Down |
| 2 | On Up      |

 **Note:** The "On While" event requires special attention. Each time the `SendKeyObject()` function is executed, IWS executes the expressions configured for the "On While" sheet (from the object's [Command](#) animation) just once.

### strMainKey

The key to be sent to the screen. The following values are accepted:

- "F1" ... "F20"
- "+"
- "-"
- "/"
- "\*"
- "HOME"
- "END"
- "INSERT"
- "DELETE"
- "DOWN"
- "UP"
- "LEFT"
- "RIGHT"
- "PAGEUP"
- "PAGEDOWN"
- "SPACE"
- "RETURN"
- "BACKSPACE"
- "ESCAPE"
- "A" ... "Z"

### numShift

A numeric value or tag; whether to include Shift with the key (0 is no, 1 is yes).

### numCtrl

A numeric value or tag; whether to include Ctrl with the key (0 is no, 1 is yes).

### numAlt

A numeric value or tag; whether to include Alt with the key (0 is no, 1 is yes).

### strTargetScreen

The name of the screen to receive the key event code.

 **Note:** The `numShift`, `numCtrl`, `numAlt` and `strTargetScreen` parameters are optional. However, if you configure one of them, then you must configure the others as well.

### optNumID

The specific instance number of the screen. (The ID is assigned when the screen is opened with the `Open()` function.) This is an optional parameter; the default ID is 0.

### Returned value

This function returns no values.

### Examples

| Tag Name | Expression                                                                                                  |
|----------|-------------------------------------------------------------------------------------------------------------|
|          | <code>SendKeyObject( 0, "R", 1, 0, 0, "main", 10 )</code> // Sends Shift-R to the "main" screen with ID 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.

| Function   | Group           | Execution   | Windows   | Embedded      | Thin Client        |
|------------|-----------------|-------------|-----------|---------------|--------------------|
| SetAppPath | Module Activity | Synchronous | Supported | Not supported | Executed on Server |

### Syntax

SetAppPath(*strPath*)

#### strPath

The file path.

### Returned value

|   |                           |
|---|---------------------------|
| 0 | Failed to set path.       |
| 1 | Succeeded in setting path |

### Examples

| Tag Name | Expression                              |
|----------|-----------------------------------------|
|          | <code>SetAppPath( "C:\Studio\" )</code> |

 **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.

| Function         | Group           | Execution   | Windows   | Embedded      | Thin Client |
|------------------|-----------------|-------------|-----------|---------------|-------------|
| SetViewerInFocus | Module Activity | Synchronous | Supported | Not Supported | Supported   |

### 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.

| Function     | Group           | Execution   | Windows   | Embedded      | Thin Client |
|--------------|-----------------|-------------|-----------|---------------|-------------|
| SetViewerPos | Module Activity | Synchronous | Supported | Not supported | Supported   |

## 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

|   |         |
|---|---------|
| 0 | Error   |
| 1 | Success |

## Examples

| Tag Name | Expression                                    |
|----------|-----------------------------------------------|
| Tag      | <code>SetViewerPos( 50, 50, 640, 480 )</code> |

 **Note:** When you omit the optional parameters (`numWidth` and `numHeight`), IWS gets size of the Viewer window from the project resolution.

## ShutDown

Shuts down all of the active project modules.

| Function | Group           | Execution    | Windows   | Embedded  | Thin Client   |
|----------|-----------------|--------------|-----------|-----------|---------------|
| ShutDown | Module Activity | Asynchronous | Supported | Supported | Not supported |

## Syntax

```
ShutDown()
```

This function takes no parameters.

## Returned value

No returned values.

## Notes

This function does not close the [development environment](#), [Database Spy](#), or [LogWin](#).

## Examples

| Tag Name | Expression |
|----------|------------|
|          | ShutDown() |

## StartTask

Starts a project module that is not currently running.

| Function  | Group           | Execution    | Windows   | Embedded  | Thin Client        |
|-----------|-----------------|--------------|-----------|-----------|--------------------|
| StartTask | Module Activity | Asynchronous | Supported | Supported | Executed on Server |

## Syntax

StartTask(*strTask*)

### strTask

The name of the task to start (must be one of the following tasks):

| Value     | Description      |
|-----------|------------------|
| BGTask    | Background Tasks |
| Viewer    | Viewer           |
| DBSpy     | Database Spy     |
| LogWin    | LogWin           |
| Driver    | Driver           |
| UnidDECl  | DDE Client       |
| UnidNDE   | DDE Server       |
| UnidODBC  | ODBC             |
| TCPServer | TCP/IP Server    |
| TCPClient | TCP/IP Client    |
| OPCClient | OPC Client       |

## Returned value

No returned values.

## Examples

| Tag Name                                                                                                                                                                                                                                                                  | Expression            |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
|                                                                                                                                                                                                                                                                           | StartTask( "Viewer" ) |
| <p> <b>Note:</b> The StartTask( "Driver" ) function starts all drivers configured in the project. To start a specific driver, you must use the <a href="#">WinExec()</a> function.</p> |                       |

## ViewerPostMessage

Posts a Windows System Message to the specified project screen.

| Function          | Group           | Execution    | Windows   | Embedded  | Thin Client |
|-------------------|-----------------|--------------|-----------|-----------|-------------|
| ViewerPostMessage | Module Activity | Asynchronous | Supported | Supported | Supported   |

|                                                                                                                                                                                                                     |  |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| <p> <b>Note:</b> This function cannot be used with <a href="#">Tasks</a> or in the <a href="#">Global Procedures</a> script.</p> |  |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|

## 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

A numeric value or tag; additional message-specific information, which is passed to `wParam` of the Windows System Message.

### numlParam

A numeric value or tag; additional message-specific information, which is passed to `lParam` of the Windows System Message.

### optNumID

The specific instance number of the screen. (The ID is assigned when the screen is opened with the `Open()` function.) This is an optional parameter; the default ID is 0.

## Returned value

This function returns no values.

## Notes

This function emulates the `PostMessage` function in Microsoft Windows. For more information, including a complete list of available Windows System Messages, please consult [Microsoft Developers Network](#).

## Examples

| Tag Name | Expression                                                                                                |
|----------|-----------------------------------------------------------------------------------------------------------|
|          | <pre>ViewerPostMessage( "main", 16, 3, 1, 10 ) // Sends message 16 to the "main" screen with ID 10.</pre> |

## WinExec

`WinExec` is a built-in scripting function that executes a Windows command as if it was entered at the command prompt.

| Function | Group           | Execution    | Windows   | Embedded              | Thin Client |
|----------|-----------------|--------------|-----------|-----------------------|-------------|
| WinExec  | Module Activity | Asynchronous | Supported | Supported (see notes) | Supported   |

## 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:

| Value | Description                                                        |
|-------|--------------------------------------------------------------------|
| 0     | Hides the program and gives control to another one.                |
| 1     | Activates and displays the program.                                |
| 2     | Activates the program and displays it as an icon.                  |
| 3     | Activates the program and maximizes it.                            |
| 4     | Shows the program at its recent size. The program is still active. |

| Value | Description                                                |
|-------|------------------------------------------------------------|
| 7     | Shows the program as an icon. The program is still active. |

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:

| Value | Description                                                                              |
|-------|------------------------------------------------------------------------------------------|
| 0     | Execute asynchronously; the function will return immediately.                            |
| 1     | Execute synchronously; the function will return when the command has finished executing. |

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 be 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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named tag.

### Returned value

This function returns the following possible values:

| Value | Description                            |
|-------|----------------------------------------|
| 0     | Command was not executed successfully. |
| 1     | Command was executed successfully.     |

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:

```
WinExec("C:\Windows\System32\notepad.exe", 4)
```

Start MS Paint, and then immediately continue to the next line of the script:

```
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`:

```
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**:

```
WinExec("CMD /C call C:\Temp\MyScript.vbs", 0, 0, "handle")
```

 **Note:** Calling VBScript files is not supported on Windows Embedded target systems.

### WinExecIsRunning

Indicates whether a program started by the WinExec function is still running.

| Function         | Group           | Execution   | Windows   | Embedded  | Thin Client |
|------------------|-----------------|-------------|-----------|-----------|-------------|
| WinExecIsRunning | Module Activity | Synchronous | Supported | Supported | Supported   |

### Syntax

```
WinExecIsRunning(numHandle, "optTagReturn")
```

#### numHandle

Handle number stored in the tag in the optStrReturnorHandle parameter of the WinExec function.

#### "optTagReturn"

Tag that receives the code returned by the program executed by the WinExec function.

### Returned value

|    |                                                                       |
|----|-----------------------------------------------------------------------|
| 0  | Successful execution.                                                 |
| -1 | Invalid parameter(s).                                                 |
| -2 | Failed to open file. Disk is write protected or file name is invalid. |

### Examples

| Tag Name | Expression                              |
|----------|-----------------------------------------|
| Tag      | WinExecIsRunning( numHandle )           |
| Tag      | WinExecIsRunning( numHandle, "return" ) |

## 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.

| Function         | Group | Execution   | Windows   | Embedded  | Thin Client |
|------------------|-------|-------------|-----------|-----------|-------------|
| DeleteOlderFiles | File  | Synchronous | Supported | Supported | Supported   |

### 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.

| Function  | Group | Execution   | Windows   | Embedded  | Thin Client        |
|-----------|-------|-------------|-----------|-----------|--------------------|
| DirCreate | File  | Synchronous | Supported | Supported | Executed on Server |

### 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

|    |                                                              |
|----|--------------------------------------------------------------|
| -1 | Invalid parameters                                           |
| 0  | Failed to create the directory (e.g., Drive does not exist.) |

|   |                                 |
|---|---------------------------------|
| 1 | Directory created successfully. |
|---|---------------------------------|

### Examples

| Tag Name | Expression                                                                                                                                  |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>DirCreate( "C:\Studio\Temp" )</code> // The Temp folder is created in the C:\Studio path (only if the C:\Studio path already exists). |
| Tag      | <code>DirCreate( "C:\Studio\Temp" , 1 )</code> // The C:\Studio\Temp full path is created.                                                  |

### DirDelete

Deletes the specified directory.

| Function  | Group | Execution   | Windows   | Embedded  | Thin Client        |
|-----------|-------|-------------|-----------|-----------|--------------------|
| DirDelete | File  | Synchronous | Supported | Supported | Executed on Server |

### 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

| Value | Description                                                                                                                      |
|-------|----------------------------------------------------------------------------------------------------------------------------------|
| #2    | Attempted to delete a non-empty directory when this action is not allowed (i.e., <code>optBooEmptyOnly</code> does not equal 0). |
| #1    | Invalid parameters.                                                                                                              |
| 0     | Failed to delete the directory (i.e., directory does not exist).                                                                 |
| 1     | Directory deleted successfully.                                                                                                  |

### Examples

| Tag Name | Expression                                                                                                      |
|----------|-----------------------------------------------------------------------------------------------------------------|
| Tag      | <code>DirDelete( "C:\Studio\Temp" )</code> // The Temp folder from C:\Studio is deleted.                        |
| Tag      | <code>DirDelete( "C:\Studio\Temp", 1 )</code> // The Temp folder from C:\Studio is deleted only if it is empty. |

### DirLength

Returns the size of a specific directory.

| Function  | Group | Execution   | Windows   | Embedded  | Thin Client        |
|-----------|-------|-------------|-----------|-----------|--------------------|
| DirLength | File  | Synchronous | Supported | Supported | Executed on Server |

### Description

Returns the size of a specific directory.

### Syntax

`DirLength( strPath )`

#### strPath

The path of the directory that will be checked.

### Returned value

|          |                                                               |
|----------|---------------------------------------------------------------|
| -2       | Directory does not exist.                                     |
| -1       | Invalid parameters                                            |
| <i>n</i> | Size (in bytes) of the files and sub-folders in the directory |

### 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 Name | Expression                                                                                                      |
|----------|-----------------------------------------------------------------------------------------------------------------|
| Tag      | <code>DirLength("C:\Studio")</code> // Returns the size (in bytes) of all files and sub-folders from C:\Studio. |

### DirRename

Renames directories.

| Function  | Group | Execution   | Windows   | Embedded  | Thin Client        |
|-----------|-------|-------------|-----------|-----------|--------------------|
| DirRename | File  | Synchronous | Supported | Supported | Executed on Server |

### Syntax

`DirRename(strPath, strDirectoryFrom, strDirectoryTo)`

#### strPath

The path of the directory that will be renamed.

 **Tip:** This function supports wildcard ( \* and ? ).

#### strDirectoryFrom

The original name of the directory that will be renamed.

#### strDirectoryTo

The target name used to rename the original directory.

### Returned value

|    |                                                                         |
|----|-------------------------------------------------------------------------|
| -1 | Invalid parameters                                                      |
| 0  | Failed to rename the directory (e.g., strDirectoryFrom does not exist.) |
| 1  | Directory renamed successfully.                                         |

### Examples

| Tag Name | Expression                                                                                        |
|----------|---------------------------------------------------------------------------------------------------|
| Tag      | <code>DirRename("C:\Studio\","Temp", "New")</code> // C:\Studio\Temp is renamed to C:\Studio\New. |

### FileCopy

Copies the file(s) configured in the strSourceFile parameter to the path/file configured in the strTargetFile parameter.

| Function | Group | Execution   | Windows   | Embedded  | Thin Client |
|----------|-------|-------------|-----------|-----------|-------------|
| FileCopy | File  | Synchronous | Supported | Supported | Supported   |

## 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

|    |                              |
|----|------------------------------|
| -1 | Timeout time expired.        |
| 0  | Failed to copy file(s).      |
| 1  | File(s) copied successfully. |

## 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

| Tag Name | Expression                                                                                |
|----------|-------------------------------------------------------------------------------------------|
| Tag      | <code>FileCopy( "C:\Studio\Project\HST\*.hst", "C:\Temp\Hst\", 1000 )</code>              |
| Tag      | <code>FileCopy( "C:\Studio\Project\opert.txt", "C:\Temp\Tuesday_Report.txt", 500 )</code> |

## FileDelete

Deletes the specified file.

| Function   | Group | Execution   | Windows   | Embedded  | Thin Client |
|------------|-------|-------------|-----------|-----------|-------------|
| FileDelete | File  | Synchronous | Supported | Supported | Supported   |

## Syntax

`FileDelete( strFile )`

### strFile

The file path and name of the file to delete.

## Returned value

|      |                                      |
|------|--------------------------------------|
| 0    | Failed to delete file                |
| Real | Returns the size of the file deleted |

## Examples

| Tag Name | Expression                                        |
|----------|---------------------------------------------------|
| Tag      | <code>FileDelete( "C:\Studio\Readme.txt" )</code> |

## FileLength

Gets the size of a file.

| Function   | Group | Execution   | Windows   | Embedded  | Thin Client |
|------------|-------|-------------|-----------|-----------|-------------|
| FileLength | File  | Synchronous | Supported | Supported | Supported   |

## Syntax

`FileLength( strFile )`

### strFile

The file path and name of the file.

## Returned value

Returns the size of the specified file in bytes.

## Examples

| Tag Name | Expression                                 |
|----------|--------------------------------------------|
| Tag      | <code>FileLength( "C:\Readme.txt" )</code> |

## FileRename

FileRename is a built-in scripting function that renames a specified file.

| Function   | Group | Execution   | Windows   | Embedded  | Thin Client |
|------------|-------|-------------|-----------|-----------|-------------|
| FileRename | File  | Synchronous | Supported | Supported | Supported   |

## 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

Writes a string to a specified ASCII or Unicode file. If the file doesn't exist, then the function will create the file.

| Function  | Group | Execution   | Windows   | Embedded  | Thin Client |
|-----------|-------|-------------|-----------|-----------|-------------|
| FileWrite | File  | Synchronous | Supported | Supported | Supported   |

## Description

Writes a string to a specified ASCII or Unicode file. If the file doesn't exist, then the function will create the file.

## Syntax

`FileWrite( strFileName, strWriteText{ | , optNumAppend }`

### strFileName

A string value enclosed in quotes, or the name of a String tag that contains the value, specifying the file name.

By default, the file will be written in your project folder (i.e., the folder that contains the file `project_name.APP`). To write in another folder, specify the complete file path.

### strWriteText

A string value enclosed in quotes, or the name of a String tag that contains the value, specifying the text to be written to the file.

### optNumAppend

A numerical value, or the name of an Integer tag that contains the value, specifying the text encoding of the file:

| Value | Description                                                                                                          |
|-------|----------------------------------------------------------------------------------------------------------------------|
| 0     | Creates a new ASCII file with the given file name. If the file already exists, then it is overwritten.               |
| 1     | Appends to an existing ASCII file with the given file name. If the file doesn't exist, then a new file is created.   |
| 2     | Creates a new Unicode (UTF-16LE) file with the given file name. If the file already exists, then it is overwritten.  |
| 3     | Appends to an existing Unicode file with the given file name. If the file doesn't exist, then a new file is created. |

*This is an optional parameter.* If any value other than 0 through 3 is given, or if the parameter is not used, then 0 is the default.

## Returned value

|    |                                                                       |
|----|-----------------------------------------------------------------------|
| 0  | Successful execution.                                                 |
| -1 | Invalid parameter(s).                                                 |
| -2 | Failed to open file. Disk is write protected or file name is invalid. |

## Examples

| Tag Name | Expression                                               |
|----------|----------------------------------------------------------|
| Tag      | <code>FileWrite( "c: est.txt", "This is a test" )</code> |
| Tag      | <code>FileWrite( strFileName, strWriteText )</code>      |
| Tag      | <code>FileWrite( strFileName, strWriteText, 1 )</code>   |

## FindFile

FindFile is a built-in scripting function that searches for all files that match a given search string.

| Function | Group | Execution   | Windows   | Embedded  | Thin Client |
|----------|-------|-------------|-----------|-----------|-------------|
| FindFile | File  | Synchronous | Supported | Supported | Supported   |

## Syntax

`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.

 **Note:** The array will receive only the file names and *not* their paths, even if you define strFile to search outside the default directory.

### 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.

### Returned value

This function will return one of the following values:

| Value    | Description                     |
|----------|---------------------------------|
| -1       | Function timed out.             |
| 0        | No matching files found.        |
| <i>n</i> | Number of matching files found. |

### Notes

This function may be called by any project client, but it is always executed on the project server. By default, it only searches the server's project folder, and if strFile includes a relative file path, then it must be relative to the that same directory.

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 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.

| Function | Group | Execution   | Windows   | Embedded  | Thin Client |
|----------|-------|-------------|-----------|-----------|-------------|
| FindPath | File  | Synchronous | Supported | Supported | Supported   |

### Syntax

```
FindPath(strPathName)
```

#### strPathName

The file path for which to search.

## Returned value

|   |                |
|---|----------------|
| 0 | Path not found |
| 1 | Path found     |

## Examples

| Tag Name | Expression                            |
|----------|---------------------------------------|
| Tag      | <code>FindPath( "C:\Windows" )</code> |

## GetFileAttributes

Reads the attributes of a specified file.

| Function          | Group | Execution   | Windows   | Embedded  | Thin Client |
|-------------------|-------|-------------|-----------|-----------|-------------|
| GetFileAttributes | File  | Synchronous | Supported | Supported | Supported   |

## Syntax

`GetFileAttributes( strFile )`

### strFile

The file path and name of the file from which to read the attributes.

## Returned value

|     |           |
|-----|-----------|
| -1  | Error     |
| 1   | Read only |
| 2   | Hidden    |
| 4   | System    |
| 16  | Directory |
| 32  | Archive   |
| 128 | Normal    |
| 256 | Temporary |

## Examples

| Tag Name | Expression                                        |
|----------|---------------------------------------------------|
| Tag      | <code>GetFileAttributes( "C:\Readme.txt" )</code> |

## GetFileTime

Reads the time and date the file was last modified.

| Function    | Group | Execution   | Windows   | Embedded  | Thin Client |
|-------------|-------|-------------|-----------|-----------|-------------|
| GetFileTime | File  | Synchronous | Supported | Supported | Supported   |

## 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

| Tag Name | Expression                                  |
|----------|---------------------------------------------|
| Tag      | <code>GetFileTime( "C:\Readme.txt" )</code> |

## GetHstInfo

Returns the Start Time, End Time, and Duration of the specified history (\*.HST) file.

| Function   | Group | Execution   | Windows   | Embedded  | Thin Client |
|------------|-------|-------------|-----------|-----------|-------------|
| GetHstInfo | File  | Synchronous | Supported | Supported | Supported   |

## 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.                                                                                                                                                          |
| -4 | 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 <a href="#">Verify Project</a> tool to update the project and try again. |

## Examples

| Tag Name | Expression                                       |
|----------|--------------------------------------------------|
| Tag      | <code>GetHstInfo( "batch", 0 )</code>            |
| Tag      | <code>GetHstInfo( "hst/02060801.hst", 1 )</code> |
| Tag      | <code>GetHstInfo( "C:\batch.bat", 2 )</code>     |

## GetLine

Gets a specified line or search string from a text file and then stores the line in a String tag.

| Function | Group | Execution   | Windows   | Embedded  | Thin Client |
|----------|-------|-------------|-----------|-----------|-------------|
| GetLine  | File  | Synchronous | Supported | Supported | Supported   |

## 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 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:

|    |                                                  |
|----|--------------------------------------------------|
| 0  | String was not found in the target file          |
| -1 | File not found                                   |
| -2 | Invalid <code>strFileName</code> parameter       |
| -3 | Invalid <code>strSeqChar</code> parameter        |
| -4 | Invalid <code>strStoreTag</code> parameter       |
| -5 | Invalid <code>optNumCase</code> parameter        |
| -6 | Invalid <code>optNumOverflowTag</code> parameter |

|  |                              |
|--|------------------------------|
|  | Invalid number of parameters |
|  | Invalid line number          |

**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:

```
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

| Example Name | Expression                                                                                                                             |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------|
|              | <pre>GetLine( "C:\TechRef v61.doc", "Studio version 6.1", "ReturnedLine" ) // Gets the first occurrence of "Studio version 6.1".</pre> |
|              | <pre>GetLine( "C:\Readme.txt", 1, "ReturnedLine", 0, "Overflow" ) // Gets the second line of the file.</pre>                           |

## TXT

Converts information from the Trend history file(s) in proprietary binary format ( \*.hst ) to a plain text ( \*.txt ) or comma-delimited ( \*.csv ) file.

| Option | Group | Execution    | Windows   | Embedded  | Thin Client |
|--------|-------|--------------|-----------|-----------|-------------|
| TXT    | File  | Asynchronous | Supported | Supported | Supported   |

## Syntax

`GetTrendHistory( strStartDate, strStartTime, numDuration, numGroup, strTargetFile, optStrSeparator, optNumMilliseconds, optStrFormat, optThinClient )`

### strStartDate

The start date of the data.

### strStartTime

The start time of the data.

### numDuration

Numerical tag containing duration of the data in hours.

### numGroup

Numerical tag containing trend group number.

### strTargetFile

String tag containing path and name of the file to be written.

### optStrSeparator

*Optional* The data separator character for file. If omitted, the function uses the TAB character (\t) to separate the values in the text file.

### optNumMilliseconds

*Optional* numeric tag. If this parameter is false (0), the text file created will not show milisecond-precision on the timestamp of each history sample.

### optStrFormat

String tag, which specifies the order of the Month (M), Day (D), and Year (Y) for the time-stamp format exported to the text file:

- "DMY": Day, Month, Year
- "MDY": Month, Day, Year
- "YMD": Year, Month, Day

If omitted, the function uses the format DMY for the timestamp in the text file.

### optNumInterval

*Optional* numeric tag specifying the sampling interval. Only line itmes at this interval are written as text to the target file; all other line items in the Trend history are discarded.

For example, if **optNumInterval** has a value of 10, then only every tenth line item is written out.

### Returned value

|    |                                                                            |
|----|----------------------------------------------------------------------------|
| -3 | Invalid number of parameters                                               |
| -2 | Dll functions not found                                                    |
| -1 | <b>InStudios.dll</b> not found                                             |
| 0  | Function was executed successfully                                         |
| 1  | Error. Previous execution of the <b>HST2TXT</b> has not yet been completed |

### Examples

| Tag Name | Expression                                                                                |
|----------|-------------------------------------------------------------------------------------------|
| Tag      | <b>HST2TXT</b> ( "04/14/2002", "06:30:00", 0.1, 3, "C:\Studio\data.txt", "\" " )          |
| Tag      | <b>HST2TXT</b> ( "04/14/2002", "06:30:00", 0.1, 3, "C:\Studio\data.csv", "\", "MDY" )     |
| Tag      | <b>HST2TXT</b> ( "04/14/2002", "06:30:00", 0.1, 3, "C:\Studio\data.csv", "\", "MDY", 10 ) |

 **Tip:** When using the comma character (,) as **optStrSeparator**, the function creates a file in the CSV format (Comma Separated Values). It is a useful tool for exporting the Trend history data from the proprietary binary format into a file that can be opened with Microsoft Excel.

### HST2TXTIsRunning

Returns the status of the HST2TXT function.

| Function         | Group | Execution   | Windows   | Embedded      | Thin Client        |
|------------------|-------|-------------|-----------|---------------|--------------------|
| HST2TXTIsRunning | File  | Synchronous | Supported | Not supported | Executed on Server |

### Syntax

HST2TXTIsRunning()

This function takes no parameters.

## Returned value

|     |                                                                                                     |
|-----|-----------------------------------------------------------------------------------------------------|
| 0   | <b>HST2TXT</b> is still running.                                                                    |
| -1  | Last conversion process was executed properly.                                                      |
| -2  | Reserved.                                                                                           |
| -3  | File not found. There are no history files in the configured time interval for the group specified. |
| -4  | Cannot open <b>HST</b> file.                                                                        |
| -5  | Cannot create/open ASCII file.                                                                      |
| -6  | Cannot read file information from <b>HST</b> file.                                                  |
| -7  | Invalid file type.                                                                                  |
| -8  | Cannot read header information from <b>HST</b> file.                                                |
| -9  | Invalid number of tag in the header information ( $0 > nTags > 250$ ).                              |
| -10 | Cannot create Header file ( <b>.hdr</b> ).                                                          |
| -20 | InStudiot.dll was not found.                                                                        |
| -30 | Cannot access dll function.                                                                         |

## Examples

`HST2TXTIsRunning()`

## LookupContains

This function verifies that an external file contains the specified keyword in its key column.

| Function       | Group | Execution   | Windows   | Embedded  | Thin Client |
|----------------|-------|-------------|-----------|-----------|-------------|
| LookupContains | File  | Synchronous | Supported | Supported | Supported   |

## Syntax

`LookupContains(strKey)`

### **strKey**

The keyword to look for in the file's keywords column.

## Returned value

This function returns the following possible values:

| Value | Description                  |
|-------|------------------------------|
| 0     | Specified keyword not found. |
| 1     | Specified keyword found.     |

## Notes

The external file must already be loaded by calling the [LookupLoad](#) function.

## Examples

`LookupContains( "customer167" )`

## LookupGet

This function gets a value from an external file by cross-referencing from a specified keyword.

| Function  | Group | Execution   | Windows   | Embedded  | Thin Client |
|-----------|-------|-------------|-----------|-----------|-------------|
| LookupGet | File  | Synchronous | Supported | Supported | Supported   |

## 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 [LookupLoad](#) function.

## 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.

| Function   | Group | Execution   | Windows   | Embedded  | Thin Client |
|------------|-------|-------------|-----------|-----------|-------------|
| LookupLoad | File  | Synchronous | Supported | Supported | Supported   |

## 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

```
LookupLoad("C:\Temp\customerlist.csv", 1, 4, ",")
```

## PDFCreate

Creates a PDF file from the specified source file.

| Function  | Group | Execution   | Windows   | Embedded      | Thin Client |
|-----------|-------|-------------|-----------|---------------|-------------|
| PDFCreate | File  | Synchronous | Supported | Not supported | Supported   |

## Syntax

```
PDFCreate(strSourceFile{ | , optStrPdfFile }
```

## strSourceFile

String specifying the file path and name of the desired source file ( `*.doc`, `*.txt`, or `*.rtf` ). If a complete path is not specified, then the function will look for the source file in the project folder.

## optStrPdfFile

*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, `\path\MyDocument.rtf` becomes `\path\MyDocument.pdf`.

 **Note:** When entering the file name without a path, a leading backslash ("\") is optional.

## Returned value

| Value | Description                      |
|-------|----------------------------------|
| 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             |

 **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 101.

## Examples

| Tag Name | Expression                                                           |
|----------|----------------------------------------------------------------------|
|          | <code>PDFCreate( "C:\Report1.rtf" )</code>                           |
|          | <code>PDFCreate( "C:\Report2.doc", "C:\Converted1.pdf" )</code>      |
|          | <code>PDFCreate( "C:\Report3.txt", "C:\Data\Converted1.pdf" )</code> |

## Print

Prints a text file.

| Function | Group | Execution    | Windows   | Embedded  | Thin Client |
|----------|-------|--------------|-----------|-----------|-------------|
| Print    | File  | Asynchronous | Supported | Supported | Supported   |

## 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

| Tag Name | Expression                             |
|----------|----------------------------------------|
|          | <code>Print("C:\ReadMe.txt")</code>    |
|          | <code>Print("C:\ReadMe.txt", 1)</code> |
|          | <code>Print(TagFileName, 0)</code>     |

 **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.

| Function | Group | Execution   | Windows   | Embedded  | Thin Client |
|----------|-------|-------------|-----------|-----------|-------------|
| RDFileN  | File  | Synchronous | Supported | Supported | Supported   |

### Syntax

`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

|   |                                          |
|---|------------------------------------------|
| 0 | Success                                  |
| 1 | One of the parameters is not a string    |
| 2 | Parameter 1 contains an invalid tag name |
| 3 | The user canceled the operation          |

### Examples

| Tag Name | Expression                                                    |
|----------|---------------------------------------------------------------|
| Tag      | <code>RDFileN( "FileName", "C:\Studio\*", "*.doc", 1 )</code> |

## WebGetFile

Downloads a file from a specified address and then saves it locally.

| Function   | Group | Execution   | Windows   | Embedded  | Thin Client |
|------------|-------|-------------|-----------|-----------|-------------|
| WebGetFile | File  | Synchronous | Supported | Supported | Supported   |

### 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

|    |                         |
|----|-------------------------|
| -1 | Timeout                 |
| 0  | File not found          |
| 1  | File saved successfully |

### Examples

| Tag Name | Expression                                                                           |
|----------|--------------------------------------------------------------------------------------|
| Tag      | <code>WebGetFile( "http://www.the-internet.com/myfile.txt", "C:\myfile.txt" )</code> |
| Tag      | <code>WebGetFile( myURL, myFilePath )</code>                                         |

## 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.)

| Function   | Group   | Execution   | Windows   | Embedded  | Thin Client |
|------------|---------|-------------|-----------|-----------|-------------|
| AutoFormat | Graphic | Synchronous | Supported | Supported | Supported   |

### 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.

| Tag Name | Expression                                                       |
|----------|------------------------------------------------------------------|
| Tag      | <code>AutoFormat ( 1.543210 )</code> // Returned value = "1.543" |
| Tag      | <code>AutoFormat ( #3.123456 )</code> // Returned value = "-3.1" |

### **GetScrInfo**

Retrieves information from the project about an open screen.

| Function   | Group   | Execution   | Windows   | Embedded  | Thin Client |
|------------|---------|-------------|-----------|-----------|-------------|
| GetScrInfo | Graphic | Synchronous | Supported | Supported | Supported   |

### Syntax

`GetScrInfo( strScreenName, "tagResult", optNumResultType, optNumID )`

#### **strScreenName**

The name of the screen for which information is required.

#### **tagResult**

The name of the tag that will receive the information retrieved by the function. This name must be enclosed in quotes, or the project will try to get the contents of the tag.

#### **optNumResultType**

A numeric flag specifying the type of information to be retrieved by the function:

| Value | Description                                                                                                                                                                               |
|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0     | Default value. Writes the TOP, LEFT, BOTTOM and RIGHT screen coordinates to each consecutive position of the <a href="#">array tag</a> specified by the <code>tagResult</code> parameter. |
| 1     | Writes the TOP screen coordinate to the tag specified by the <code>tagResult</code> parameter.                                                                                            |
| 2     | Writes the LEFT screen coordinate to the tag specified by the <code>tagResult</code> parameter.                                                                                           |

| Value | Description                                                                          |
|-------|--------------------------------------------------------------------------------------|
| 3     | Writes the BOTTOM screen coordinate to the tag specified by the tagResult parameter. |
| 4     | Writes the RIGHT screen coordinate to the tag specified by the tagResult parameter.  |

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 [Open](#) function.) This is an optional parameter; the default ID is 0.

### Returned value

|    |                                                                        |
|----|------------------------------------------------------------------------|
| 0  | Success                                                                |
| -1 | The first and/or second parameters are not strings.                    |
| -2 | Memory allocation error.                                               |
| -3 | optNumResultType is zero, but tagResult does not specify an array tag. |
| -4 | Invalid tag by the tagResult parameter.                                |

### Examples

| Tag Name     | Expression                                                                                                                                                                                             |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TagErrorCode | <code>GetScrInfo( "main" , "TagXY[ 0]" )</code> // Retrieves the TOP, LEFT, BOTTOM and RIGHT coordinates of the "main" screen and then writes them to the first four positions of the array tag TagXY. |
| TagErrorCode | <code>GetScrInfo( "main", "TagXY", 3 )</code> // Retrieves the BOTTOM coordinate of the "main" screen and then writes it to TagXY.                                                                     |
| TagErrorCode | <code>GetScrInfo( "main" , "TagXY", 2, 10 )</code> // Retrieves the LEFT coordinate of the "main" screen with ID 10 and then writes it to TagXY.                                                       |

### PrintSetup

Opens the standard setup dialog from the operating system, from where the printer can be selected and configured.

| Function   | Group   | Execution    | Windows   | Embedded  | Thin Client |
|------------|---------|--------------|-----------|-----------|-------------|
| PrintSetup | Graphic | Asynchronous | Supported | Supported | Supported   |

 **Note:** This function cannot be used with [Tasks](#) or in the [Global Procedures](#) script.

### Syntax

`PrintSetup()`

This function takes no parameters.

### Returned value

No returned values.

### Examples

`PrintSetup()`

### 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.

| Function    | Group   | Execution    | Windows   | Embedded  | Thin Client |
|-------------|---------|--------------|-----------|-----------|-------------|
| PrintWindow | Graphic | Asynchronous | Supported | Supported | Supported   |

 **Note:** This function cannot be used with [Tasks](#) or in the [Global Procedures](#) script.

## Syntax

`PrintWindow( strScreenName , optNumOrientation , 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 paper orientation:

| Value | Description |
|-------|-------------|
| 0     | Portrait    |
| 1     | Landscape   |

Default value is 0.

 **Note:** The `optNumOrientation` parameter is not supported when running the project on a Windows Embedded target system.

### **optNumID**

The specific instance number of the screen. (The ID is assigned when the screen is opened with the [Open](#) function.) This is an optional parameter; 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 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:** The `optStrMnemonicList` parameter does not work for a screen that is already open; if the screen has been opened, then the custom properties have already received their default values.

## Returned value

This function does not return any value.

## Examples

| Tag Name | Expression                                                                                                                                                                                                                           |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|          | <code>PrintWindow( )</code> // Prints the currently active screen in portrait orientation.                                                                                                                                           |
|          | <code>PrintWindow( "Main", 1 )</code> // Prints the "Main" screen in landscape orientation.                                                                                                                                          |
|          | <code>PrintWindow( TagScreenName )</code> // Prints the screen specified by TagScreenName.                                                                                                                                           |
|          | <code>PrintWindow( "Main", 1, 10 )</code> // Prints the "Main" screen with ID 10.                                                                                                                                                    |
|          | <code>PrintWindow ( "Main", 1, 0, "#Mne1:Tag1 #Mne2:Tag2" )</code> // Prints the "Main" screen, replacing the custom properties <code>Mne1</code> and <code>Mne2</code> with <code>Tag1</code> and <code>Tag2</code> , respectively. |

 **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.

| Function                | Group   | Execution   | Windows   | Embedded  | Thin Client |
|-------------------------|---------|-------------|-----------|-----------|-------------|
| ResetDecimalPointsTable | Graphic | Synchronous | Supported | Supported | Supported   |

**ResetDecimalPointsTable()**

### Syntax

ResetDecimalPointsTable()

This function takes no parameters.

### Returned value

This function does not return any value.

### Examples

| Tag Name | Expression                                                                |
|----------|---------------------------------------------------------------------------|
|          | <b>ResetDecimalPointsTable()</b> // Resets the virtual table of settings. |

## RGBColor

Returns the number of the color defined by the RGB (Red, Green, Blue) codes.

| Function | Group   | Execution   | Windows   | Embedded  | Thin Client |
|----------|---------|-------------|-----------|-----------|-------------|
| RGBColor | Graphic | Synchronous | Supported | Supported | Supported   |

### 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

| Tag Name | Expression                                                                                                                                                     |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TagColor | <b>RGBColor(51,153,102)</b> // This function returns the value 13434828, which is the color code for Sea Green.                                                |
| TagColor | <b>RGBColor(TagRed,TagGreen,TagBlue)</b> // This function returns the color code of the RGB values set in the tags TagRed, TagGreen and TagBlue, respectively. |

 **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.

| Function     | Group   | Execution   | Windows   | Embedded  | Thin Client |
|--------------|---------|-------------|-----------|-----------|-------------|
| RGBComponent | Graphic | Synchronous | Supported | Supported | Supported   |

## 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

This function takes a screen shot of the project screen and saves it to an image file.

| Function       | Group   | Execution   | Windows   | Embedded                          | Thin Client        |
|----------------|---------|-------------|-----------|-----------------------------------|--------------------|
| SaveScreenShot | Graphic | Synchronous | Supported | Supported (only for open screens) | Executed on Server |

## Syntax

`SaveScreenShot( { | optStrScreenName{ | , optStrOutputFile{ | , { optNumFormat | 0 | 1 | 2 | 3 | 4 | 5 } } } )`

`SaveScreenShot( )`

`SaveScreenShot( optStrScreenName )`

`SaveScreenShot( optStrScreenName, optStrOutputFile )`

`SaveScreenShot( optStrScreenName, optStrOutputFile, optNumFormat )`

### optStrScreenName

The name of a project screen file (\* .scr). If no file path is specified, then the file must be located in the Screen folder of the project (e.g., `\projectname\Screen\screenname.scr`).

For projects running on Windows, the screen may be either open or closed. For projects running on Windows Embedded, the screen must be open.

This is an optional parameter; if no value (or " ") is specified, then the currently open and active screen is used.

### optStrOutputFile

The name of the output file. If no file path is specified, then the file is saved in the Web folder of the project (e.g., `\projectname\Web\screenname.jpg`)

This is an optional parameter; if no value is specified, then either the value of `optStrScreenName` or simply `ScreenShot.jpg` is used.

### optNumFormat

The format of the image file:

| Value | Description |
|-------|-------------|
| 0     | BMP         |

| Value | Description |
|-------|-------------|
| 1     | JPG         |
| 2     | PNG         |
| 3     | GIF         |
| 4     | TIFF        |
| 5     | Auto        |

This is an optional parameter; if no value is specified, then the default is 1 (JPG).

### Returned value

This function returns the following possible values:

| Value  | Description                                                                                                      |
|--------|------------------------------------------------------------------------------------------------------------------|
| 0      | Success.                                                                                                         |
| -1     | Wrong number of parameters.                                                                                      |
| -2     | Wrong parameter types.                                                                                           |
| -3     | Invalid directory.                                                                                               |
| -4     | Second parameter cannot be empty.                                                                                |
| -5     | Wrong format / invalid option for third parameter.                                                               |
| -6     | Failed to save file.                                                                                             |
| -7     | Failed to create compatible bitmap.                                                                              |
| -10000 | Project is not running. (This typically happens when you try to use the function in the Database Spy.) See note. |

### Notes

For this function to be properly executed, the project must be running and the Viewer task must be started. The function can be called by the Background Tasks task — that is, by a Script or Math worksheet — but it will fail if the Viewer task is not also started. This is because the Viewer task is needed to actually render the screen.

### Examples

Save the currently active screen to `\projectname\Web\ScreenShot.JPG`:

```
saveScreenShot ()
```

Save the screen file `main.scr` to `\projectname\Web\main.JPG`:

```
saveScreenShot("main.scr")
```

Save the currently active screen as a BMP with the name of the currently logged user:

```
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.

| Function         | Group   | Execution   | Windows   | Embedded  | Thin Client |
|------------------|---------|-------------|-----------|-----------|-------------|
| SetDecimalPoints | Graphic | Synchronous | Supported | Supported | Supported   |

### Syntax

```
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

|   |         |
|---|---------|
| 0 | Error   |
| 1 | Success |

### 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

| Tag Name | Expression                                                                                                              |
|----------|-------------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>SetDecimalPoints( 1.5, 3 )</code> // Displays 3 decimal places for all real numbers greater than or equal to 1.5. |
| Tag      | <code>SetDecimalPoints( #3, 1 )</code> // Displays 1 decimal place for all real numbers less than or equal to -3.       |

### 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.

| Function       | Group   | Execution   | Windows   | Embedded  | Thin Client |
|----------------|---------|-------------|-----------|-----------|-------------|
| SetDisplayUnit | Graphic | Synchronous | Supported | Supported | Supported   |

### 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

|    |                                           |
|----|-------------------------------------------|
| 0  | Success.                                  |
| -1 | Wrong number of parameters.               |
| -2 | strUnitOrigin parameter is empty.         |
| -3 | numDiv parameter is invalid (equal to 0). |

## 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

| Tag Name | Expression                                                                                                                                                                                                                                                         |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>SetDisplayUnit( "C", "F", 0.555556, 32 )</code> // For all tags and object values with a <b>Unit</b> of "C", the <b>DisplayUnit</b> tag field is set to "F", the <b>UnitDiv</b> tag field is set to 0.555556, and the <b>UnitAdd</b> tag field is set to 32. |

## SetTagDisplayUnit

Sets the **DisplayUnit**, **UnitDiv**, and **UnitAdd** properties on a specific tag.

| Function          | Group   | Execution   | Windows   | Embedded  | Thin Client |
|-------------------|---------|-------------|-----------|-----------|-------------|
| SetTagDisplayUnit | Graphic | Synchronous | Supported | Supported | Supported   |

## 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

|    |                                           |
|----|-------------------------------------------|
| 0  | Success.                                  |
| -1 | Wrong number of parameters.               |
| -2 | Specified tag doesn't exist.              |
| -3 | numDiv parameter is invalid (equal to 0). |

## Examples

| Tag Name | Expression                                                                                                                                                                                                                               |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>SetTagDisplayUnit( "TagTemp", "F", 0.555556, 32 )</code> // For the tag "TagTemp", the <b>DisplayUnit</b> tag field is set to "F", the <b>UnitDiv</b> tag field is set to 0.555556, and the <b>UnitAdd</b> tag field is set to 32. |

## Translation functions

These functions are used to access the translation tool during runtime.

### Ext

Translates specified text using the active translation file.

| Function | Group       | Execution   | Windows   | Embedded  | Thin Client |
|----------|-------------|-------------|-----------|-----------|-------------|
| Ext      | Translation | Synchronous | Supported | Supported | Supported   |

### Syntax

Ext ( *strText* )

#### **strText**

The text to be translated.

### Returned value

Returns the text translation using the active translation file.

### Examples

| Tag Name | Expression                                                                        |
|----------|-----------------------------------------------------------------------------------|
| Tag      | <b>Ext</b> ( "Start" ) // Returned value if translating to Portuguese = "Iniciar" |
| Tag      | <b>Ext</b> ( "Stop" ) // Returned value if translating to German = "Anschlag"     |

### SetLanguage

This function sets the language of the project to one of the languages configured in the Translation Table.

| Function    | Group       | Execution   | Windows   | Embedded  | Thin Client |
|-------------|-------------|-------------|-----------|-----------|-------------|
| SetLanguage | Translation | Synchronous | Supported | Supported | Supported   |

### Syntax

SetLanguage ( *numLanguageCode* )

#### **numLanguageCode**

The code 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 the following possible values:

| Value | Description |
|-------|-------------|
| 0     | Error       |
| 1     | Success     |

### Notes

Language codes are defined by [ISO 639-1](#).

### Examples

Set the language to "English – United States":

```
SetLanguage(1033)
```

Set the language to "French – France":

```
SetLanguage(1036)
```

## SetTranslationFile

Sets the active translation file and translates all enabled text within the project.

| Function           | Group       | Execution   | Windows   | Embedded  | Thin Client |
|--------------------|-------------|-------------|-----------|-----------|-------------|
| SetTranslationFile | Translation | Synchronous | Supported | Supported | Supported   |

### Syntax

```
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

|    |                                                |
|----|------------------------------------------------|
| 0  | Success.                                       |
| -1 | Invalid number of parameters.                  |
| -2 | Wrong parameter type.                          |
| -3 | Translation file could not be found or opened. |

### Examples

| Tag Name | Expression                                                            |
|----------|-----------------------------------------------------------------------|
| Tag      | <code>SetTranslationFile( "Portuguese.tra" )</code>                   |
| Tag      | <code>SetTranslationFile( "German.tra" )</code>                       |
| Tag      | <code>SetTranslationFile( "Mytranslation.csv" , "Portuguese" )</code> |
| Tag      | <code>SetTranslationFile( "Mytranslation.csv" , "German" )</code>     |



**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](#).

## Multimedia functions

These functions are used to play external audio and video files.

### Play

Plays a specified WAV audio file.

| Function | Group      | Execution    | Windows   | Embedded  | Thin Client |
|----------|------------|--------------|-----------|-----------|-------------|
| Play     | Multimedia | Asynchronous | Supported | Supported | Supported   |

 **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

```
Play(strFileName{ | , optNumSynchronous }
```

#### **strFileName**

The file path and name of the WAV file to play.

#### **optNumSynchronous**

A numeric flag specifying whether the function executes synchronously or asynchronously:

| Value | Description                                                                           |
|-------|---------------------------------------------------------------------------------------|
| 0     | Asynchronous (i.e., the project continues without waiting for the function to return) |
| 1     | Synchronous (i.e., the project pauses while it waits for the function to return)      |

This is an optional paramter; if no value is specified, then the default is 0.

### Returned value

This function does not return any value.

### Examples

| Tag Name | Expression                        |
|----------|-----------------------------------|
|          | Play( "C:\Sounds\Wav\alarm.wav" ) |

## System Info functions

These functions are used 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.

| Function  | Group       | Execution   | Windows   | Embedded  | Thin Client |
|-----------|-------------|-------------|-----------|-----------|-------------|
| DbVersion | System Info | Synchronous | Supported | Supported | Supported   |

### Syntax

`DbVersion()`

This function takes no paramters.

### 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.

| Function                   | Group       | Execution   | Windows   | Embedded      | Thin Client        |
|----------------------------|-------------|-------------|-----------|---------------|--------------------|
| GetAppHorizontalResolution | System Info | Synchronous | Supported | Not supported | Executed on Server |

### 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

| Tag Name | Expression                                                        |
|----------|-------------------------------------------------------------------|
| Tag      | <code>GetAppHorizontalResolution()</code> // Returned value = 640 |
| Tag      | <code>GetAppHorizontalResolution()</code> // Returned value = 800 |

### ***GetAppPath***

Returns the file path of the project folder.

| Function   | Group       | Execution   | Windows   | Embedded  | Thin Client        |
|------------|-------------|-------------|-----------|-----------|--------------------|
| GetAppPath | System Info | Synchronous | Supported | Supported | Executed on Server |

### Syntax

`GetAppPath()`

This function takes no paramters.

## Returned value

Returns the file path as a string.

## Examples

| Tag Name | Expression                                                                        |
|----------|-----------------------------------------------------------------------------------|
| Tag      | <code>GetAppPath ( )</code> // Returned value = "C:\DemoProject"                  |
| Tag      | <code>GetAppPath ( )</code> // Returned value = "C:\Studio\Projects\project_name" |

 **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       | Execution   | Windows   | Embedded      | Thin Client        |
|--------------------------|-------------|-------------|-----------|---------------|--------------------|
| GetAppVerticalResolution | System Info | Synchronous | Supported | Not supported | Executed on Server |

## 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

| Tag Name | Expression                                                        |
|----------|-------------------------------------------------------------------|
| Tag      | <code>GetAppVerticalResolution ( )</code> // Returned value = 480 |
| Tag      | <code>GetAppVerticalResolution ( )</code> // Returned value = 600 |

## GetComputerIP

Returns the first IP Address of the local computer.

| Function      | Group       | Execution   | Windows   | Embedded  | Thin Client |
|---------------|-------------|-------------|-----------|-----------|-------------|
| GetComputerIP | System Info | Synchronous | Supported | Supported | Supported   |

## Syntax

`GetComputerIP ( )`

This function takes no parameters.

## Returned value

Returns the first IP Address of the local station as a string.

## Examples

| Tag Name | Expression                                                       |
|----------|------------------------------------------------------------------|
| Tag      | <code>GetComputerIP ( )</code> // Returned value = "192.168.0.1" |
| Tag      | <code>GetComputerIP ( )</code> // Returned value = "248.12.2.78" |

## GetComputerName

Returns the local computer name.

| Function        | Group       | Execution   | Windows   | Embedded      | Thin Client |
|-----------------|-------------|-------------|-----------|---------------|-------------|
| GetComputerName | System Info | Synchronous | Supported | Not supported | Supported   |

## Syntax

GetComputerName ( )

This function takes no parameters.

## Returned value

Returns the local computer name as a string.

## Examples

| Tag Name | Expression                                                    |
|----------|---------------------------------------------------------------|
| Tag      | <b>GetComputerName ( )</b> // Returned value = "Terminal53"   |
| Tag      | <b>GetComputerName ( )</b> // Returned value = "BobsComputer" |

## GetCursorX

Gets the X-coordinate of the mouse cursor on the screen.

| Function   | Group       | Execution   | Windows   | Embedded  | Thin Client |
|------------|-------------|-------------|-----------|-----------|-------------|
| GetCursorX | System Info | Synchronous | Supported | Supported | Supported   |

## 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

| Tag Name | Expression                                     |
|----------|------------------------------------------------|
|          | <b>GetCursorX ( )</b> // Returned value = 1024 |

## GetCursorY

Gets the Y-coordinate of the mouse cursor on the screen.

| Function   | Group       | Execution   | Windows   | Embedded  | Thin Client |
|------------|-------------|-------------|-----------|-----------|-------------|
| GetCursorY | System Info | Synchronous | Supported | Supported | Supported   |

## 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

| Tag Name | Expression                                    |
|----------|-----------------------------------------------|
|          | <b>GetCursorY ( )</b> // Returned value = 768 |

## GetDisplayHorizontalResolution

Gets the horizontal resolution (in pixels) of the display connected to the local station.

| Function                       | Group       | Execution   | Windows   | Embedded  | Thin Client |
|--------------------------------|-------------|-------------|-----------|-----------|-------------|
| GetDisplayHorizontalResolution | System Info | Synchronous | Supported | Supported | Supported   |

## Syntax

GetDisplayHorizontalResolution()

This function takes no parameters.

## Returned value

This function returns the horizontal resolution of the display as an integer.

## Examples

| Tag Name | Expression                                                             |
|----------|------------------------------------------------------------------------|
|          | <code>GetDisplayHorizontalResolution()</code> // Returned value = 1024 |

## GetDisplayVerticalResolution

Gets the vertical resolution (in pixels) of the display connected to the local station.

| Function                     | Group       | Execution   | Windows   | Embedded  | Thin Client |
|------------------------------|-------------|-------------|-----------|-----------|-------------|
| GetDisplayVerticalResolution | System Info | Synchronous | Supported | Supported | Supported   |

## Syntax

GetDisplayVerticalResolution()

This function takes no parameters.

## Returned value

This function returns the vertical resolution of the display as an integer.

## Examples

| Tag Name | Expression                                                          |
|----------|---------------------------------------------------------------------|
|          | <code>GetDisplayVerticalResolution()</code> // Returned value = 768 |

## GetFreeMemoryCE

Returns the free memory available in a Windows Embedded device.

| Function        | Group       | Execution   | Windows       | Embedded  | Thin Client |
|-----------------|-------------|-------------|---------------|-----------|-------------|
| GetFreeMemoryCE | System Info | Synchronous | Not supported | Supported | Supported   |

## Syntax

GetFreeMemoryCE({ | *optNumType* })

### optNumType

A numeric flag that specifies which type of free memory the project should retrieve from the device:

| Value | Description               |
|-------|---------------------------|
| 0     | Total free program memory |

This is an optional parameter; if no value is specified, then the default is 0.

## Returned value

|    |                                 |
|----|---------------------------------|
| >0 | Size of free memory in bytes.   |
| -1 | Core.dll .dll file not found.   |
| -2 | GetMemoryCE function not found. |
| -3 | Invalid optional parameter.     |
| -4 | Type of memory unavailable.     |

## Examples

| Tag Name | Expression                       |
|----------|----------------------------------|
| Tag      | <code>GetFreeMemoryCE ( )</code> |

### ***GetHardKeyModel***

Returns the model name of your Hardkey.

| Function        | Group       | Execution   | Windows   | Embedded      | Thin Client        |
|-----------------|-------------|-------------|-----------|---------------|--------------------|
| GetHardKeyModel | System Info | Synchronous | Supported | Not supported | Executed on Server |

### 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

| Tag Name | Expression                                                             |
|----------|------------------------------------------------------------------------|
| Tag      | <code>GetHardKeyModel ( )</code> // Returned value = "Local Interface" |
| Tag      | <code>GetHardKeyModel ( )</code> // Returned value = "Advanced Server" |

### ***GetHardKeySN***

Returns the serial number of the Hardkey.

| Function     | Group       | Execution   | Windows   | Embedded      | Thin Client        |
|--------------|-------------|-------------|-----------|---------------|--------------------|
| GetHardKeySN | System Info | Synchronous | Supported | Not supported | Executed on Server |

### 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

| Tag Name | Expression                                                |
|----------|-----------------------------------------------------------|
| Tag      | <code>GetHardkeySN ( )</code> // Returned value = 120.745 |
| Tag      | <code>GetHardkeySN ( )</code> // Returned value = 224.941 |

## GetIPAll

Returns the number of IP Addresses assigned to the local station and stores the IP Addresses in a string array tag.

| Function | Group       | Execution   | Windows   | Embedded  | Thin Client |
|----------|-------------|-------------|-----------|-----------|-------------|
| GetIPAll | System Info | Synchronous | Supported | Supported | Supported   |

### 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

|          |                              |
|----------|------------------------------|
| <i>n</i> | Number of IP addresses found |
| -1       | Invalid number of parameters |
| -2       | Invalid parameter type       |

### Examples

| Tag Name | Expression                                                          |
|----------|---------------------------------------------------------------------|
| Tag      | <code>GetIPAll( "TagArrayIP" ) // Returned value = 1</code>         |
| Tag      | <code>GetIPAll( "TagArrayIP", Second ) // Returned value = 2</code> |

## GetMemoryCE

Returns the total memory available in a Windows Embedded device.

| Function    | Group       | Execution   | Windows       | Embedded  | Thin Client |
|-------------|-------------|-------------|---------------|-----------|-------------|
| GetMemoryCE | System Info | Synchronous | Not supported | Supported | Supported   |

### Syntax

`GetMemoryCE( { | optNumType } )`

#### optNumType

A numeric flag that specifies which type of memory the project should retrieve from the device:

| Value | Description          |
|-------|----------------------|
| 0     | Total program memory |
| 1     | Total storage memory |
| 2     | Total memory         |

This is an optional parameter; if no value is specified, then the default is 0.

### Returned value

|    |                                 |
|----|---------------------------------|
| >0 | Size of memory in bytes.        |
| -1 | Core.dll .dll file not found.   |
| -2 | GetMemoryCE function not found. |

|    |                             |
|----|-----------------------------|
| -3 | Invalid optional parameter. |
|----|-----------------------------|

### Examples

| Tag Name | Expression                    |
|----------|-------------------------------|
| Tag      | <code>GetMemoryCE( 1 )</code> |

### GetNetMACID

Gets the MAC ID unique code from the currently installed network adapter(s).

| Function    | Group       | Execution   | Windows   | Embedded  | Thin Client |
|-------------|-------------|-------------|-----------|-----------|-------------|
| GetNetMACID | System Info | Synchronous | Supported | Supported | Supported   |

### Syntax

`GetNetMACID( "optTagMACID", "optTagAdapterName" )`

#### 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.

#### 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.

### Returned value

| Value | Description                                         |
|-------|-----------------------------------------------------|
| >0    | Number of network adapters found.                   |
| 0     | No network adapters found.                          |
| -1    | Invalid number of parameters.                       |
| -2    | One of the parameters is not string type.           |
| -3    | Tag configured in optTagMACID does not exist.       |
| -4    | Tag configured in optTagAdapterName does not exist. |

### Examples

| Tag Name | Expression                                                  |
|----------|-------------------------------------------------------------|
| NumNIC   | <code>GetNetMACID( "MACIDTag" )</code>                      |
| NumNIC   | <code>GetNetMACID( "MACIDTag", "AdapterName" )</code>       |
| NumNIC   | <code>GetNetMACID( "MACIDTag[1]", "AdapterName[1]" )</code> |

### GetOS

Reports the current operating system.

| Function | Group       | Execution   | Windows   | Embedded  | Thin Client |
|----------|-------------|-------------|-----------|-----------|-------------|
| GetOS    | System Info | Synchronous | Supported | Supported | Supported   |

### Syntax

`GetOS()`

This function takes no parameters.

## Returned value

|   |                         |
|---|-------------------------|
| 0 | Windows 3.11            |
| 1 | Windows 95/98/ME        |
| 2 | Windows 2000/XP/Vista/7 |
| 3 | Windows CE/Mobile       |

## Examples

| Tag Name | Expression                                |
|----------|-------------------------------------------|
| Tag      | <code>GetOS()</code> //Returned value = 2 |

## GetPrivateProfileString

Reads a specified parameter from an .ini file using the standard .ini format.

| Function                | Group       | Execution   | Windows   | Embedded  | Thin Client |
|-------------------------|-------------|-------------|-----------|-----------|-------------|
| GetPrivateProfileString | System Info | Synchronous | Supported | Supported | Supported   |

## 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

| Tag Name | Expression                                                                                                   |
|----------|--------------------------------------------------------------------------------------------------------------|
| Tag      | <code>GetPrivateProfileString( "boot loader", "timeout", "50", "C:\boot.ini" )</code> // Returned value = 30 |

## GetProductPath

Gets the path to the program directory.

| Function       | Group       | Execution   | Windows   | Embedded  | Thin Client |
|----------------|-------------|-------------|-----------|-----------|-------------|
| GetProductPath | System Info | Synchronous | Supported | Supported | Supported   |

## Syntax

`GetProductPath()`

This function takes no parameters.

## Returned value

Returns the path to the program directory as a string.

## Examples

| Tag Name | Expression                                            |
|----------|-------------------------------------------------------|
| Tag      | <code>GetProductPath( )</code> // Returned value = "" |

## GetRegValue

Gets a the value of a variable in the Windows registry.

| Function    | Group       | Execution   | Windows   | Embedded  | Thin Client   |
|-------------|-------------|-------------|-----------|-----------|---------------|
| GetRegValue | System Info | Synchronous | Supported | Supported | Not supported |

## Syntax

`GetRegValue( numMainKey, strKey, strValueName )`

### numMainKey

Numeric tag with the following possible values:

|   |                       |
|---|-----------------------|
| 0 | HKEY_LOCAL_MACHINE    |
| 1 | HKEY_CLASSES_ROOT     |
| 2 | HKEY_CURRENT_USER     |
| 3 | HKEY_USERS            |
| 4 | HKEY_CURRENT_CONFIG   |
| 5 | HKEY_PERFORMANCE_DATA |

### 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:

|    |                                                                                             |
|----|---------------------------------------------------------------------------------------------|
| -1 | Invalid number of parameters or invalid Main Key.                                           |
| -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

| Tag Name | Expression                                                                                                     |
|----------|----------------------------------------------------------------------------------------------------------------|
| Tag      | <code>GetRegValue( 0, "HARDWARE\DESCRIPTION\System", "SystemBiosDate" )</code> // Returned value = "08/14/03"  |
| Tag      | <code>GetRegValue( 2, "Control Panel\Current", "Color Schemes" )</code> // Returned value = "Windows Standard" |

## GetRegValueType

Gets the data type of the value of a variable in the Windows registry.

| Function        | Group       | Execution   | Windows   | Embedded  | Thin Client   |
|-----------------|-------------|-------------|-----------|-----------|---------------|
| GetRegValueType | System Info | Synchronous | Supported | Supported | Not supported |

## Syntax

`GetRegValueType( numMainKey, strKey, strValueName )`

## numMainKey

Numeric tag with the following possible values:

|   |                       |
|---|-----------------------|
| 0 | HKEY_LOCAL_MACHINE    |
| 1 | HKEY_CLASSES_ROOT     |
| 2 | HKEY_CURRENT_USER     |
| 3 | HKEY_USERS            |
| 4 | HKEY_CURRENT_CONFIG   |
| 5 | HKEY_PERFORMANCE_DATA |

## 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

|    |                                                                                             |
|----|---------------------------------------------------------------------------------------------|
| 1  | Variable type is String.                                                                    |
| 0  | Variable type is DWord.                                                                     |
| -1 | Invalid number of parameters or invalid Main Key.                                           |
| -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

| Tag Name | Expression                                                                                               |
|----------|----------------------------------------------------------------------------------------------------------|
| Tag      | <code>GetRegValueType( 0, "HARDWARE\DESCRIPTION\System", "SystemBiosDate" ) // Returned value = 1</code> |
| Tag      | <code>GetRegValueType( 2, "Control Panel\Desktop", "Smooth Scroll" ) // Returned value = 0</code>        |

## GetServerHostName

Gets the host name of the project's Server station.

| Function          | Group       | Execution   | Windows       | Embedded      | Thin Client |
|-------------------|-------------|-------------|---------------|---------------|-------------|
| GetServerHostName | System Info | Synchronous | Not supported | Not supported | Supported   |

## 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.

| Function     | Group       | Execution   | Windows   | Embedded  | Thin Client |
|--------------|-------------|-------------|-----------|-----------|-------------|
| GetTickCount | System Info | Synchronous | Supported | Supported | Supported   |

## 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

| Tag Name | Expression                                          |
|----------|-----------------------------------------------------|
| Tag      | <b>GetTickCount ( )</b> // Returned value = 9400907 |

## InfoAppAlrDir

Returns the file path of the project's Alarm sub-folder.

| Function      | Group       | Execution   | Windows   | Embedded  | Thin Client |
|---------------|-------------|-------------|-----------|-----------|-------------|
| InfoAppAlrDir | System Info | Synchronous | Supported | Supported | Supported   |

## Syntax

InfoAppAlrDir ( )

This function takes no parameters.

## Returned value

Returns the *Alarm* directory of the current project as a string.

## Examples

| Tag Name | Expression                                                                  |
|----------|-----------------------------------------------------------------------------|
| Tag      | <b>InfoAppAlrDir ( )</b> // Returned value = "C:\DemoApp\Alarm"             |
| Tag      | <b>InfoAppAlrDir ( )</b> // Returned value = "C:\Studio\Projects\App\Alarm" |

## InfoAppHSTDir

Returns the file path of the project's History sub-folder.

| Function      | Group       | Execution   | Windows   | Embedded  | Thin Client |
|---------------|-------------|-------------|-----------|-----------|-------------|
| InfoAppHSTDir | System Info | Synchronous | Supported | Supported | Supported   |

## Syntax

InfoAppHstDir ( )

This function takes no parameters.

## Returned value

Returns the *History* directory for the current project as a string.

## Examples

| Tag Name | Expression                                                                |
|----------|---------------------------------------------------------------------------|
| Tag      | <b>InfoAppHSTDir ( )</b> // Returned value = "C:\DemoApp\HST"             |
| Tag      | <b>InfoAppHSTDir ( )</b> // Returned value = "C:\Studio\Projects\App\HST" |

## InfoDiskFree

Returns free disk space on the local computer.

| Function     | Group       | Execution   | Windows   | Embedded      | Thin Client |
|--------------|-------------|-------------|-----------|---------------|-------------|
| InfoDiskFree | System Info | Synchronous | Supported | Not supported | Supported   |

## 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

| Tag Name | Expression                                                       |
|----------|------------------------------------------------------------------|
| Tag      | <b>InfoDiskFree( "C" )</b> // Returned value = 2803804605.000000 |

## InfoResources

Returns the local computer's disposable resources.

| Function      | Group       | Execution   | Windows   | Embedded  | Thin Client |
|---------------|-------------|-------------|-----------|-----------|-------------|
| InfoResources | System Info | Synchronous | Supported | Supported | Supported   |

## Syntax

InfoResources( *numSelect* )

### numSelect

A numeric flag that specifies which resource to examine:

| Value | Description          |
|-------|----------------------|
| 0     | System functions (%) |
| 1     | GDI functions (%)    |
| 2     | USER functions (%)   |
| 3     | Memory (in bytes)    |

## Examples

| Tag Name | Expression                                                    |
|----------|---------------------------------------------------------------|
| Tag      | <b>InfoResources( 0 )</b> // Returned value = 76.000000       |
| Tag      | <b>InfoResources( 1 )</b> // Returned value = 76.000000       |
| Tag      | <b>InfoResources( 2 )</b> // Returned value = 80.000000       |
| Tag      | <b>InfoResources( 3 )</b> // Returned value = 16150528.000000 |

 **Note:** The only valid selection on an 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.

| Function     | Group       | Execution   | Windows   | Embedded  | Thin Client |
|--------------|-------------|-------------|-----------|-----------|-------------|
| IsActiveXReg | System Info | Synchronous | Supported | Supported | Supported   |

## Syntax

IsActiveXReg( *numType*, *strProgIDorFileName* )

### numType

A numeric flag that specifies a format for the strProgIDorFileName parameter:

|   |                      |
|---|----------------------|
| 0 | Verify by Program ID |
| 1 | Verify by File Name  |

### strProgIDorFileName

The program ID or file path of the ActiveX control.

### Returned value

|   |                            |
|---|----------------------------|
| 0 | ActiveX is not registered. |
| 1 | ActiveX is registered.     |

### Examples

| Tag Name | Expression                                                                                |
|----------|-------------------------------------------------------------------------------------------|
| Tag      | <code>IsActiveXReg( 0, "ISSYMBOL.ISSymbolCtrl.1" ) // Returned value = 0</code>           |
| Tag      | <code>IsActiveXReg( 1, "C:\WinNT\system32\MediaPlayer.ocx" ) // Returned value = 1</code> |

### IsAppChangedOnServer

When executed on the Client, this function checks to see if the project files available on the Server are newer than the files currently on the Client.

| Function             | Group       | Execution   | Windows   | Embedded  | Thin Client |
|----------------------|-------------|-------------|-----------|-----------|-------------|
| IsAppChangedOnServer | System Info | Synchronous | Supported | Supported | Supported   |

### Syntax

```
InfoDiskFree({ "optTagUpdateTrigger" })
```

#### optTagUpdateTrigger

An alphanumeric string enclosed in quotes, or the name of a [String tag](#) that contains the desired string. This string, in turn, should be the name of the tag that will trigger the function — when the value of this tag changes, the function is automatically executed. (Normally, a function executes only when it is explicitly called, such as with the [Command animation](#).) To execute the function at a regular interval, you can use one of project's [system tags](#) like [Day](#) or [Month](#).

This parameter is optional.

### Returned value

|   |     |
|---|-----|
| 0 | No  |
| 1 | Yes |

### Notes

For this function, "Server" means the station that is actually running your project and has the TCP/IP Server module enabled, and "Client" means a Thin Client or Secure Viewer that is communicating with the Server via TCP/IP. For more information, see [Configuring a Web Solution](#).

 **Tip:** If the files on the Server are newer — that is, if this function returns TRUE — then you can use [ReloadAppFromServer](#) to update the Client.

### Examples

| Tag Name | Expression                                                                                                                       |
|----------|----------------------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>IsAppChangedOnServer ( )</code>                                                                                            |
| Tag      | <code>IsAppChangedOnServer ( "CheckVersion" ) // Function is automatically called when the value of CheckVersion changes.</code> |

### NoInputTime

Returns the time elapsed since the last keyboard action.

| Function    | Group       | Execution   | Windows   | Embedded  | Thin Client         |
|-------------|-------------|-------------|-----------|-----------|---------------------|
| NoInputTime | System Info | Synchronous | Supported | Supported | Keyboard input only |

## Syntax

NoInputTime({ "optTagUpdateTrigger" })

### optTagUpdateTrigger

Optional tag that triggers an update when this function is used in a [Text Data Link](#) animation. Every time this tag's value changes, the project triggers the function.

## Returned value

Returns the time (in seconds) since the last keyboard action.

## Examples

| Tag Name | Expression    |
|----------|---------------|
| Tag      | NoInputTime() |

 **Note:** You cannot implement this function directly from a Text object.

## ProductVersion

Returns the program version number.

| Function       | Group       | Execution   | Windows   | Embedded  | Thin Client |
|----------------|-------------|-------------|-----------|-----------|-------------|
| ProductVersion | System Info | Synchronous | Supported | Supported | Supported   |

## Syntax

ProductVersion()

This function takes no paramters.

## Returned value

Returns the program version number as a real number.

## Examples

| Tag Name | Expression                                    |
|----------|-----------------------------------------------|
| Tag      | ProductVersion() // Returned value = 7.000000 |

## 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.

| Function  | Group       | Execution   | Windows       | Embedded  | Thin Client   |
|-----------|-------------|-------------|---------------|-----------|---------------|
| RegSaveCE | System Info | Synchronous | Not supported | Supported | Not supported |

## Syntax

RegSaveCE()

This function takes no paramters.

## Returned value

|    |                                    |
|----|------------------------------------|
| 0  | Success.                           |
| -1 | Failed to save HKEY_CLASSES_ROOT.  |
| -2 | Failed to save HKEY_CURRENT_USER.  |
| -3 | Failed to save HKEY_LOCAL_MACHINE. |
| -4 | Failed to save HKEY_USERS.         |
| -5 | Function executed in NT platform.  |

 **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

| Tag Name | Expression                                               |
|----------|----------------------------------------------------------|
| n/a      | <b>RegSaveCE</b> ( ) // Returned value = 0 if successful |

### ReloadAppFromServer

When executed on the Client, this function reloads the necessary project files from the Server while maintaining the current state of the project on the Client.

| Function            | Group       | Execution   | Windows   | Embedded  | Thin Client |
|---------------------|-------------|-------------|-----------|-----------|-------------|
| ReloadAppFromServer | System Info | Synchronous | Supported | Supported | Supported   |

### Syntax

ReloadAppFromServer ( )

This function takes no parameters.

### Returned value

This function always returns 0.

### Notes

For this function, "Server" means the station that is actually running your project and has the TCP/IP Server module enabled, and "Client" means a Thin Client or Secure Viewer that is communicating with the Server via TCP/IP. For more information, see [Configuring a Web Solution](#).

 **Tip:** Before calling this function, you can use [IsAppChangedOnServer](#) to check the version of the project files that are already on the Client. If the files on the Client match the files on the Server, then you may choose not to call this function.

### Examples

| Tag Name | Expression                     |
|----------|--------------------------------|
| Tag      | <b>ReloadAppFromServer</b> ( ) |

### 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.

| Function      | Group       | Execution   | Windows   | Embedded  | Thin Client   |
|---------------|-------------|-------------|-----------|-----------|---------------|
| SaveAlarmFile | System Info | Synchronous | Supported | Supported | Not supported |

### 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 <b>OptRemotePath</b> 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

|   |                               |
|---|-------------------------------|
| 0 | Success                       |
| 1 | 2nd parameter is not a string |
| 2 | 2nd parameter is missing      |

### Examples

| Tag Name | Expression                                         |
|----------|----------------------------------------------------|
| Tag      | <code>SaveAlarmFile( 0 )</code>                    |
| Tag      | <code>SaveAlarmFile( 1 )</code>                    |
| Rag      | <code>SaveAlarmFile( 2, "Z:\Apps\AppDemo" )</code> |

### SetAppAlarmPath

Sets the *Alarm* path for the current project.

| Function        | Group       | Execution   | Windows   | Embedded  | Thin Client        |
|-----------------|-------------|-------------|-----------|-----------|--------------------|
| SetAppAlarmPath | System Info | Synchronous | Supported | Supported | Executed on Server |

### Syntax

SetAppAlarmPath(*strPath*)

#### **strPath**

The new *Alarm* path for the current project.

### Returned value

This function does not return any value.

### Examples

| Tag Name | Expression                                         |
|----------|----------------------------------------------------|
|          | <code>SetAppAlarmPath( "C:\Studio\Alarm\" )</code> |

### SetAppHSTPath

Sets the file path (directory) where Trend history files will be saved, in the proprietary format ( *.HST* ).

| Function      | Group       | Execution   | Windows   | Embedded  | Thin Client        |
|---------------|-------------|-------------|-----------|-----------|--------------------|
| SetAppHSTPath | System Info | Synchronous | Supported | Supported | Executed on Server |

### 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

| Tag Name | Expression                                       |
|----------|--------------------------------------------------|
|          | <code>SetAppHstPath("C:\Studio\History\")</code> |

## SetDateFormat

Sets the format and separator for the date string.

| Function      | Group       | Execution   | Windows   | Embedded  | Thin Client |
|---------------|-------------|-------------|-----------|-----------|-------------|
| SetDateFormat | System Info | Synchronous | Supported | Supported | Supported   |

## 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.

|     |                  |
|-----|------------------|
| DMY | Day, Month, Year |
| MDY | Month, Day, Year |
| YMD | Year, Month, Day |

## Returned value

|   |                   |
|---|-------------------|
| 0 | No error          |
| 1 | Invalid parameter |

## Examples

| Tag Name | Expression                                                    |
|----------|---------------------------------------------------------------|
| Tag      | <code>SetDateFormat( "/", "MDY" ) // Date = 04/18/2002</code> |
| Tag      | <code>SetDateFormat( ":", "MYD" ) // Date = 04:2002:18</code> |

## SetKeyboardLanguage

Sets the language of any Virtual Keyboards in the project.

| Function            | Group       | Execution   | Windows   | Embedded  | Thin Client |
|---------------------|-------------|-------------|-----------|-----------|-------------|
| SetKeyboardLanguage | System Info | Synchronous | Supported | Supported | Supported   |

## Syntax

`SetKeyboardLanguage( strLanguage)`

### strLanguage

String tag with the language code used for the Virtual Keyboards. The currently available options include:

|    |                   |
|----|-------------------|
| EN | English (default) |
| GE | German            |

## Returned value

|   |         |
|---|---------|
| 0 | Success |
| 1 | Error   |

## Examples

| Tag Name | Expression                                      |
|----------|-------------------------------------------------|
| Tag      | <code>SetKeyboardLanguage( "EN" )</code>        |
| Tag      | <code>SetKeyboardLanguage( tagLanguage )</code> |

## SetRegValue

Sets the value of a variable in the Windows registry.

| Function    | Group       | Execution   | Windows   | Embedded  | Thin Client   |
|-------------|-------------|-------------|-----------|-----------|---------------|
| SetRegValue | System Info | Synchronous | Supported | Supported | Not supported |

## Syntax

`SetRegValue( numMainKey, strKey, strVariableName, numType, strOrNumValue )`

### numMainKey

Numeric tag with the following possible values:

|   |                       |
|---|-----------------------|
| 0 | HKEY_LOCAL_MACHINE    |
| 1 | HKEY_CLASSES_ROOT     |
| 2 | HKEY_CURRENT_USER     |
| 3 | HKEY_USERS            |
| 4 | HKEY_CURRENT_CONFIG   |
| 5 | HKEY_PERFORMANCE_DATA |

### 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:

|   |        |
|---|--------|
| 0 | DWord  |
| 1 | String |

### strOrNumValue

Variable value.

## Returned value

|    |                                                                                     |
|----|-------------------------------------------------------------------------------------|
| 0  | Success.                                                                            |
| -1 | Invalid number of parameters or invalid Main Key.                                   |
| -2 | Invalid type.                                                                       |
| -3 | Failed to read the variable value; verify that you have the proper security rights. |

```
(0, "HARDWARE\DEVICEMAP\SERIALCOMM", "\Device\Serial1",
// Returned value = 0 if successful
(2, "Control Panel\Desktop", "Smooth Scroll", 0, 1) //
if successful
```

affect the Windows system configuration. You should be extremely *only* when you are certain about the configuration.

current project. The settings configured in the function are updated

| Execution   | Windows   | Embedded  | Thin Client        |
|-------------|-----------|-----------|--------------------|
| Synchronous | Supported | Supported | Executed on Server |

*optStrBackupURL, optStrPathFile, optNumHostPort, optStrSecondaryServerIP, optNumProtocolFlag, optNumGtwPort, optStrGtwIP, opt*

address (or hostname) of the computer where the TCP Server

the project's Web pages. The Thin Client will look for the Web not find them in the same URL written in the Address field of the

of the HTML file to be updated. If you specify only the file path of the HTML files in the specified file path will be updated.

ify an individual file (e.g., you only want to update one y useful for projects running on Windows Embedded

that the Thin Client must use to exchange data with the TCP

ver IP address. The Thin Client will attempt to connect to the n this IP Address if it is not able to connect to the TCP Server address specified in the strServerIP parameter.

Web Tunneling Gateway option, this parameter specifies whether er HTTP to exchange data with the Web Server or HTTPS (SSL – flag has the value 0, the Thin Client will use HTTP. If this flag ient will use HTTPS (SSL).

that the Thin Client must use to exchange data with the Web Tunneling Gateway.

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

| Error | Description                  |
|-------|------------------------------|
| 0     | No error                     |
| 1     | Invalid number of parameters |
| 2     | Invalid Server IP address 1  |
| 3     | Invalid URL                  |
| 4     | Invalid optional path        |
| 5     | No Web pages found           |

### Examples

| Tag Name | Expression                                                                                                                                                      |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Tag      | SetWebConfig( "192.168.1.28" )                                                                                                                                  |
| Tag      | SetWebConfig( "192.168.1.28", "http://192.168.1.28/" )                                                                                                          |
| Tag      | SetWebConfig( GetComputerIP(), "http://" + GetComputerIP() + "/" )                                                                                              |
| Tag      | SetWebConfig( "192.168.1.28", "http://192.168.1.28/", "C:\MyWebPages\" )                                                                                        |
| Tag      | SetWebConfig( "192.168.1.28", "http://192.168.1.28/", "C:\MyWebPages\", 1234 )                                                                                  |
| Tag      | SetWebConfig( "192.168.1.28", "http://200.0.0.10/", "C:\MyWebPages\", 1234, "192.168.1.29", 0, 80, "200.0.0.1", "200.0.0.10", "http://200.0.0.10/MyISSymbol/" ) |

### SNMPGet

Gets information from computers or network devices through the SNMP protocol.

| Function | Group       | Execution   | Windows   | Embedded      | Thin Client |
|----------|-------------|-------------|-----------|---------------|-------------|
| SNMPGet  | System Info | Synchronous | Supported | Not supported | Supported   |

## 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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named tag.

## Returned value

| Value | Description                                      |
|-------|--------------------------------------------------|
| 0     | No error                                         |
| -1    | Invalid number of parameters                     |
| -2    | Invalid parameter                                |
| -3    | Cannot connect to the remote machine             |
| -4    | Cannot connect to the remote machine             |
| -5    | GET operation failed                             |
| -6    | Invalid OID                                      |
| -7    | Invalid tag name                                 |
| -8    | Invalid tag type                                 |
| -9    | This function is not supported in the current OS |

## Examples

| Tag Name | Expression                                                                                                                                                                                                                           |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ErrorTag | <code>SNMPGet( "127.0.0.1", "public", ".1.3.6.1.2.1.1.1.0", "SysDescrTag" )</code> //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.

| Function | Group       | Execution   | Windows   | Embedded      | Thin Client |
|----------|-------------|-------------|-----------|---------------|-------------|
| SNMPSet  | System Info | Synchronous | Supported | Not supported | Supported   |

## Syntax

`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:

| Value | Type             | Description                      |
|-------|------------------|----------------------------------|
| 0     | OCTETSTRING      | An octet string variable         |
| 1     | INTEGER32        | A 32-bit signed integer variable |
| 2     | TIMETICKS        | A timeticks variable             |
| 3     | GAUGE32          | A gauge variable                 |
| 4     | COUNTER32        | A counter variable               |
| 5     | IPADDRESS        | An IP address variable           |
| 6     | OBJECTIDENTIFIER | An object identifier variable    |
| 7     | SEQUENCE         | An ASN sequence variable         |
| 8     | OPAQUE           | An opaque variable               |

## Returned value

| Value | Description                                      |
|-------|--------------------------------------------------|
| 0     | No error                                         |
| -1    | Invalid number of parameters                     |
| -2    | Invalid parameter                                |
| -3    | Cannot connect to the remote machine             |
| -4    | Cannot connect to the remote machine             |
| -5    | SET operation failed                             |
| -6    | Invalid OID                                      |
| -7    | Invalid tag name                                 |
| -8    | Invalid tag type                                 |
| -9    | This function is not supported in the current OS |

## Examples

| Tag Name | Expression                                                                                                                                                    |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
|          | <code>SNMPSet( "127.0.0.1", "public", ".1.3.6.1.2.1.1.1.0", 123, 1 ) //Sets an integer value of 123 to the specified OID on the localhost (127.0.0.1).</code> |

## WritePrivateProfileString

Writes a specified setting to the project viewer initialization file, using the standard `.ini` format.

| Function                  | Group  | Execution   | Windows   | Embedded  | Thin Client |
|---------------------------|--------|-------------|-----------|-----------|-------------|
| WritePrivateProfileString | System | Synchronous | Supported | Supported | Supported   |

## 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**

| Tag Name | Expression                                                                           |
|----------|--------------------------------------------------------------------------------------|
| Tag      | <code>WritePrivateProfileString( Section, Name, Value, FileName )</code>             |
| Tag      | <code>WritePrivateProfileString( "Options", "ds1", "Value", "C:\Viewer.ini" )</code> |

## 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.

| Function        | Group         | Execution   | Windows   | Embedded  | Thin Client |
|-----------------|---------------|-------------|-----------|-----------|-------------|
| ExecuteAlarmAck | Tags Database | Synchronous | Supported | Supported | Supported   |

### Syntax

```
ExecuteAlarmAck("strTagName", optStrComment, optStrAlarmType)
```

```
ExecuteAlarmAck("strTagName" ,{ | ,optStrComment{ | ,optStrAlarmType } })
```

#### strTagName

Name of the tag on which the alarm will be acknowledged.

 **Note:** The tag name must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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

| Value | Description                                                             |
|-------|-------------------------------------------------------------------------|
| 0     | Successfully executed.                                                  |
| -1    | Invalid number of parameters.                                           |
| -2    | Invalid tag name.                                                       |
| -3    | Executed, but did not wait for confirmation from Alarms task. See note. |

### 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:

```
ExecuteAlarmAck ("A", "Hi alarm on tag A", "Hi")
```

### ForceTagChange

Forces the database to write a value to a tag and act as if it were a tag change even if the new value is equal to the old value.

| Function       | Group         | Execution   | Windows   | Embedded  | Thin Client |
|----------------|---------------|-------------|-----------|-----------|-------------|
| ForceTagChange | Tags Database | Synchronous | Supported | Supported | Supported   |

## Syntax

`ForceTagChange( "strTagName", numValue )`

### strTagName

The name of the tag being forced to accept the new value.

 **Note:** The tag name must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named tag.

### numValue

The new value to be written to the specified tag.

## Returned value

This function does not return any value.

## Examples

| Tag Name | Expression                               |
|----------|------------------------------------------|
| n/a      | <code>ForceTagChange( "TagA", 5 )</code> |

## GetTagValue

Gets the value of the specified tag from the project tags database.

| Function    | Group         | Execution   | Windows   | Embedded  | Thin Client |
|-------------|---------------|-------------|-----------|-----------|-------------|
| GetTagValue | Tags Database | Synchronous | Supported | Supported | Supported   |

## Syntax

`GetTagValue( "strTagName", optNumRefresh )`

### strTagName

The name of the tag of which you want to get the value.

 **Note:** The tag name must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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

| Tag Name                    | Expression                                                      |
|-----------------------------|-----------------------------------------------------------------|
| requiredTag = 15            | <code>GetTagName( "requiredTag", Second ) // Return = 15</code> |
| pointerTag = "Required Tag" | <code>GetTagName( pointerTag, optNum ) // Return = 15</code>    |

| Tag Name                            | Expression |
|-------------------------------------|------------|
| requiredTag = 15<br>optNum = Second |            |

## SetTagValue

Sets the value of the specified tag in the project tags database.

| Function    | Group         | Execution   | Windows   | Embedded  | Thin Client |
|-------------|---------------|-------------|-----------|-----------|-------------|
| SetTagValue | Tags Database | Synchronous | Supported | Supported | Supported   |

## Syntax

```
SetTagValue("strTagName", TagValue)
```

### strTagName

The name of the tag that you want to set.

 **Note:** The tag name must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named tag.

### TagValue

The new value to be set to the specified tag.

## Returned value

| Value | Description      |
|-------|------------------|
| -1    | Invalid tag name |
| 0     | No error         |

## Examples

| Tag Name       | Expression                                                    |
|----------------|---------------------------------------------------------------|
| TagA           | <code>SetTagValue( "TagA", "Hello" ) // Return = Hello</code> |
| TagA           | <code>SetTagValue( "TagA", 123 ) // Return = 123</code>       |
| TagA TagB = 15 | <code>SetTagValue( "TagA", TagB ) // Return = 15</code>       |

## Loop functions

These functions are used to implement repeating, incrementing loop within a script.

### For ... Next

Implements a **For** ... **Next** loop within a Math worksheet or Command animation. The section of the script affected by the loop begins with the **For** ( ) function call and ends with the **Next** notation. The **Next** notation directs back to the beginning of the loop.

| Function | Group | Execution | Windows   | Embedded  | Thin Client |
|----------|-------|-----------|-----------|-----------|-------------|
| For      | Loop  | N/A       | Supported | Supported | Supported   |

### Syntax

**For**( *numInitialValue*, *numFinalValue*, *numStep* ) ... **Next**

#### **numInitialValue**

The initial step (increment) of the loop.

#### **numFinalValue**

The final step (increment) of the loop.

#### **numStep**

The step (increment) of the loop.

### Returned value

Returns the step on which the loop is currently running.

### Examples

| Tag Name | Expression             |
|----------|------------------------|
| Tag      | <b>For</b> ( 1, 5, 1 ) |
| Next     |                        |

 **Note:** You must partner every For function with a Next notation. As shown in the example, you must place the Next notation in the tag field of the math script.

## 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.

| Function       | Group | Execution   | Windows   | Embedded      | Thin Client        |
|----------------|-------|-------------|-----------|---------------|--------------------|
| ODBCBeginTrans | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

#### Syntax

ODBCBeginTrans(*numHandler*)

#### **numHandler**

The handler returned by the [ODBCOpen](#) function.

#### Returned value

|   |                             |
|---|-----------------------------|
| 0 | Success                     |
| 1 | Invalid handler             |
| 2 | Database not open           |
| 3 | Error beginning transaction |

#### Examples

| Tag Name | Expression          |
|----------|---------------------|
| Tag      | ODBCBeginTrans( 5 ) |

### ODBCBindCol

Binds a column to a tag.

| Function    | Group | Execution   | Windows   | Embedded      | Thin Client        |
|-------------|-------|-------------|-----------|---------------|--------------------|
| ODBCBindCol | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

#### 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
- 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

|   |                                           |
|---|-------------------------------------------|
| 0 | Success                                   |
| 1 | Invalid Handler                           |
| 2 | Invalid parameter type                    |
| 3 | One of the parameters has an empty string |
| 4 | ColType contains an invalid type          |

### Notes

Every time you finish binding columns, you must call the [ODBCQuery](#) function.

### Examples

| Tag Name | Expression                                                           |
|----------|----------------------------------------------------------------------|
| Tag      | <code>ODBCBindCol( 5, "OrderDate", "SQL_DATE", "Order_Date" )</code> |

See also: [ODBCUnbindCol\(\)](#)

### ODBCCanAppend

Returns whether the database will allow you to add new records.

| Function      | Group | Execution   | Windows   | Embedded      | Thin Client        |
|---------------|-------|-------------|-----------|---------------|--------------------|
| ODBCCanAppend | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

### Syntax

`ODBCCanAppend( numHandler )`

### numHandler

The handler returned by the [ODBCOpen](#) function.

### Returned value

|          |                                               |
|----------|-----------------------------------------------|
| 0        | Database does not allow appending new records |
| Non-Zero | Database does allow appending new records     |

### Examples

| Tag Name | Expression                      |
|----------|---------------------------------|
| Tag      | <code>ODBCCanAppend( 5 )</code> |

## ODBCCanTransact

Returns whether the database allows transactions.

| Function        | Group | Execution   | Windows   | Embedded      | Thin Client        |
|-----------------|-------|-------------|-----------|---------------|--------------------|
| ODBCCanTransact | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

### Syntax

ODBCCanTransact(*numHandler*)

#### numHandler

The handler returned by the [ODBCOpen](#) function.

### Returned value

|          |                                       |
|----------|---------------------------------------|
| 0        | Database does not allow transactions. |
| Non-Zero | Database does allow transactions.     |

### Examples

| Tag Name | Expression           |
|----------|----------------------|
| Tag      | ODBCCanTransact( 2 ) |

## ODBCCanUpdate

Returns whether the database can be updated.

| Function      | Group | Execution   | Windows   | Embedded      | Thin Client        |
|---------------|-------|-------------|-----------|---------------|--------------------|
| ODBCCanUpdate | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

### Syntax

ODBCCanUpdate(*numHandler*)

#### numHandler

The handler returned by the [ODBCOpen](#) function.

### Returned value

|          |                                  |
|----------|----------------------------------|
| 0        | Database does not allow updates. |
| Non-Zero | Database does allow updates.     |

### Examples

| Tag Name | Expression         |
|----------|--------------------|
| Tag      | ODBCCanUpdate( 6 ) |

## ODBCClose

Closes a connection to the database.

| Function  | Group | Execution   | Windows   | Embedded      | Thin Client        |
|-----------|-------|-------------|-----------|---------------|--------------------|
| ODBCClose | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

### Syntax

ODBCClose(*numHandler*)

#### numHandler

The handler returned by the [ODBCOpen](#) function.

## Returned value

|   |                 |
|---|-----------------|
| 0 | Success         |
| 1 | Invalid Handler |

## Examples

| Tag Name | Expression                  |
|----------|-----------------------------|
| Tag      | <code>ODBCClose( 5 )</code> |

## **ODBCCommitTrans**

Commits a transaction. Call this function upon completing transactions.

| Function        | Group | Execution   | Windows   | Embedded      | Thin Client        |
|-----------------|-------|-------------|-----------|---------------|--------------------|
| ODBCCommitTrans | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

## Syntax

`ODBCCommitTrans( numHandler )`

### **numHandler**

The handler returned by the [ODBCOpen](#) function.

## Returned value

|   |                              |
|---|------------------------------|
| 0 | Success                      |
| 1 | Invalid handler              |
| 2 | Database not open            |
| 3 | Error committing transaction |

## Examples

| Tag Name | Expression                        |
|----------|-----------------------------------|
| Tag      | <code>ODBCCommitTrans( 1 )</code> |

## **ODBCDelete**

Deletes the current record.

| Function   | Group | Execution   | Windows   | Embedded      | Thin Client        |
|------------|-------|-------------|-----------|---------------|--------------------|
| ODBCDelete | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

## Syntax

`ODBCDelete( numHandler )`

### **numHandler**

The handler returned by the [ODBCOpen](#) function.

## Returned value

|   |                   |
|---|-------------------|
| 0 | Success           |
| 1 | Invalid handler   |
| 2 | Database not open |
| 3 | Delete error      |

## 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

| Tag Name | Expression      |
|----------|-----------------|
| Tag      | ODBCDelete( 5 ) |

### **ODBCExecuteSQL**

Directly executes an SQL statement.

| Group | Execution   | Windows PC | Windows CE    | Thin Client        |
|-------|-------------|------------|---------------|--------------------|
| ODBC  | Synchronous | Supported  | Not supported | Executed on Server |

## Syntax

ODBCExecuteSQL( *numHandler*, *strSqlCommand* )

### **numHandler**

The handler returned by the [ODBCOpen](#) function.

### **strSqlCommand**

A valid SQL statement.

## Returned value

|   |                             |
|---|-----------------------------|
| 0 | Success                     |
| 1 | Invalid handler             |
| 2 | Database not open           |
| 3 | Invalid parameter           |
| 4 | Error executing SQL command |

 **Note:** This function does not return any records, regardless of the statement that is executed.

## Examples

| Tag Name | Expression             |
|----------|------------------------|
| Tag      | ODBCExecuteSQL( 3, " ) |
| Tag      | ODBCExecuteSQL( 4, " ) |

### **ODBCInsert**

Inserts a new record in the database.

| Function   | Group | Execution   | Windows   | Embedded      | Thin Client        |
|------------|-------|-------------|-----------|---------------|--------------------|
| ODBCInsert | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

## Syntax

ODBCInsert( *numHandler* )

### **numHandler**

The handler returned by the [ODBCOpen](#) function.

## Returned value

|   |                   |
|---|-------------------|
| 0 | Success           |
| 1 | Invalid handler   |
| 2 | Database not open |
| 3 | Insert error      |

## Notes

This function uses the values of the tags bound by the [ODBCBindCol](#) function to create the new record.

## Examples

| Tag Name | Expression                   |
|----------|------------------------------|
| Tag      | <code>ODBCInsert( 7 )</code> |

## ODBCIsBOF

Returns whether you have gone above the first record of the record set. (Call this function before scrolling from record to record.)

| Function  | Group | Execution   | Windows   | Embedded      | Thin Client        |
|-----------|-------|-------------|-----------|---------------|--------------------|
| ODBCIsBOF | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

## Syntax

`ODBCIsBOF( numHandler )`

### numHandler

The handler returned by the [ODBCOpen](#) function.

## Returned value

|          |                                                                             |
|----------|-----------------------------------------------------------------------------|
| 0        | Record found                                                                |
| non-zero | Record set contains no records or you move backward, above the first record |

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

| Tag Name | Expression                  |
|----------|-----------------------------|
| Tag      | <code>ODBCIsBOF( 1 )</code> |

## ODBCIsDeleted

Reports whether the current record was deleted.

| Function      | Group | Execution   | Windows   | Embedded      | Thin Client        |
|---------------|-------|-------------|-----------|---------------|--------------------|
| ODBCIsDeleted | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

## Syntax

`ODBCIsDeleted( numHandler )`

### numHandler

The handler returned by the [ODBCOpen](#) function.

## Returned value

|          |                                                   |
|----------|---------------------------------------------------|
| 0        | Record set is not positioned on a deleted record. |
| non-zero | Record set is positioned on a deleted record.     |

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

| Tag Name | Expression                      |
|----------|---------------------------------|
| Tag      | <code>ODBCIsDeleted( 8 )</code> |

## ODBCIsEOF

Reports whether you have gone beyond the last record of the record set. (Call this function as you scroll from record to record.)

| Function  | Group | Execution   | Windows   | Embedded      | Thin Client        |
|-----------|-------|-------------|-----------|---------------|--------------------|
| ODBCIsEOF | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

### Syntax

ODBCIsEOF( *numHandler* )

#### **numHandler**

The handler returned by the [ODBCOpen](#) function.

### Returned value

|          |                                                                             |
|----------|-----------------------------------------------------------------------------|
| 0        | Record found.                                                               |
| non-zero | Record set contains no records or you moved forward, after the last record. |

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

| Tag Name | Expression     |
|----------|----------------|
| Tag      | ODBCIsEOF( 5 ) |

## ODBCIsFieldNULL

Reports whether a specified field in a record set was flagged as **NULL**.

| Function        | Group | Execution   | Windows   | Embedded      | Thin Client        |
|-----------------|-------|-------------|-----------|---------------|--------------------|
| ODBCIsFieldNULL | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

### Syntax

ODBCIsFieldNULL( *numHandler*, *strColName* )

#### **numHandler**

The handler returned by the [ODBCOpen](#) function.

#### **strColName**

The column name.

### Returned value

|          |                                             |
|----------|---------------------------------------------|
| 0        | The specified field is not flagged as Null. |
| Non-Zero | The specified field is flagged as Null.     |

### Examples

| Tag Name | Expression                           |
|----------|--------------------------------------|
| Tag      | ODBCIsFieldNULL( 7, "CustomerName" ) |
| Tag      | ODBCIsFieldNULL( 3, "CompanyName" )  |

## ODBCIsFieldNullable

Reports whether a specified field is nullable (i.e., can be set to a **NULL** value).

| Function        | Group | Execution   | Windows   | Embedded      | Thin Client        |
|-----------------|-------|-------------|-----------|---------------|--------------------|
| ODBCIsFieldNull | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

### Syntax

ODBCIsFieldNullable(*numHandler*, *strColName*)

#### numHandler

The handler returned by the [ODBCOpen](#) function.

#### strColName

The column name.

### Returned value

|          |                                                 |
|----------|-------------------------------------------------|
| 0        | The specified field is not flagged as Nullable. |
| Non-Zero | The specified field is flagged as Nullable.     |

### Examples

| Tag Name | Expression                        |
|----------|-----------------------------------|
| Tag      | ODBCIsFieldNullable( 1, "Price" ) |
| Tag      | ODBCIsFieldNullable( 1, "Model" ) |

## ODBCMove

Moves the current record pointer within a record set, either forward or backward.

| Function | Group | Execution   | Windows   | Embedded      | Thin Client        |
|----------|-------|-------------|-----------|---------------|--------------------|
| ODBCMove | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

### 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

|   |                   |
|---|-------------------|
| 0 | Success           |
| 1 | Invalid handler   |
| 2 | Database not open |
| 3 | Move error        |

## Examples

| Tag Name | Expression                    |
|----------|-------------------------------|
| Tag      | <code>ODBCMove( 2, 3 )</code> |
| Tag      | <code>ODBCMove( 8, 2 )</code> |

### **ODBCMoveFirst**

Moves to the first record within the record set.

| Function      | Group | Execution   | Windows   | Embedded      | Thin Client        |
|---------------|-------|-------------|-----------|---------------|--------------------|
| ODBCMoveFirst | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

### Syntax

`ODBCMoveFirst( numHandler)`

#### **numHandler**

The handler returned by the [ODBCOpen](#) function.

### Returned value

|   |                    |
|---|--------------------|
| 0 | Success.           |
| 1 | Invalid handler.   |
| 2 | Database not open. |
| 3 | Move error.        |

## Examples

| Tag Name | Expression                      |
|----------|---------------------------------|
| Tag      | <code>ODBCMoveFirst( 4 )</code> |

### **ODBCMoveLast**

Moves to the last record within the record set.

| Function     | Group | Execution   | Windows   | Embedded      | Thin Client        |
|--------------|-------|-------------|-----------|---------------|--------------------|
| ODBCMoveLast | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

### Syntax

`ODBCMoveLast( numHandler)`

#### **numHandler**

The handler returned by the [ODBCOpen](#) function.

### Returned value

|   |                   |
|---|-------------------|
| 0 | Success           |
| 1 | Invalid handler   |
| 2 | Database not open |
| 3 | Move error        |

## Examples

| Tag Name | Expression                     |
|----------|--------------------------------|
| Tag      | <code>ODBCMoveLast( 7 )</code> |

## ODBCMoveNext

Moves to the next record within the record set.

| Function     | Group | Execution   | Windows   | Embedded      | Thin Client        |
|--------------|-------|-------------|-----------|---------------|--------------------|
| ODBCMoveNext | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

### Syntax

ODBCMoveNext ( *numHandler* )

#### numHandler

The handler returned by the [ODBCOpen](#) function.

### Returned value

|   |                           |
|---|---------------------------|
| 0 | Success                   |
| 1 | Invalid handler           |
| 2 | Database not open         |
| 3 | End of record set reached |
| 4 | Move error                |

### Examples

| Tag Name | Expression         |
|----------|--------------------|
| Tag      | ODBCMoveNext ( 9 ) |

## ODBCMovePrev

Moves to the next record within the record set.

| Function     | Group | Execution   | Windows   | Embedded      | Thin Client        |
|--------------|-------|-------------|-----------|---------------|--------------------|
| ODBCMovePrev | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

### Syntax

ODBCMovePrev ( *numHandler* )

#### numHandler

The handler returned by the [ODBCOpen](#) function.

### Returned value

|   |                                 |
|---|---------------------------------|
| 0 | Success                         |
| 1 | Invalid handler                 |
| 2 | Database not open               |
| 3 | Beginning of record set reached |
| 4 | Move error                      |

### Examples

| Tag Name | Expression         |
|----------|--------------------|
| Tag      | ODBCMovePrev ( 2 ) |

## ODBCOpen

Opens a connection to the database and returns a numeric handler to be used by other ODBC functions.

| Function | Group | Execution   | Windows   | Embedded      | Thin Client        |
|----------|-------|-------------|-----------|---------------|--------------------|
| ODBCOpen | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

## 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

|          |                                                          |
|----------|----------------------------------------------------------|
| <i>n</i> | On success, returns the handler to identify the database |
| -1       | Invalid parameter                                        |
| -2       | DSN or TableName contain an empty string                 |

## 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

| Tag Name | Expression                                                                |
|----------|---------------------------------------------------------------------------|
| Tag      | ODBCOpen( "MyDSNFile", "Alex", "", "Table1", "Name='Mayer'", "Name ASC" ) |
| Tag      | ODBCOpen( "DSNFileName", "Robert", "Robot", "Table1", "", "" )            |

## ODBCQuery

Retrieves the currently selected record from a database.

| Function  | Group | Execution   | Windows   | Embedded      | Thin Client        |
|-----------|-------|-------------|-----------|---------------|--------------------|
| ODBCQuery | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

## Syntax

ODBCQuery( *numHandler* )

### numHandler

The handler returned by the [ODBCOpen](#) function.

## Returned value

|   |                         |
|---|-------------------------|
| 0 | Success                 |
| 1 | Invalid handler         |
| 2 | No columns bound        |
| 3 | Cannot open database    |
| 4 | Cannot restart database |

|   |             |
|---|-------------|
| 5 | Query error |
|---|-------------|

### Notes

If you modify the [column binding](#), or if you modify the [filter](#) and [sort](#), then you must call this function again.

### Examples

| Tag Name | Expression                  |
|----------|-----------------------------|
| Tag      | <code>ODBCQuery( 6 )</code> |

### ODBCRollBack

Reverses the changes made during a transaction.

| Function     | Group | Execution   | Windows   | Embedded      | Thin Client        |
|--------------|-------|-------------|-----------|---------------|--------------------|
| ODBCRollBack | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

### Syntax

`ODBCRollBack( numHandler )`

#### numHandler

The handler returned by the [ODBCOpen](#) function.

### Returned value

|   |                                |
|---|--------------------------------|
| 0 | Success                        |
| 1 | Invalid handler                |
| 2 | Database no open               |
| 3 | Error rolling back transaction |

### Examples

| Tag Name | Expression                     |
|----------|--------------------------------|
| Tag      | <code>ODBCRollback( 4 )</code> |

### ODBCSetFieldNULL

Flags a field data member in the record set as **NULL** (specifically having no value) or as **non-NULL**.

| Function         | Group | Execution   | Windows   | Embedded      | Thin Client        |
|------------------|-------|-------------|-----------|---------------|--------------------|
| ODBCSetFieldNULL | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

### Syntax

`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

|   |                 |
|---|-----------------|
| 0 | Success         |
| 1 | Invalid handler |

|   |                     |
|---|---------------------|
| 2 | Database not open   |
| 3 | Invalid parameter   |
| 4 | Invalid column name |

**Examples**

| Tag Name | Expression                              |
|----------|-----------------------------------------|
| Tag      | ODBCSetFieldNULL( 2, "Price", 1 )       |
| Tag      | ODBCSetFieldNULL( 4, "CompanyName", 0 ) |

**ODBCSetFilter**

Constrains the records selected in a database.

| Function      | Group | Execution   | Windows   | Embedded      | Thin Client        |
|---------------|-------|-------------|-----------|---------------|--------------------|
| ODBCSetFilter | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

**Syntax**

ODBCSetFilter( *numHandler*, *strFilter* )

**numHandler**

The handler returned by the [ODBCOpen](#) function.

**strFilter**

The a SQL **WHERE** clause.

**Returned value**

|   |                   |
|---|-------------------|
| 0 | Success           |
| 1 | Invalid handler   |
| 2 | Invalid parameter |

**Notes**

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**

| Tag Name | Expression                          |
|----------|-------------------------------------|
| Tag      | ODBCSetFilter( 3, "Name='Morgan'" ) |

**ODBCSetSort**

Sorts the records selected in a database.

| Function    | Group | Execution   | Windows   | Embedded      | Thin Client        |
|-------------|-------|-------------|-----------|---------------|--------------------|
| ODBCSetSort | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

**Syntax**

ODBCSetSort( *numHandler*, *strSort* )

**numHandler**

The handler returned by the [ODBCOpen](#) function.

**strSort**

The SQL **ORDER BY** clause.

**Returned value**

|   |         |
|---|---------|
| 0 | Success |
|---|---------|

|   |                        |
|---|------------------------|
| 1 | Invalid handler        |
| 2 | Invalid parameter type |

### Notes

You can use this feature to sort the records in one or more columns. Remember to call [ODBCQuery](#) after calling this function.

### Examples

| Tag Name | Expression                                 |
|----------|--------------------------------------------|
| Tag      | <code>ODBCSetSort( 5, "Name DESC" )</code> |

### ODBCUnbindCol

Unbinds a column that was previously bound using the [ODBCBindCol](#) function.

| Function      | Group | Execution   | Windows   | Embedded      | Thin Client        |
|---------------|-------|-------------|-----------|---------------|--------------------|
| ODBCUnbindCol | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

### Syntax

`ODBCUnbindCol( numHandler, strColName)`

#### numHandler

The handler returned by the [ODBCOpen](#) function.

#### strColName

The column name.

### Returned value

|   |                        |
|---|------------------------|
| 0 | Success                |
| 1 | Invalid handler        |
| 2 | Invalid parameter type |
| 3 | Column not bound       |

### Examples

| Tag Name | Expression                              |
|----------|-----------------------------------------|
| Tag      | <code>ODBCUnbindCol( 7, "Name" )</code> |

### ODBCUpdate

Updates the current record.

| Function   | Group | Execution   | Windows   | Embedded      | Thin Client        |
|------------|-------|-------------|-----------|---------------|--------------------|
| ODBCUpdate | ODBC  | Synchronous | Supported | Not supported | Executed on Server |

### Syntax

`ODBCUpdate( numHandler)`

#### numHandler

The handler returned by the [ODBCOpen](#) function.

### Returned value

|   |                   |
|---|-------------------|
| 0 | Success           |
| 1 | Invalid handler   |
| 2 | Database not open |

|   |              |
|---|--------------|
| 3 | Update error |
|---|--------------|

### Notes

This function uses the values of the tags bound by the `ODBCBindCol` function to update the current record.

### Examples

| Tag Name | Expression                   |
|----------|------------------------------|
| Tag      | <code>ODBCUpdate( 1 )</code> |

## Mail 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.

| Function | Group | Execution   | Windows   | Embedded                         | Thin Client |
|----------|-------|-------------|-----------|----------------------------------|-------------|
| CnfEmail | Email | Synchronous | Supported | Supported, except for encryption | Supported   |

### Syntax

```
CnfEmail(strSMTP, strFrom, strPOP3, strUser, strPassword, optNumTimeout, optNumAuthType, optStrSMTPUser, optStrSMTPPassword)
```

```
CnfEmail(strSMTP, strFrom, strPOP3, strUser, strPassword{ 1 | 2 } , optNumTimeout{ 1 | 2 } , { optNumAuthType | 1 | 2 } , 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.)

 **Note:** Encryption via TLS/SSL is not supported in projects running on Windows Embedded target systems.

### Returned value

| Value | Description                                               |
|-------|-----------------------------------------------------------|
| 0     | Success                                                   |
| 1     | Invalid format for strSMTP                                |
| 2     | Invalid format for strFrom                                |
| 3     | Invalid format for strPOP3                                |
| 4     | Invalid format for strUser                                |
| 5     | Invalid format for strPassword                            |
| 6     | Invalid format for optNumTimeout                          |
| 7     | Wrong number of parameters                                |
| 8     | Error getting host IP address (invalid POP3 server)       |
| 9     | Error connecting to POP3 server                           |
| 10    | Error sending username                                    |
| 11    | Error sending password                                    |
| 12    | SMTP server does not support selected authentication mode |
| 13    | Invalid SMTP username                                     |
| 14    | Authentication failed                                     |

### 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.

| Function              | Group | Execution   | Windows   | Embedded  | Thin Client |
|-----------------------|-------|-------------|-----------|-----------|-------------|
| GetStatusSendEmailExt | Email | Synchronous | Supported | Supported | Supported   |

### 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

|    |                                                                                     |
|----|-------------------------------------------------------------------------------------|
| -2 | Incorrect version of the <b>INDMail.DLL</b> library.                                |
| -1 | The <b>INDMail.DLL</b> library is corrupted.                                        |
| 0  | SendEmailExt function is not being executed.                                        |
| 1  | Still sending last email. Cannot execute the SendEmailExt function.                 |
| 2  | Last email was sent successfully. You can execute the SendEmailExt function again.  |
| 3  | There was an error sending the last email. Execute the SendEmailExt function again. |

### Examples

| Tag Name | Expression                      |
|----------|---------------------------------|
| Tag      | GetStatusSendEmailExt( Second ) |
| Tag      | GetStatusSendEmailExt( )        |

## SendEmail

This function sends an email message.

| Function  | Group | Execution   | Windows   | Embedded  | Thin Client |
|-----------|-------|-------------|-----------|-----------|-------------|
| SendEmail | Email | Synchronous | Supported | Supported | Supported   |

### Syntax

SendEmail( *strSubject*, *strMessage*, *strTo* )

SendEmail( *strSubject*, *strMessage*, *strTo* )

#### strSubject

The subject of the email.

#### strMessage

The message body of the email, up to 255 characters long.

#### strTo

The email address of the intended recipient.

### Returned value

| Value | Description                   |
|-------|-------------------------------|
| 0     | Success                       |
| 1     | Invalid format for strSubject |
| 2     | Invalid format for strMessage |
| 3     | Invalid format for strTo      |

| Value | Description                                               |
|-------|-----------------------------------------------------------|
| 4     | Wrong number of parameters                                |
| 5     | Start socket error                                        |
| 6     | Error getting host IP Address (i.e., invalid SMTP server) |
| 7     | Error connecting to SMTP server                           |
| 8     | Error sending HELO command (i.e., initialization)         |
| 9     | Error sending MAIL command (i.e., the "From" address)     |
| 10    | Error sending RCPT command (i.e., the "To" address)       |
| 11    | Error sending DATA (i.e., the message body)               |
| 12    | Error sending SMTP authentication command                 |
| 13    | Invalid username                                          |
| 14    | Invalid password                                          |

### Notes

Before you can send any email, you must first configure your project's email settings either by using the *E-mail Settings* dialog or by calling the `CnfEmail` function. Incorrect settings can result in several different error codes (see "Returned value" above).

Also, `SendEmail` cannot be used when encryption via TLS/SSL is enabled or to send an email that contains Unicode characters. Use the `SendEmailExt` function instead.

### Examples

```
SendEmail("Hi!", "How are you?", "rogers@pnd.net")
SendEmail(statusSummary, statusDetail, adminAddress)
```

### SendEmailExt

Sends e-mail messages with attached files.

| Function     | Group | Execution    | Windows   | Embedded  | Thin Client |
|--------------|-------|--------------|-----------|-----------|-------------|
| SendEmailExt | Email | Asynchronous | Supported | Supported | Supported   |

### Syntax

```
SendEmailExt(strSubject, strMessage, strTo, optStrCc, optStrBcc, optStrFile1,
... , optStrFileN)
```

#### strSubject

The e-mail subject (up to 255 characters).

#### strMessage

The e-mail 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.

### optStrFile (1-N)

Complete file paths and names of file attachments.

### Returned value

|    |                                                                            |
|----|----------------------------------------------------------------------------|
| -4 | Some of the attached files were not found.                                 |
| -3 | Wrong number of parameters (at least three parameters are required).       |
| -2 | The <b>INDMail.DLL</b> library version is incorrect.                       |
| -1 | The <b>INDMail.DLL</b> library is corrupted.                               |
| 0  | Success                                                                    |
| 1  | Cannot execute the function because the last e-mail has not been sent yet. |
| 2  | Internal error                                                             |

### Notes

Before you can send any email, you must first use the [CnfEmail](#) function to configure the email settings. Incorrect settings can result in several different error codes (see "Returned value" above).

### Examples

| Tag Name | Expression                                                                                                                        |
|----------|-----------------------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>SendEmailExt( "Subject", "Message", "Sam@universe.com", "", "", "C:\Projects eport.txt" )</code>                            |
| Tag      | <code>SendEmailExt( "Subject", "Message", "David@Ohio.net", "Ted@Austin.com", "Bart@Springfield.gov", "C:\TechRef51.doc" )</code> |

## 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.

| Function  | Group   | Execution   | Windows              | Embedded  | Thin Client |
|-----------|---------|-------------|----------------------|-----------|-------------|
| DialError | Dial-up | Synchronous | Supported (see note) | Supported | Supported   |

### 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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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

| Value | Description                                       |
|-------|---------------------------------------------------|
| 0     | OK                                                |
| -1    | Error: INDRas.DLL not found.                      |
| -2    | Error: INDRas.DLL damaged.                        |
| -3    | Error: invalid number of parameters (minimum=2).  |
| -4    | Invalid value for the numType parameter (0 or 1). |
| -5    | PhoneBook or Modem does not exist.                |
| 600   | An operation is pending.                          |
| 601   | The port handle is invalid.                       |
| 602   | The port is already open.                         |
| 603   | Caller's buffer is too small.                     |
| 604   | Wrong information specified.                      |

| Value | Description                                                                 |
|-------|-----------------------------------------------------------------------------|
| 605   | Cannot set port information.                                                |
| 606   | The port is not connected                                                   |
| 607   | The event is invalid.                                                       |
| 608   | The device does not exist.                                                  |
| 609   | The device type does not exist.                                             |
| 610   | The buffer is invalid.                                                      |
| 611   | The route is not available.                                                 |
| 612   | The route is not allocated.                                                 |
| 613   | Invalid compression specified.                                              |
| 614   | Out of buffers.                                                             |
| 615   | The port was not found.                                                     |
| 616   | An asynchronous request is pending.                                         |
| 617   | The port or device is already disconnecting.                                |
| 618   | The port is not open.                                                       |
| 619   | The port is disconnected.                                                   |
| 620   | There are no endpoints.                                                     |
| 621   | Cannot open the phone book file.                                            |
| 622   | Cannot load the phone book file.                                            |
| 623   | Cannot find the phone book entry.                                           |
| 624   | Cannot write the phone book file.                                           |
| 625   | Invalid information found in the phone book file.                           |
| 626   | Cannot load a string.                                                       |
| 627   | Cannot find key.                                                            |
| 628   | The port was disconnected.                                                  |
| 629   | The data link was terminated by the remote machine.                         |
| 630   | The port was disconnected due to hardware failure.                          |
| 631   | The port was disconnected by the user.                                      |
| 632   | The structure size is incorrect.                                            |
| 633   | The port is already in use or is not configured for Remote Access dial out. |
| 634   | Cannot register your computer on on the remote network.                     |
| 635   | Unknown error.                                                              |
| 636   | The wrong device is attached to the port.                                   |
| 637   | The string could not be converted.                                          |
| 638   | The request has timed out.                                                  |
| 639   | No asynchronous net available.                                              |
| 640   | A NetBIOS error occurred.                                                   |
| 641   | The server cannot allocate NetBIOS resources needed to support the client.  |
| 642   | One of your NetBIOS names is already registered on the remote network.      |
| 643   | A network adapter at the server failed.                                     |
| 644   | You will not receive network message pop-ups.                               |
| 645   | Internal authentication error.                                              |
| 646   | The account is not permitted to log on at this time of day.                 |

| Value | Description                                                                               |
|-------|-------------------------------------------------------------------------------------------|
| 647   | The account is disabled.                                                                  |
| 648   | The password has expired.                                                                 |
| 649   | The account does not have Remote Access permission.                                       |
| 650   | The Remote Access server is not responding.                                               |
| 651   | Your modem (or other connecting device) has reported an error.                            |
| 652   | Unrecognized response from the device.                                                    |
| 653   | A macro required by the device was not found in the device <b>.INF</b> file section.      |
| 654   | A command or response in the device <b>.INF</b> file section refers to an undefined acro. |
| 655   | The <message macro was not found in the device <b>.INF</b> file section.                  |
| 656   | The <defaultoff macro in the device <b>.INF</b> file section contains an undefined macro. |
| 657   | The device <b>.INF</b> file could not be opened.                                          |
| 658   | The device name in the device <b>.INF</b> or media <b>.INI</b> file is too long.          |
| 659   | The media <b>.INI</b> file refers to an unknown device name.                              |
| 660   | The device <b>.INF</b> file contains no responses for the command.                        |
| 661   | The device <b>.INF</b> file is missing a command.                                         |
| 662   | Attempted to set a macro not listed in device <b>.INF</b> file section.                   |
| 663   | The media <b>.INI</b> file refers to an unknown device type.                              |
| 664   | Cannot allocate memory.                                                                   |
| 665   | The port is not configured for Remote Access.                                             |
| 666   | Your modem (or other connecting device) is not functioning.                               |
| 667   | Cannot read the media <b>.INI</b> file.                                                   |
| 668   | The connection dropped.                                                                   |
| 669   | The usage parameter in the media <b>.INI</b> file is invalid.                             |
| 670   | Cannot read the section name from the media <b>.INI</b> file.                             |
| 671   | Cannot read the device type from the media <b>.INI</b> file.                              |
| 672   | Cannot read the device name from the media <b>.INI</b> file.                              |
| 673   | Cannot read the usage from the media <b>.INI</b> file.                                    |
| 674   | Cannot read the maximum connection BPS rate from the media <b>.INI</b> file.              |
| 675   | Cannot read the maximum carrier BPS rate from the media <b>.INI</b> file.                 |
| 676   | The line is busy.                                                                         |
| 677   | A person answered instead of a modem.                                                     |
| 678   | There is no answer.                                                                       |
| 679   | Cannot detect carrier.                                                                    |
| 680   | There is no dial tone.                                                                    |
| 681   | General error reported by device.                                                         |
| 682   | ERROR_WRITING_SECTIONNAME                                                                 |
| 683   | ERROR_WRITING_DEVICETYPE                                                                  |
| 684   | ERROR_WRITING_DEVICENAME                                                                  |
| 685   | ERROR_WRITING_MAXCONNECTBPS                                                               |
| 686   | ERROR_WRITING_MAXCARRIERBPS                                                               |
| 687   | ERROR_WRITING_USAGE                                                                       |
| 688   | ERROR_WRITING_DEFAULTOFF                                                                  |

| Value | Description                                                                                                                |
|-------|----------------------------------------------------------------------------------------------------------------------------|
| 689   | ERROR_READING_DEFAULTOFF                                                                                                   |
| 690   | ERROR_EMPTY_INI_FILE                                                                                                       |
| 691   | Access denied because username and/or password is invalid on the domain.                                                   |
| 692   | Hardware failure in port or attached device.                                                                               |
| 693   | ERROR_NOT_BINARY_MACRO                                                                                                     |
| 694   | ERROR_DCB_NOT_FOUND                                                                                                        |
| 695   | ERROR_STATE_MACHINES_NOT_STARTED                                                                                           |
| 696   | ERROR_STATE_MACHINES_ALREADY_STARTED                                                                                       |
| 697   | ERROR_PARTIAL_RESPONSE_LOOPING                                                                                             |
| 698   | A response keyname in the device <b>.INF</b> file is not in the expected format.                                           |
| 699   | The device response caused buffer overflow.                                                                                |
| 700   | The expanded command in the device <b>.INF</b> file is too long.                                                           |
| 701   | The device moved to a BPS rate not supported by the COM driver.                                                            |
| 702   | Device response received when none expected.                                                                               |
| 703   | The Application does not allow user interaction the connection requires interaction with the user to complete successfully |
| 704   | ERROR_BAD_CALLBACK_NUMBER                                                                                                  |
| 705   | ERROR_INVALID_AUTH_STATE                                                                                                   |
| 706   | ERROR_WRITING_INITBPS                                                                                                      |
| 707   | X.25 diagnostic indication.                                                                                                |
| 708   | The account has expired.                                                                                                   |
| 709   | Error changing password on domain The password may be too short or may match a previously used password.                   |
| 710   | Serial overrun errors were detected while communicating with your modem.                                                   |
| 711   | RasMan initialization failure Check the event log.                                                                         |
| 712   | Biplex port initializing Wait a few seconds and redial.                                                                    |
| 713   | No active ISDN lines are available.                                                                                        |
| 714   | No ISDN channels are available to make the call.                                                                           |
| 715   | Too many errors occurred because of poor phone line quality.                                                               |
| 716   | The Remote Access IP configuration is unusable.                                                                            |
| 717   | No IP addresses are available in the static pool of Remote Access IP addresses.                                            |
| 718   | Timed out waiting for a valid response from the remote PPP peer.                                                           |
| 719   | PPP terminated by remote machine.                                                                                          |
| 720   | No PPP control protocols configured.                                                                                       |
| 721   | Remote PPP peer is not responding.                                                                                         |
| 722   | The PPP packet is invalid.                                                                                                 |
| 723   | The phone number including prefix and suffix is too long.                                                                  |
| 724   | The IPX protocol cannot dial-out on the port because the machine is an IPX router.                                         |
| 725   | The IPX protocol cannot dial-in on the port because the IPX router is not installed                                        |
| 726   | The IPX protocol cannot be used for dial-out on more than one port at a time.                                              |
| 727   | Cannot access <b>TCPCFG.DLL</b> .                                                                                          |
| 728   | Cannot find an IP adapter bound to Remote Access.                                                                          |
| 729   | SLIP cannot be used unless the IP protocol is installed.                                                                   |
| 730   | Computer registration is not complete.                                                                                     |

| Value | Description                                                                                                                                      |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| 731   | The protocol is not configured.                                                                                                                  |
| 732   | The PPP negotiation is not converging.                                                                                                           |
| 733   | The PPP control protocol for this network protocol is not available on the server.                                                               |
| 734   | The PPP link control protocol terminated.                                                                                                        |
| 735   | The requested address was rejected by the server.                                                                                                |
| 736   | The remote computer terminated the control protocol.                                                                                             |
| 737   | Loopback detected.                                                                                                                               |
| 738   | The server did not assign an address.                                                                                                            |
| 739   | The authentication protocol required by the remote server cannot use the Windows NT encrypted password Redial, entering the password explicitly. |
| 740   | Invalid TAPI configuration.                                                                                                                      |
| 741   | The local computer does not support the required encryption type.                                                                                |
| 742   | The remote computer does not support the required encryption type.                                                                               |
| 743   | The remote computer requires encryption.                                                                                                         |
| 744   | Cannot use the IPX network number assigned by remote server Check the event log.                                                                 |
| 745   | ERROR_INVALID_SMM                                                                                                                                |
| 746   | ERROR_SMM_UNINITIALIZED                                                                                                                          |
| 747   | ERROR_NO_MAC_FOR_PORT                                                                                                                            |
| 748   | ERROR_SMM_TIMEOUT                                                                                                                                |
| 749   | ERROR_BAD_PHONE_NUMBER                                                                                                                           |
| 750   | ERROR_WRONG_MODULE                                                                                                                               |
| 751   | Invalid callback number Only the characters 0 to 9, T, P, W, (, ), -, @, and space are allowed in the number.                                    |
| 752   | A syntax error was encountered while processing a script.                                                                                        |
| 753   | The connection could not be disconnected because it was created by the Multi-Protocol Router.                                                    |

## Notes

This function is not supported on Windows 7.

## Examples

| Tag Name | Expression                                                                           |
|----------|--------------------------------------------------------------------------------------|
| Tag      | <code>DialError( 0, "Office DialUp" )</code>                                         |
| Tag      | <code>DialError( 1, "USRobotics_SportsterFaxModem", "StatusMessage", second )</code> |
| Tag      | <code>DialError( 2, "DirectDial", "DialupError" )</code>                             |

See also: [FindModem\(\)](#)

## DialGetClientIP

This function gets the IP address of a Remote Access Service (RAS) client station.

| Function        | Group   | Execution   | Windows              | Embedded  | Thin Client |
|-----------------|---------|-------------|----------------------|-----------|-------------|
| DialGetClientIP | Dial-up | Synchronous | Supported (see note) | Supported | Supported   |

## 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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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:

| Value    | Description                                  |
|----------|----------------------------------------------|
| -5       | DialGetClientIP was not found in IndRAS.dll. |
| -4       | Invalid type; check numType.                 |
| -3       | Incorrect number of parameters.              |
| -2       | DialStatus was not found in IndRAS.dll.      |
| -1       | IndRAS.dll was not loaded.                   |
| <i>n</i> | Status code returned by DialStatus.          |

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.

| Group   | Execution   | Windows               | Embedded      | Thin Client |
|---------|-------------|-----------------------|---------------|-------------|
| Dial-up | Synchronous | Supported (see Notes) | Not Supported | Supported   |

**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.

 **Note:** The tag name must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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:

| Value    | Description                                  |
|----------|----------------------------------------------|
| -5       | DialGetServerIP was not found in IndRAS.dll. |
| -4       | Invalid type; check numType.                 |
| -3       | Incorrect number of parameters.              |
| -2       | DialStatus was not found in IndRAS.dll.      |
| -1       | IndRAS.dll was not loaded.                   |
| <i>n</i> | Status code returned by DialStatus.          |

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 "USRobotics\_SportsterFaxModem," write the resulting IP address to **ServerIPAddress**, and refresh every second (that is, every time the system tag **Second** changes):

```
DialGetServerIP(1, "USRobotics_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.

| Function   | Group   | Execution   | Windows              | Embedded  | Thin Client |
|------------|---------|-------------|----------------------|-----------|-------------|
| DialStatus | Dial-up | Synchronous | Supported (see note) | Supported | Supported   |

 **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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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

| Value | Description                                                  |
|-------|--------------------------------------------------------------|
| -5    | PhoneBook or modem does not exist                            |
| -4    | Invalid value for the numType parameter (0 or 1)             |
| -3    | Error: invalid number of parameters (minimum = 2)            |
| -2    | Error: <b>INDRAS . DLL</b> damaged                           |
| -1    | Error: <b>INDRAS . DLL</b> not found                         |
| 0     | Opening the port...                                          |
| 1     | Port was opened successfully.                                |
| 2     | Connecting to the device...                                  |
| 3     | The device has connected successfully.                       |
| 4     | All devices in the device chain have successfully connected. |
| 5     | Verifying the user name and password...                      |
| 6     | An authentication event has occurred.                        |
| 7     | Requested another validation attempt with a new user.        |
| 8     | Server has requested a callback number.                      |
| 9     | The client has requested to change the password              |

| Value | Description                                                        |
|-------|--------------------------------------------------------------------|
| 10    | Registering your computer on the network...                        |
| 11    | The link-speed calculation phase is starting...                    |
| 12    | An authentication request is being acknowledged.                   |
| 13    | Reauthentication (after callback) is starting.                     |
| 14    | The client has successfully completed authentication.              |
| 15    | The line is about to disconnect for callback.                      |
| 16    | Delaying to give the modem time to reset for callback.             |
| 17    | Waiting for an incoming call from server.                          |
| 18    | Projection result information is available.                        |
| 19    | User authentication is being initiated or retried.                 |
| 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 <b>RASPHONE . EXE</b> .                |
| 25    | Retry authentication state supported by <b>RASPHONE . EXE</b> .    |
| 26    | Callback state supported by <b>RASPHONE . EXE</b> .                |
| 27    | Change password state supported by <b>RASPHONE . EXE</b> .         |
| 8192  | Connected to remote server successfully!                           |
| 8193  | Disconnected.                                                      |

## Examples

| Tag Name | Expression                                                                            |
|----------|---------------------------------------------------------------------------------------|
| Tag      | <code>DialStatus( 0, "Office DialUp" )</code>                                         |
| Tag      | <code>DialStatus( 1, "USRobotics_SportsterFaxModem", "StatusMessage", second )</code> |
| Tag      | <code>DialStatus( 2, "DirectDial", "DialupError" )</code>                             |

See also: [FindModem\(\)](#)

## DialUp

Establishes a dial-up connection.

| Function | Group   | Execution    | Windows              | Embedded  | Thin Client |
|----------|---------|--------------|----------------------|-----------|-------------|
| DialUp   | Dial-up | Asynchronous | Supported (see note) | Supported | Supported   |

 **Note:** This function is not supported on Windows 7.

## Syntax

```
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

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

|    |                                                                                                                                                                                                                                 |
|----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0  | OK: dialing started                                                                                                                                                                                                             |
| -1 | Error: <b>INDRAS.DLL</b> not found                                                                                                                                                                                              |
| -2 | Error: <b>INDRAS.DLL</b> damaged                                                                                                                                                                                                |
| -3 | Error: invalid number of parameters (minimum=5)                                                                                                                                                                                 |
| -4 | Invalid value for the numType parameter (0 or 1)                                                                                                                                                                                |
| -5 | Invalid value for the strPhonebookEntryOrModem parameter (string)                                                                                                                                                               |
| -6 | PhoneBook or Modem does not exist                                                                                                                                                                                               |
| -7 | PhoneBook or Modem is in use                                                                                                                                                                                                    |
| -8 | Depends of the numType parameter: <ul style="list-style-type: none"> <li>• If numType = 0: Could not read properties from PhoneBook.</li> <li>• If numType = 1: More than 1000 connections are enabled at same time.</li> </ul> |
| -9 | Unable to create a temporary PhoneBook.                                                                                                                                                                                         |

### Notes

The operating system's RAS Server executes the dial-in for Windows PC stations automatically.

### Examples

| Tag Name | Expression                                                                      |
|----------|---------------------------------------------------------------------------------|
| Tag      | DialUp( 0, "OfficeDialup", "Guest", "Password" )                                |
| 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.

| Function   | Group   | Execution    | Windows              | Embedded      | Thin Client |
|------------|---------|--------------|----------------------|---------------|-------------|
| DialUpToCE | Dial-up | Asynchronous | Supported (see note) | Not supported | Supported   |

 **Note:** This function is not supported on Windows 7.

## Syntax

```
DialUpToCE(numModem, strDialPhone, strMyNumber, strUser, strPassword, optStrDomain, optAutoDial, optAutoClose)
```

### 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

|   |                                          |
|---|------------------------------------------|
| 0 | Fail, unable to call <b>DialUpToCE</b> . |
| 1 | Success, <b>DialUpToCE</b> executed.     |

## 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

| Tag Name | Expression                                                                           |
|----------|--------------------------------------------------------------------------------------|
| Tag      | <b>DialUpToCE</b> ( 0, "12344321", "98765432", "Administrator", "MyPass" )           |
| Tag      | <b>DialUpToCE</b> ( 0, "12344321", "98765432", "Administrator", "MyPass", "", 1, 1 ) |

## FindAllDevices

Returns the list of all the available modems and direct connection interfaces (COM ports) in the local station.

| Function       | Group   | Execution   | Windows              | Embedded  | Thin Client |
|----------------|---------|-------------|----------------------|-----------|-------------|
| FindAllDevices | Dial-up | Synchronous | Supported (see note) | Supported | Supported   |

 **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

| Tag Name | Expression                               |
|----------|------------------------------------------|
| Tag      | FindAllDevices( "SerialConnections[1]" ) |

## FindModem

Returns the list of all available modems in the local station.

| Function  | Group   | Execution   | Windows              | Embedded  | Thin Client |
|-----------|---------|-------------|----------------------|-----------|-------------|
| FindModem | Dial-up | Synchronous | Supported (see note) | Supported | Supported   |

 **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

| Tag Name | Expression               |
|----------|--------------------------|
| Tag      | FindModem( "Modems[1]" ) |

## HangUp

Hangs-up a dial-up connection.

| Function | Group   | Execution   | Windows              | Embedded  | Thin Client |
|----------|---------|-------------|----------------------|-----------|-------------|
| HangUp   | Dial-up | Synchronous | Supported (see note) | Supported | Supported   |

 **Note:** This function is not supported on Windows 7.

## Syntax

`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

|    |                                                         |
|----|---------------------------------------------------------|
| 0  | OK.                                                     |
| -1 | Error: <b>INDRAS.DLL</b> not found                      |
| -2 | Error: <b>INDRAS.DLL</b> damaged                        |
| -3 | Invalid value for the <b>numType</b> parameter (0 or 1) |
| -4 | PhoneBook or modem does not exist                       |
| -5 | No configured modems exist                              |

## Examples

| Tag Name | Expression                                                      |
|----------|-----------------------------------------------------------------|
| Tag      | <code>HangUp( 0, "OfficeDialup" )</code>                        |
| Tag      | <code>HangUp( 1, "USRobotics_SportsterFaxModem" )</code>        |
| Tag      | <code>HangUp( 2, "DirectDial", "Rberton", "MyPassword" )</code> |

See also: [FindModem\(\)](#)

## PhoneDialUp

Dials to a phone number using Telephony Application Program Interface (TAPI).

| Function    | Group   | Execution    | Windows              | Embedded      | Thin Client |
|-------------|---------|--------------|----------------------|---------------|-------------|
| PhoneDialUp | Dial-up | Asynchronous | Supported (see note) | Not supported | Supported   |

 **Note:** This function is not supported on Windows 7.

## Syntax

`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

|    |                                                                          |
|----|--------------------------------------------------------------------------|
| 0  | OK (dial triggered)                                                      |
| -1 | Invalid number of parameters                                             |
| -3 | INDTAPI.DLL library not found                                            |
| -4 | PhoneDialUp () function not supported by the current INDTAPI.DLL library |

### Examples

| Tag Name | Expression                                                            |
|----------|-----------------------------------------------------------------------|
|          | <code>PhoneDialUp ( "512-123-4567" )</code>                           |
|          | <code>PhoneDialUp ( StringPhoneNumberTag )</code>                     |
|          | <code>PhoneDialUp ( StringPhoneNumberTag, StringModemNameTag )</code> |

## PhoneDisableListen

Stops listening to the modem for incoming calls.

| Function           | Group  | Execution   | Windows              | Embedded      | Thin Client |
|--------------------|--------|-------------|----------------------|---------------|-------------|
| PhoneDisableListen | Dialup | Synchronous | Supported (see note) | Not supported | Supported   |

 **Note:** This function is not supported on Windows 7.

### Syntax

`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

|    |                                                                                 |
|----|---------------------------------------------------------------------------------|
| 1  | OK (stop listening for incoming calls)                                          |
| -1 | INDTAPI.DLL library not found                                                   |
| -2 | PhoneDisableListen () function not supported by the current INDTAPI.DLL library |

### Examples

| Tag Name | Expression                                                          |
|----------|---------------------------------------------------------------------|
|          | <code>PhoneDisableListen()</code>                                   |
|          | <code>PhoneDisableListen( "Hayes Compatible Modem on COM1" )</code> |
|          | <code>PhoneDisableListen( StringModemNameTag )</code>               |

See also: [PhoneEnableListen\(\)](#)

## PhoneEnableListen

Resumes listening to the modem for incoming calls.

| Function          | Group  | Execution   | Windows              | Embedded      | Thin Client |
|-------------------|--------|-------------|----------------------|---------------|-------------|
| PhoneEnableListen | Dialup | Synchronous | Supported (see note) | Not supported | Supported   |

 **Note:** This function is not supported on Windows 7.

### 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

|    |                                                                                |
|----|--------------------------------------------------------------------------------|
| 1  | OK (listening for incoming calls)                                              |
| 0  | Error executing the PhoneEnableListen() function                               |
| -1 | INDTAPI.DLL library not found                                                  |
| -2 | PhoneEnableListen () function not supported by the current INDTAPI.DLL library |

### Examples

| Tag Name | Expression                                                         |
|----------|--------------------------------------------------------------------|
|          | <code>PhoneEnableListen()</code>                                   |
|          | <code>PhoneEnableListen( "Hayes Compatible Modem on COM1" )</code> |
|          | <code>PhoneEnableListen( StringModemNameTag )</code>               |

See also: [PhoneDisableListen\(\)](#)

### PhoneHangUp

Hangs up a dial-up connection previously established with the PhoneDialUp function.

| Function    | Group   | Execution    | Windows              | Embedded      | Thin Client |
|-------------|---------|--------------|----------------------|---------------|-------------|
| PhoneHangUp | Dial-up | Asynchronous | Supported (see note) | Not supported | Supported   |

 **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

|    |                                                                          |
|----|--------------------------------------------------------------------------|
| 1  | OK (dial connection was dropped)                                         |
| -1 | INDTAPI.DLL library not found                                            |
| -2 | PhoneHangUp () function not supported by the current INDTAPI.DLL library |

### Examples

| Tag Name | Expression                                                   |
|----------|--------------------------------------------------------------|
|          | <code>PhoneHangUp()</code>                                   |
|          | <code>PhoneHangUp( "Hayes Compatible Modem on COM1" )</code> |
|          | <code>PhoneHangUp( StringModemNameTag )</code>               |

## PhoneStatus

| Function    | Group   | Execution   | Windows              | Embedded      | Thin Client |
|-------------|---------|-------------|----------------------|---------------|-------------|
| PhoneStatus | Dial-up | Synchronous | Supported (see note) | Not supported | Supported   |

 **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

|   |                        |
|---|------------------------|
| 0 | "Ready to make a call" |
| 1 | "Call was shut down"   |
| 2 | "Line Ringing"         |
| 3 | "Dial Tone"            |
| 4 | "Dialing Call"         |
| 5 | "Call is Proceeding"   |
| 6 | "Ring Back"            |
| 7 | "Line is Busy"         |
| 8 | "Line is Idle"         |
| 9 | "Disconnected"         |

### Examples

| Tag Name        | Expression                                                                        |
|-----------------|-----------------------------------------------------------------------------------|
| Tag Status Code | <code>PhoneStatus( "String Tag Status" )</code>                                   |
| Tag Status Code | <code>PhoneStatus( "String Tag Status", "Hayes Compatible Modem on COM1" )</code> |
| Tag Status Code | <code>PhoneStatus( "String Tag Status", "StringModemNameTag" )</code>             |

## 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

This function gets the current value of a Property on an ActiveX Control or .NET Control object.

| Function | Group                    | Execution    | Windows   | Embedded                          | Thin Client |
|----------|--------------------------|--------------|-----------|-----------------------------------|-------------|
| XGet     | ActiveX and .NET Control | Asynchronous | Supported | ActiveX Controls only (see Notes) | Supported   |

### Syntax

XGet ( *strName* , *strProperties* )

#### **strName**

The unique name of the ActiveX Control or .NET Control object, as configured in the **Name** field of the *Object Properties* dialog.

#### **strProperties**

The Property that you want to get the value of. Available Properties are listed in the *Configuration* (for an ActiveX Control) or *Members* (for a .NET Control) dialog.

### Returned value

Returns the value of the specified Property.

### Notes

This function cannot be used in [Tasks](#) or in the [Global Procedures](#) script. Also, this function is not supported for .NET Control objects running on a Windows Embedded station.

### Examples

Get the current value of the Color property on the ActiveX Control object named "ActXRec":

```
XGet("ActXRec", "Color")
```

### XRun

This function runs a Method on an ActiveX Control or .NET Control object.

| Function | Group                    | Execution    | Windows   | Embedded                          | Thin Client |
|----------|--------------------------|--------------|-----------|-----------------------------------|-------------|
| XRun     | ActiveX and .NET Control | Asynchronous | Supported | ActiveX Controls only (see Notes) | Supported   |

### Syntax

XRun ( *strName* , *strMethod* , *Parameter1* , *ParameterN* )

#### **strName**

The unique name of the ActiveX Control or .NET Control object, as configured in the **Name** field of the *Object Properties* dialog.

#### **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.

#### **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

Returns the Method result as reported by the ActiveX Control or .NET Control object. Not all Methods return results.

## Notes

This function cannot be used in [Tasks](#) or in the [Global Procedures](#) script. Also, this function is not supported for .NET Control objects running on a Windows Embedded station.

## 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

This function sets the value of a Property on an ActiveX Control or .NET Control object.

| Function | Group                    | Execution    | Windows   | Embedded                          | Thin Client |
|----------|--------------------------|--------------|-----------|-----------------------------------|-------------|
| XSet     | ActiveX and .NET Control | Asynchronous | Supported | ActiveX Controls only (see Notes) | Supported   |

## Syntax

```
XSet (strName, strProperties, Value)
```

### strName

The unique name of the ActiveX Control or .NET Control object, as configured in the **Name** field of the *Object Properties* dialog.

### 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.

### 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

This function cannot be used in [Tasks](#) or in the [Global Procedures](#) script. Also, this function is not supported for .NET Control objects running on a Windows Embedded station.

## 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.

| Function  | Group        | Execution   | Windows   | Embedded  | Thin Client |
|-----------|--------------|-------------|-----------|-----------|-------------|
| SendEvent | Event Logger | Synchronous | Supported | Supported | Supported   |

### 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)
```

#### 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.

### Returned value

|   |                                                                                               |
|---|-----------------------------------------------------------------------------------------------|
| 0 | Success                                                                                       |
| 1 | Event Logger is disabled in the <i>Event Settings</i> dialog                                  |
| 2 | Event Logger is enabled, but Custom Messages are disabled in the <i>Event Settings</i> dialog |

### 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

| Tag Name | Expression                                                                                                                            |
|----------|---------------------------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>SendEvent( "Valve Open" )</code> // Saves the event message.                                                                    |
| Tag      | <code>SendEvent( "Valve Open Oven No. " + OvenID )</code> // Saves the event message concatenated with the value of the OvenID tag.   |
| Tag      | <code>SendEvent( "Valve Open" , 1 )</code> // Displays the dialog where the operator can type comments.                               |
| Tag      | <code>SendEvent( "Valve Open" , 1 , TagComment )</code> // Saves the event message with the comment configured in the TagComment tag. |

## 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.

| Function | Group | Execution   | Windows   | Embedded  | Thin Client |
|----------|-------|-------------|-----------|-----------|-------------|
| CnfFTP   | FTP   | Synchronous | Supported | Supported | Supported   |

### 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.

#### **optStrPwn**

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.)

|   |                               |
|---|-------------------------------|
| 0 | Passive FTP mode is disabled. |
| 1 | Passive FTP mode is enabled.  |

This parameter is optional; if no value is given, then Passive FTP mode is disabled by default.

#### **optNumPort**

The TCP/IP port number.

This parameter is optional; if no value is given, then port 21 is used by default.

### Returned value

|    |                              |
|----|------------------------------|
| 0  | Success                      |
| -1 | Invalid number of parameters |
| -2 | Invalid server name          |
| -3 | Invalid user name            |



**Note:** 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.

### Notes

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

| Tag Name | Expression                                                                                                       |
|----------|------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>CnfFTP( "ftp.mycompany.com", "admin", "12345", 1 )</code> // Configures the FTP server using passive mode. |

## FTPGet

Gets a file from a remote server and saves it on the local machine.

| Function | Group | Execution    | Windows   | Embedded  | Thin Client |
|----------|-------|--------------|-----------|-----------|-------------|
| FTPGet   | FTP   | Asynchronous | Supported | Supported | Supported   |

## 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.

|    |                          |
|----|--------------------------|
| 0  | Unknown                  |
| 1  | ASCII                    |
| 2  | Binary                   |
| 10 | Unknown, without caching |
| 11 | ASCII, without caching   |
| 12 | Binary, without caching  |

 **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.

|   |                                                                |
|---|----------------------------------------------------------------|
| 0 | Do not overwrite — return an error if the file already exists. |
| 1 | Overwrite.                                                     |

## Returned value

|    |                               |
|----|-------------------------------|
| 1  | Failed to open FTP connection |
| 0  | Success                       |
| -1 | Invalid number of parameters  |
| -2 | Unknown system error          |
| -3 | Invalid remote file           |

|    |                       |
|----|-----------------------|
| -4 | Invalid local file    |
| -5 | Invalid transfer type |

## 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

| Tag Name | Expression                                                                                                                                                                                                                                                                                        |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>FTPGet( "\Reports\040303.txt", "C:\Report.txt" )</code> // Retrieves the file <code>040303.txt</code> in the <code>Reports</code> directory, from the FTP server that was previously specified by <a href="#">CnfFTP()</a> , It then saves the file locally at <code>C:\Report.txt</code> . |

## FTPPut

Puts a file on a remote server.

| Function | Group | Execution    | Windows   | Embedded  | Thin Client |
|----------|-------|--------------|-----------|-----------|-------------|
| FTPPut   | FTP   | Asynchronous | Supported | Supported | Supported   |

## 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

|   |         |
|---|---------|
| 0 | Unknown |
| 1 | ASCII   |
| 2 | Binary  |

Default is 0.

## Returned value

|    |                              |
|----|------------------------------|
| 1  | Failed to create FTP thread  |
| 0  | Success                      |
| -1 | Invalid number of parameters |
| -2 | Unknown system error         |
| -3 | Invalid remote file          |
| -4 | Invalid local file           |
| -5 | Invalid transfer type        |

## 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

| Tag Name | Expression                                                                                                                                                                                               |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>FTPput("C:\Report.txt", "\Reports\040303.txt")</code> // Retrieves the file <code>C:\Report.txt</code> with the name <code>040303.txt</code> in the folder <code>Reports</code> in the FTP Server. |

## FTPStatus

This function returns the current status of a file transfer started with `FTPGet` or `FTPput`.

| Function  | Group | Execution   | Windows   | Embedded  | Thin Client |
|-----------|-------|-------------|-----------|-----------|-------------|
| FTPStatus | FTP   | Synchronous | Supported | Supported | Supported   |

## Syntax

`FTPStatus("strStatusTag")`

### strStatusTag

Name of the string tag that will receive the current status description when the function returns.

 **Note:** The tag name must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named tag.

## Returned value

|    |                                                               |
|----|---------------------------------------------------------------|
| 1  | Transaction executed successfully                             |
| 2  | Resolving name                                                |
| 3  | Name resolved                                                 |
| 4  | Connecting to server                                          |
| 5  | Connected to server                                           |
| 6  | Closing connection                                            |
| 7  | Connection closed                                             |
| 8  | Sending request                                               |
| 9  | Request sent                                                  |
| 10 | Receiving response                                            |
| 11 | Intermediate response received                                |
| 12 | Response received                                             |
| 13 | Request completed                                             |
| 0  | No transaction is being executed                              |
| -2 | Invalid <code>opttagErrorDescription</code>                   |
| -6 | Error opening connection (see status string for details)      |
| -7 | Error establishing connection (see status string for details) |
| -8 | Error receiving the file (see status string for details)      |
| -9 | Transfer pending                                              |

## Example

| Tag Name | Expression                                                                                                                                                                                                             |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>FTPstatus("StatusDescription")</code> // Retrieves the status of a current transfer. The return code is stored in the <code>StatusCode</code> tag and the description in the <code>StatusDescription</code> tag. |

## 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.

| Function      | Group        | Execution   | Windows   | Embedded  | Thin Client |
|---------------|--------------|-------------|-----------|-----------|-------------|
| DBCursorClose | Database/ERP | Synchronous | Supported | Supported | Supported   |

 **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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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](#).

### Examples

As used in a Math worksheet:

| Tag Name   | Expression               |
|------------|--------------------------|
| nErrorCode | DBCursorClose( nCursor ) |

### **DBCursorColumnCount**

Gets the total number of columns in a SQL result set.

| Function            | Group        | Execution   | Windows   | Embedded  | Thin Client |
|---------------------|--------------|-------------|-----------|-----------|-------------|
| DBCursorColumnCount | Database/ERP | Synchronous | Supported | Supported | Supported   |

 **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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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](#).

**Examples**

As used in a Math worksheet:

| Tag Name   | Expression                     |
|------------|--------------------------------|
| nErrorCode | DBCursorColumnCount( nCursor ) |

**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.

| Function       | Group        | Execution   | Windows   | Embedded  | Thin Client |
|----------------|--------------|-------------|-----------|-----------|-------------|
| DBCursorColumn | Database/ERP | Synchronous | Supported | Supported | Supported   |

 **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:

| Value | Description      |
|-------|------------------|
| 0     | Column name      |
| 1     | Column data type |

**optStrErrorTag**

The name of a String tag that will receive detailed error messages, if errors occur during runtime.

 **Note:** The tag name must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named tag.

This is an optional parameter.

### Returned value

In case of error, returns a negative number. Otherwise, returns 0.

### Examples

As used in a Math worksheet:

| Tag Name   | Expression                                                                                          |
|------------|-----------------------------------------------------------------------------------------------------|
| nErrorCode | DBCursorColumnInfo( nCursor, 2, 0 ) // 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.

| Function           | Group        | Execution   | Windows   | Embedded  | Thin Client |
|--------------------|--------------|-------------|-----------|-----------|-------------|
| DBCursorCurrentRow | Database/ERP | Synchronous | Supported | Supported | Supported   |

 **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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named tag.

This is an optional parameter.

### Returned value

Returns the number of the current row. In case of error, returns a negative number.

### Examples

As used in a Math worksheet:

| Tag Name | Expression                    |
|----------|-------------------------------|
| nRow     | DBCursorCurrentRow( nCursor ) |

### DBCursorGetValue

Gets the value of the specified column of the current row (i.e., the cursor position) in a SQL result set.

| Function         | Group        | Execution   | Windows   | Embedded  | Thin Client |
|------------------|--------------|-------------|-----------|-----------|-------------|
| DBCursorGetValue | Database/ERP | Synchronous | Supported | Supported | Supported   |

 **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

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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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.

## Examples

As used in a Math worksheet:

| Tag Name | Expression                                          |
|----------|-----------------------------------------------------|
| Tag      | <code>DBCursorGetValue( nCursor, "Column1" )</code> |

## 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.

| Function       | Group        | Execution   | Windows   | Embedded  | Thin Client |
|----------------|--------------|-------------|-----------|-----------|-------------|
| DBCursorMoveTo | Database/ERP | Synchronous | Supported | Supported | Supported   |

 **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, numRows, "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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named tag.

This is an optional parameter.

## Returned value

In case of error, returns a negative number. Otherwise, returns 0.

## Examples

As used in a Math worksheet:

| Tag Name   | Expression                                                                                                    |
|------------|---------------------------------------------------------------------------------------------------------------|
| nErrorCode | DBCursorMoveTo( nCursor, 4 ) // Moves the cursor to the fourth row of the result set and copies those values. |

## 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.

| Function      | Group        | Execution   | Windows   | Embedded  | Thin Client |
|---------------|--------------|-------------|-----------|-----------|-------------|
| DBCursorClose | Database/ERP | Synchronous | Supported | Supported | Supported   |

| Group  | Execution   | Windows PC | Windows CE | Thin Client |
|--------|-------------|------------|------------|-------------|
| DB/ERP | Synchronous | Supported  | Supported  | Supported   |

 **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

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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named tag.

This is an optional parameter.

## Returned value

In case of error, returns a negative number. Otherwise, returns 0.

## Examples

As used in a Math worksheet:

| Tag Name   | Expression              |
|------------|-------------------------|
| nErrorCode | DBCursorNext( nCursor ) |

## 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.

| Function     | Group        | Execution   | Windows   | Embedded  | Thin Client |
|--------------|--------------|-------------|-----------|-----------|-------------|
| DBCursorOpen | Database/ERP | Synchronous | Supported | Supported | Supported   |

 **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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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](#).

## Examples

As used in a Math worksheet:

| Tag Name | Expression                                                                                                                                       |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| nCursor  | DBCursorOpen( "DB1", "Table1", "Column1 > 3", "Column1, Column2", "Tag1, Tag2", "Column1, Column2 DESC", "TagError" ) // Opens Table1 of DB1 and |

| Tag Name | Expression                                                                                                                                                                                                                            |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|          | 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. |

## 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.)

| Function        | Group        | Execution   | Windows   | Embedded  | Thin Client |
|-----------------|--------------|-------------|-----------|-----------|-------------|
| DBCursorOpenSQL | Database/ERP | Synchronous | Supported | Supported | Supported   |

 **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:

```
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:

```
$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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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](#).

### Examples

As used in a Math worksheet:

| Tag Name | Expression                                                                                                                    |
|----------|-------------------------------------------------------------------------------------------------------------------------------|
| nCursor  | DBCursorOpenSQL( "DB1", "SELECT Column1, Column2 FROM Table1 WHERE Column1 > 3 ORDER BY Column1, Column2 DESC", "Tag1, Tag2") |

### 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.

| Function         | Group        | Execution   | Windows   | Embedded  | Thin Client |
|------------------|--------------|-------------|-----------|-----------|-------------|
| DBCursorPrevious | Database/ERP | Synchronous | Supported | Supported | Supported   |

 **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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named tag.

This is an optional parameter.

### Returned value

In case of error, returns a negative number. Otherwise, returns 0.

### Examples

As used in a Math worksheet:

| Tag Name   | Expression                  |
|------------|-----------------------------|
| nErrorCode | DBCursorPrevious( nCursor ) |

### DBCursorRowCount

Gets the total number of rows in a SQL result set.

| Function         | Group        | Execution   | Windows   | Embedded  | Thin Client |
|------------------|--------------|-------------|-----------|-----------|-------------|
| DBCursorRowCount | Database/ERP | Synchronous | Supported | Supported | Supported   |

 **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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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](#).

## Examples

As used in a Math worksheet:

| Tag Name  | Expression                  |
|-----------|-----------------------------|
| nRowCount | DBCursorRowCount( nCursor ) |

## DBDelete

Deletes selected rows from a database table. (This function is equivalent to a SQL DELETE statement.)

| Function | Group        | Execution   | Windows   | Embedded  | Thin Client |
|----------|--------------|-------------|-----------|-----------|-------------|
| DBDelete | Database/ERP | Synchronous | Supported | Supported | Supported   |

 **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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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:

| Tag Name     | Expression                                                                                                                                                                                                             |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| nRowsDeleted | DBDelete( "DB1", "Table1", "Column1 > 1000", "TagError" ) // 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. |
| Tag          | DBDelete( "DB1", "Table1", " " ) // Deletes all rows of Table1.                                                                                                                                                        |

**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.

| Function  | Group        | Execution   | Windows   | Embedded  | Thin Client |
|-----------|--------------|-------------|-----------|-----------|-------------|
| DBExecute | Database/ERP | Synchronous | Supported | Supported | Supported   |

**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:

```
AuxTag = "{Microsoft Text Driver (*.txt; *.csv)}"
```

```
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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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

```
DBExecute("DB1", "INSERT INTO Table1(Column1,Column2) values(1,1)")
```

```
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.)

| Function | Group        | Execution   | Windows   | Embedded  | Thin Client |
|----------|--------------|-------------|-----------|-----------|-------------|
| DBInsert | Database/ERP | Synchronous | Supported | Supported | Supported   |

 **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

```
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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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:

| Tag Name   | Expression                                                  |
|------------|-------------------------------------------------------------|
| nErrorCode | DBInsert( "DB1", "Table1", "1, 'one'", "Column1, Column2" ) |

### 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.

| Function | Group        | Execution   | Windows   | Embedded  | Thin Client |
|----------|--------------|-------------|-----------|-----------|-------------|
| DBSelect | Database/ERP | Synchronous | Supported | Supported | Supported   |

### 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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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.)

| Function | Group        | Execution   | Windows   | Embedded  | Thin Client |
|----------|--------------|-------------|-----------|-----------|-------------|
| DBUpdate | Database/ERP | Synchronous | Supported | Supported | Supported   |

 **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 must be enclosed in quotes, as shown in the syntax diagram, or else the project will try to get the value of the named 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:

| Tag Name | Expression                                                                                                                                                |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Tag      | DBUpdate( "DB1", "Table1", "'X'", "Column2", "Column1 = 1", "TagError" ) // In Table1 of DB1, for all rows where Column1 equals 1, writes "X" to Column2. |

## SyncAlarm

Synchronizes the alarm database.

| Function  | Group        | Execution    | Windows   | Embedded  | Thin Client        |
|-----------|--------------|--------------|-----------|-----------|--------------------|
| SyncAlarm | Database/ERP | Asynchronous | Supported | Supported | Executed on Server |

## Syntax

```
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

| Value | Description                                                                 |
|-------|-----------------------------------------------------------------------------|
| 1     | Fail to start synchronization; the database is probably being synchronized. |
| 0     | Success                                                                     |
| -1    | Invalid group number                                                        |
| -2    | The format is not set to "Database".                                        |
| -4    | Start date specified is invalid.                                            |
| -5    | End date specified is invalid.                                              |
| -6    | Start date is greater than the end date.                                    |

### 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

| Tag Name | Expression                                                                                                       |
|----------|------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>SyncAlarm( )</code> // Synchronizes the database using the current date                                    |
| Tag      | <code>SyncAlarm( "10/20/2004" )</code> // Synchronizes the database only for the day 10/20/2004                  |
| Tag      | <code>SyncAlarm( "10/20/2004" , "10/28/2004" )</code> // Synchronizes the database from 10/20/2004 to 10/28/2004 |

### SyncAlarmStatus

Returns the status of a previously called `SyncAlarm` function.

| Function        | Group        | Execution   | Windows   | Embedded  | Thin Client        |
|-----------------|--------------|-------------|-----------|-----------|--------------------|
| SyncAlarmStatus | Database/ERP | Synchronous | Supported | Supported | Executed on Server |

### Syntax

`SyncAlarmStatus( )`

This function takes no parameters.

### Returned value

| Value | Description                           |
|-------|---------------------------------------|
| 3     | Synchronization has finished.         |
| 2     | Fail synchronizing                    |
| 1     | Still synchronizing                   |
| 0     | No synchronization is being executed. |
| -1    | The format is not set to "Database".  |

### Examples

| Tag Name | Expression                      |
|----------|---------------------------------|
| Tag      | <code>SyncAlarmStatus( )</code> |

**SyncEvent**

Synchronizes the event database.

| Function  | Group        | Execution    | Windows   | Embedded  | Thin Client        |
|-----------|--------------|--------------|-----------|-----------|--------------------|
| SyncEvent | Database/ERP | Asynchronous | Supported | Supported | Executed on Server |

**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**

| Value | Description                                                                 |
|-------|-----------------------------------------------------------------------------|
| 1     | Fail to start synchronization; the database is probably being synchronized. |
| 0     | Success                                                                     |
| -1    | Invalid group number                                                        |
| -2    | The format is not set to "Database".                                        |
| -4    | Start date specified is invalid.                                            |
| -5    | End date specified is invalid.                                              |
| -6    | Start date is greater than the end date.                                    |

**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**

| Tag Name | Expression                                                                                                        |
|----------|-------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>SyncEvent ( )</code> // Synchronizes the database using the current date                                    |
| Tag      | <code>SyncEvent ( "10/20/2004" )</code> // Synchronizes the database only for the day 10/20/2004                  |
| Tag      | <code>SyncEvent ( "10/20/2004" , "10/28/2004" )</code> // Synchronizes the database from 10/20/2004 to 10/28/2004 |

**SyncEventStatus**

Returns the status of a previously called SyncEvent function.

| Function        | Group        | Execution   | Windows   | Embedded  | Thin Client        |
|-----------------|--------------|-------------|-----------|-----------|--------------------|
| SyncEventStatus | Database/ERP | Synchronous | Supported | Supported | Executed on Server |

**Syntax**

`SyncEventStatus( )`

This function takes no parameters.

**Returned value**

| Value | Description                   |
|-------|-------------------------------|
| 3     | Synchronization has finished. |

| Value | Description                           |
|-------|---------------------------------------|
| 2     | Fail synchronizing                    |
| 1     | Still synchronizing                   |
| 0     | No synchronization is being executed. |
| -1    | The format is not set to "Database".  |

## Examples

| Tag Name | Expression          |
|----------|---------------------|
| Tag      | SyncEventStatus ( ) |

## SyncTrend

Synchronizes the trend database.

| Function  | Group        | Execution    | Windows   | Embedded  | Thin Client        |
|-----------|--------------|--------------|-----------|-----------|--------------------|
| SyncTrend | Database/ERP | Asynchronous | Supported | Supported | Executed on Server |

## 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

| Value | Description                                                                 |
|-------|-----------------------------------------------------------------------------|
| 1     | Fail to start synchronization; the database is probably being synchronized. |
| 0     | Success                                                                     |
| -1    | Invalid group number                                                        |
| -2    | The format is not set to "Database".                                        |
| -4    | Start date specified is invalid.                                            |
| -5    | End date specified is invalid.                                              |
| -6    | Start date is greater than the end date.                                    |

## 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

| Tag Name | Expression                                                                                                                 |
|----------|----------------------------------------------------------------------------------------------------------------------------|
| Tag      | <code>SyncTrend( 1 )</code> // Synchronizes the group 1 database using the current date                                    |
| Tag      | <code>SyncTrend( 1, "10/20/2004" )</code> // Synchronizes the group 1 database only for the day 10/20/2004                 |
| Tag      | <code>SyncTrend( 1, "10/20/2004", "10/28/2004" )</code> // Synchronizes the group 1 database from 10/20/2004 to 10/28/2004 |

## SyncTrendStatus

Returns the status of a previously called SyncTrend function.

| Function        | Group        | Execution   | Windows   | Embedded  | Thin Client        |
|-----------------|--------------|-------------|-----------|-----------|--------------------|
| SyncTrendStatus | Database/ERP | Synchronous | Supported | Supported | Executed on Server |

### Syntax

SyncTrendStatus(*numGroup*)

#### numGroup

Trend group/worksheet number.

### Returned value

| Value | Description                           |
|-------|---------------------------------------|
| 3     | Synchronization has finished.         |
| 2     | Fail synchronizing                    |
| 1     | Still synchronizing                   |
| 0     | No synchronization is being executed. |
| -1    | The format is not set to "Database".  |

### Examples

| Tag Name | Expression           |
|----------|----------------------|
| Tag      | SyncTrendStatus( 1 ) |

## Appendix: VBScript

---

## 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 Server station), Microsoft Windows Embedded devices (via 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 CEView will be able to execute the scripts configured in the VBScript language on the device.

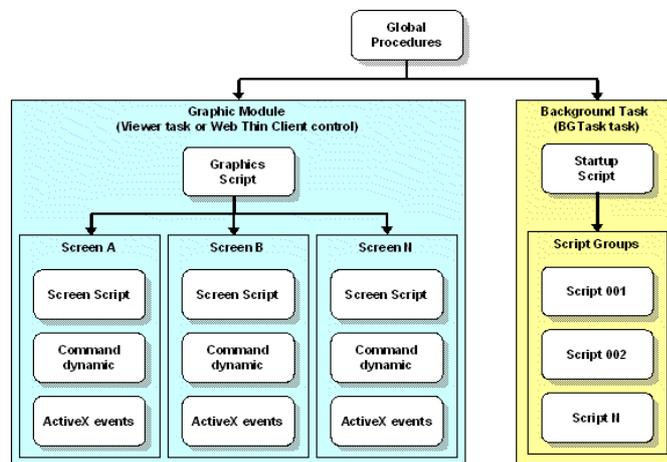
The aim of this documentation is to provide an overview about the integration of VBScript with InduSoft Web Studio. Furthermore, it can be used as a quick reference for the most used features of the language. For a full description of the language as well as its interfaces and functions, please consult 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:

| Interface         | Scope for Procedures and Variables    | Execution                      | Functionality                                                      |
|-------------------|---------------------------------------|--------------------------------|--------------------------------------------------------------------|
| Global Procedures | Graphics and Tasks                    | -                              | Declaration of Procedures                                          |
| Graphics Script   | Graphics Script interface only        | Server (Viewer) + Thin Clients | Declaration of Variables<br>Declaration of Procedures<br>Execution |
| Screen Script     | Screen where the script is configured | Server (Viewer) + Thin Clients | Declaration of Variables<br>Declaration of Procedures<br>Execution |
| Command animation | Object where the script is configured | Server (Viewer) + Thin Clients | Declaration of Variables<br>Execution                              |
| ActiveX Events    | Object where the script is configured | Server (Viewer) + Thin Clients | Declaration of Variables<br>Execution                              |
| Startup Script    | All Script Sheets from Tasks          | Server (BGTask)                | Declaration of Variables<br>Declaration of Procedures<br>Execution |
| Script Groups     | Script Group only                     | Server (BGTask)                | Declaration of Variables<br>Execution                              |

The following picture illustrates the scope of each VBScript interface and the order that 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

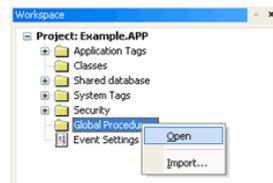
This Global Procedures interface is used create a library of [VBScript functions and sub-routines](#) that can be called by any other scripting interface in IWS. The procedures declared here are never directly executed during runtime; they must be explicitly called by another script.



**Caution:** IWS 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.

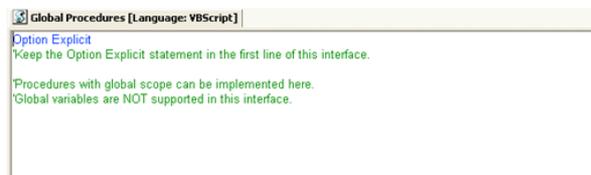
To use the Global Procedures interface:

1. In the [Global tab](#) of the Project Explorer, right-click the **Global Procedures** folder and choose **Open** from the shortcut menu.



Global Procedures - Open option

The *Global Procedures* interface is displayed.



Global Procedures Interface

2. Declare your functions and sub-routines in the interface. For example:

```
Option Explicit
'Keep the Option Explicit statement in the first line of this interface.

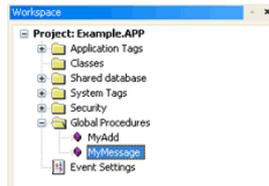
'Procedures with global scope can be implemented here
'Global variables are NOT supported in this interface

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 functions and sub-routines are added to the Global Procedures folder in the *Project Explorer*.



**Global Procedures Tree**

## Organizing Procedures into Subfolders

You can organize declared procedures into subfolders within the Global Procedures folder. To organize procedures:

1. In the Global Procedures interface, insert the following line before the procedures that you want to group together:

```
'$region:foldername
```

...where *foldername* is the name of the subfolder. For example:

```
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
```

```
'$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
```

2. Save your changes. The procedures are organized into subfolders in the *Project Explorer*.

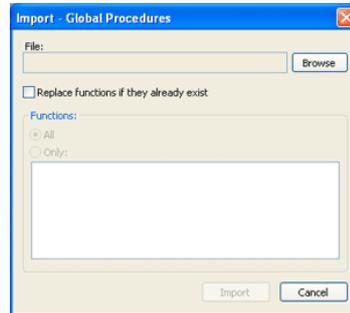
## Importing Functions from an External File

You can also import functions from an external file and add them to the Global Procedures folder. This is useful if you have a library of existing functions that you want to reuse.

To import functions:

1. Save and close all open screens and worksheets.
2. Right-click the *Global Procedures* folder and then choose **Import...** from the shortcut menu.

The *Import - Global Procedures* dialog is displayed.



*Import - Global Procedures dialog*

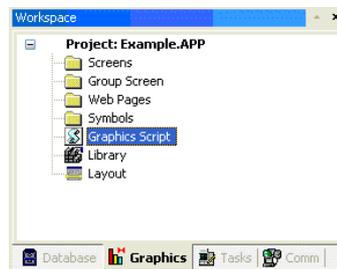
3. In the **File** field, click **Browse** to open a standard Windows file browser and select a global procedures file. (This is a plain text file that has been saved with the **.gis** file extension.)
4. Select **Replace functions if they already exist** to overwrite functions in the Global Procedures folder with functions imported from the file, if the functions have the same names.
5. In the *Functions* area, you can import **All** functions from the global procedures file (**\*.gis**) or **Only** selected functions.
6. Click **Import**.

After the functions are imported, they should be available in the *Global Procedures* folder.

## 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 in CEView on 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 (where InduSoft Web Studio or CEView is 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**.

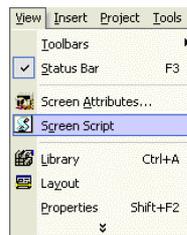
Taking the function **MyNewProcedure** that was declared in the example above, you could place a Button object on your project screen and then apply a Command animation to it with the following line:

```
$NewTag = Graphics.MyNewProcedure($OldTag)
```

## SCREEN SCRIPT

To edit the screen script for a project screen, do one of the following:

- On the Graphics tab of the ribbon, in the Screen group, click **Script**; or
- Right-click in the screen and then select **Screen Script** option from the shortcut menu.



This interface can be used to execute logics on the following events, based on preconfigured sub-routines:

- **Screen\_IsClosedByReplace()**: 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:

```
'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!")
End Sub
```

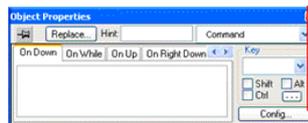
## COMMAND ANIMATION

To edit the Command animation interface, do the following:

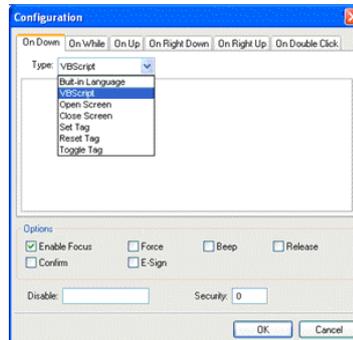
1. Select the object.
2. Click the **Command animations** icon in the **Animations** group.



3. Right-click on the object. The *Object Properties* dialog for the Command animation will open.



4. Click the **Config...** button.
5. Select **VBScript** as the **Type**.



Use this interface to execute logics when the user clicks on the object where the Command animation is applied (during runtime) or presses the shortcut (Key) associated with the Command animation.

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 Command animation is configured.

 **Note:** For more information, see [Command animation](#).

Example:

```
'The script below will be executed when the user clicks on the object
'where this animation is configured
$MyValue = InputBox("Please enter the new set-point", "Set-point")
```

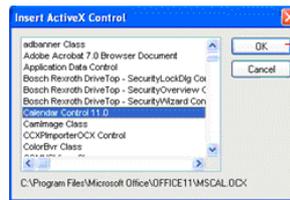
## ACTIVEX EVENTS

To edit the ActiveX Events interface, select the Script option from the Events tab of the ActiveX object inserted on the screen.

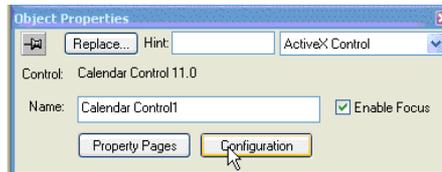
1. Click the ActiveX Control icon in the Active Objects toolbar.



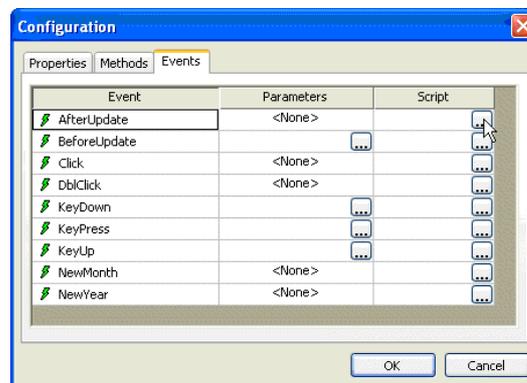
The *Insert ActiveX Control* dialog opens.



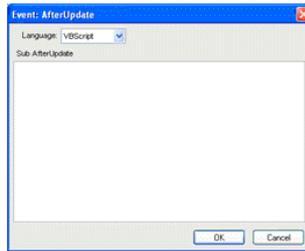
2. Select the ActiveX Control that you wish to use and then click **OK**.
3. The object that symbolizes the selected ActiveX Control will display. Right-click on this object to open the *Object Properties* dialog.



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:

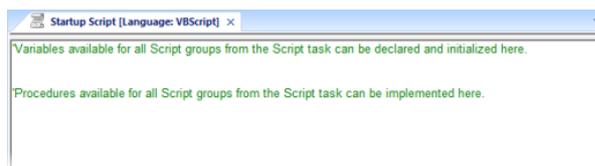
```
'The script below will be executed when the Calendar Control ActiveX
'triggers its "AfterUpdate" event
$MyYear = CalendarControll.Year
$MyMonth = CalendarControll.Month
$MyDay = CalendarControll.Day
```

## Background Task

### 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*

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:

```
'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

```

## SCRIPT WORKSHEET

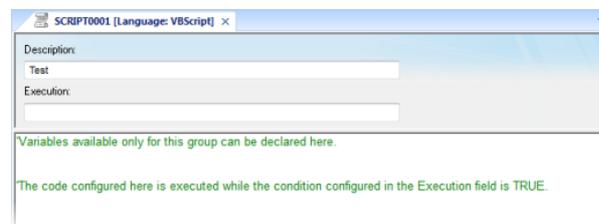
A Script worksheet is used to implement program logic (using VBScript) that should be continuously executed during runtime, rather than on specific actions like the user pressing a button on a screen.

**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.



*Script worksheet*

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:

```

'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.

## Language Reference

### Operators

#### Arithmetic Operators

| Symbol | Name                 | Description                                                                                       |
|--------|----------------------|---------------------------------------------------------------------------------------------------|
| ^      | Exponentiation       | Raises a number to the power of an exponent.                                                      |
| -      | Unary negation       | Finds the difference between two numbers or indicates the negative value of a numeric expression. |
| *      | Multiplication       | Multiplies two numbers.                                                                           |
| /      | Division             | Divides two numbers and returns a floating-point result.                                          |
| \      | Integer division     | Divides two numbers and returns an integer result.                                                |
| Mod    | Modulus arithmetic   | Divides two numbers and returns only the remainder.                                               |
| +      | Addition             | Finds the sum of two numbers.                                                                     |
| -      | Subtraction          | Finds the difference between two numbers or indicates the negative value of a numeric expression. |
| &      | String concatenation | Forces string concatenation of two expressions.                                                   |

#### Comparison Operators

| Symbol | Name                     | Description                                                                                                |
|--------|--------------------------|------------------------------------------------------------------------------------------------------------|
| =      | Equality                 | Comparison is True if the first expression is equal to the second expression.                              |
| <>     | Inequality               | Comparison is True if the first expression is different from the second expression.                        |
| <      | Less than                | Comparison is True if the first expression is less than the second expression.                             |
| >      | Greater than             | Comparison is True if the first expression is greater than the second expression.                          |
| <=     | Less than or equal to    | Comparison is True if the first expression is less than or equal to the second expression.                 |
| >=     | Greater than or equal to | Comparison is True if the first expression is greater than or equal to the second expression.              |
| Is     | Object equivalence       | Compares two object reference variables. Comparison is True if both object names refer to the same object. |

#### Logical Operators

| Symbol | Name                | Description                                        |
|--------|---------------------|----------------------------------------------------|
| Not    | Logical negation    | Performs logical negation on an expression.        |
| And    | Logical conjunction | Performs a logical conjunction on two expressions. |
| Or     | Logical disjunction | Performs a logical disjunction on two expressions. |
| Xor    | Logical exclusion   | Performs a logical exclusion on two expressions.   |
| Eqv    | Logical equivalence | Performs a logical equivalence on two expressions. |
| Imp    | Logical implication | Performs a logical implication on two expressions. |

#### Assignment Operators

| Symbol | Name       | Description                                |
|--------|------------|--------------------------------------------|
| =      | Assignment | Assigns a value to a variable or property. |

### Constants

#### Color Constants

| Constant | Value | Description |
|----------|-------|-------------|
| vbBlack  | &h00  | Black       |

| Constant         | Value    | Description |
|------------------|----------|-------------|
| <b>vbRed</b>     | &hFF     | Red         |
| <b>vbGreen</b>   | &hFF00   | Green       |
| <b>vbYellow</b>  | &hFFFF   | Yellow      |
| <b>vbBlue</b>    | &hFF0000 | Blue        |
| <b>vbMagenta</b> | &hFF00FF | Magenta     |
| <b>vbCyan</b>    | &hFFFF00 | Cyan        |
| <b>vbWhite</b>   | &hFFFFFF | White       |

### Comparison Constants

| Constant               | Value | Description                  |
|------------------------|-------|------------------------------|
| <b>vbBinaryCompare</b> | 0     | Perform a binary comparison  |
| <b>vbTextCompare</b>   | 1     | Perform a textual comparison |

### Date & Time Constants

| Constant                    | Value | Description                                                                              |
|-----------------------------|-------|------------------------------------------------------------------------------------------|
| <b>vbSunday</b>             | 1     | Sunday                                                                                   |
| <b>vbMonday</b>             | 2     | Monday                                                                                   |
| <b>vbTuesday</b>            | 3     | Tuesday                                                                                  |
| <b>vbWednesday</b>          | 4     | Wednesday                                                                                |
| <b>vbThursday</b>           | 5     | Thursday                                                                                 |
| <b>vbFriday</b>             | 6     | Friday                                                                                   |
| <b>vbSaturday</b>           | 7     | Saturday                                                                                 |
| <b>vbUseSystemDayOfWeek</b> | 0     | Use the day of the week specified in your system settings for the first day of the week. |
| <b>vbFirstJan1</b>          | 1     | Use the week in which January 1 occurs (default).                                        |
| <b>vbFirstFourDays</b>      | 2     | Use the first week that has at least four days in the new year.                          |
| <b>vbFirstFullWeek</b>      | 3     | Use the first full week of the year.                                                     |

### Date Format Constants

| Constant             | Value | Description                                                                                                                                                                                                                              |
|----------------------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>vbGeneralDate</b> | 0     | Display a date and/or time. For real numbers, display a date and time. If there is no fractional part, display only a date. If there is no integer part, display time only. Date and time display is determined by your system settings. |
| <b>vbLongDate</b>    | 1     | Display a date using the long date format specified in your computer's regional settings.                                                                                                                                                |
| <b>vbShortDate</b>   | 2     | Display a date using the short date format specified in your computer's regional settings.                                                                                                                                               |
| <b>vbLongTime</b>    | 3     | Display a time using the long time format specified in your computer's regional settings.                                                                                                                                                |
| <b>vbShortTime</b>   | 4     | Display a time using the short time format specified in your computer's regional settings.                                                                                                                                               |

### Miscellaneous Constants

| Constant             | Value       | Description                                                   |
|----------------------|-------------|---------------------------------------------------------------|
| <b>vbObjectError</b> | -2147221504 | User-defined error numbers should be greater than this value. |

### Box Constants – Buttons & Icons

| Constant          | Value | Description                    |
|-------------------|-------|--------------------------------|
| <b>vbOKOnly</b>   | 0     | Display OK button only.        |
| <b>vbOKCancel</b> | 1     | Display OK and Cancel buttons. |

| Constant                  | Value | Description                                                                                                                                                                                                                                        |
|---------------------------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>vbAbortRetryIgnore</b> | 2     | Display Abort, Retry, and Ignore buttons.                                                                                                                                                                                                          |
| <b>vbYesNoCancel</b>      | 3     | Display Yes, No, and Cancel buttons.                                                                                                                                                                                                               |
| <b>vbYesNo</b>            | 4     | Display Yes and No buttons.                                                                                                                                                                                                                        |
| <b>vbRetryCancel</b>      | 5     | Display Retry and Cancel buttons.                                                                                                                                                                                                                  |
| <b>vbCritical</b>         | 16    | Display Critical Message icon.                                                                                                                                                                                                                     |
| <b>vbQuestion</b>         | 32    | Display Warning Query icon.                                                                                                                                                                                                                        |
| <b>vbExclamation</b>      | 48    | Display Warning Message icon.                                                                                                                                                                                                                      |
| <b>vbInformation</b>      | 64    | Display Information Message icon.                                                                                                                                                                                                                  |
| <b>vbDefaultButton1</b>   | 0     | First button is the default.                                                                                                                                                                                                                       |
| <b>vbDefaultButton2</b>   | 256   | Second button is the default.                                                                                                                                                                                                                      |
| <b>vbDefaultButton3</b>   | 512   | Third button is the default.                                                                                                                                                                                                                       |
| <b>vbDefaultButton4</b>   | 768   | Fourth button is the default.                                                                                                                                                                                                                      |
| <b>vbApplicationModal</b> | 0     | Application modal. The user must respond to the message box before continuing work in the current application.                                                                                                                                     |
| <b>vbSystemModal</b>      | 4096  | 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. |

#### Box Constants – Selected Button

| Constant        | Value | Description                |
|-----------------|-------|----------------------------|
| <b>vbOK</b>     | 1     | OK button was clicked.     |
| <b>vbCancel</b> | 2     | Cancel button was clicked. |
| <b>vbAbort</b>  | 3     | Abort button was clicked.  |
| <b>vbRetry</b>  | 4     | Retry button was clicked.  |
| <b>vbIgnore</b> | 5     | Ignore button was clicked. |
| <b>vbYes</b>    | 6     | Yes button was clicked.    |
| <b>vbNo</b>     | 7     | No button was clicked.     |

#### String Constants

| Constant             | Value                        | Description                                                                     |
|----------------------|------------------------------|---------------------------------------------------------------------------------|
| <b>vbCr</b>          | Chr(13)                      | Carriage return                                                                 |
| <b>VbCrLf</b>        | Chr(13) & Chr(10)            | Carriage return...linefeed combination                                          |
| <b>vbFormFeed</b>    | Chr(12)                      | Form feed; not useful in Microsoft Windows                                      |
| <b>vbLf</b>          | Chr(10)                      | Line feed                                                                       |
| <b>vbNewLine</b>     | Chr(13) & Chr(10) or Chr(10) | Platform-specific newline character; whatever is appropriate for the platform   |
| <b>vbNullChar</b>    | Chr(0)                       | Character having the value 0                                                    |
| <b>vbNullString</b>  | String having value 0        | Not the same as a zero-length string (""); used for calling external procedures |
| <b>vbTab</b>         | Chr(9)                       | Horizontal tab                                                                  |
| <b>vbVerticalTab</b> | Chr(11)                      | Vertical tab; not useful in Microsoft Windows                                   |

#### Tristate Constants

| Constant            | Value | Description                                    |
|---------------------|-------|------------------------------------------------|
| <b>vbUseDefault</b> | -2    | Use default from computer's regional settings. |
| <b>vbTrue</b>       | -1    | TRUE                                           |
| <b>vbFalse</b>      | 0     | FALSE                                          |

## VarType Constants

| Constant                  | Value | Description                                |
|---------------------------|-------|--------------------------------------------|
| <code>vbEmpty</code>      | 0     | Uninitialized (default)                    |
| <code>vbNull</code>       | 1     | Contains no valid data                     |
| <code>vbInteger</code>    | 2     | Integer subtype                            |
| <code>vbLong</code>       | 3     | Long subtype                               |
| <code>vbSingle</code>     | 4     | Single subtype                             |
| <code>vbDouble</code>     | 5     | Double subtype                             |
| <code>vbCurrency</code>   | 6     | Currency subtype                           |
| <code>vbDate</code>       | 7     | Date subtype                               |
| <code>vbString</code>     | 8     | String subtype                             |
| <code>vbObject</code>     | 9     | Object                                     |
| <code>vbError</code>      | 10    | Error subtype                              |
| <code>vbBoolean</code>    | 11    | Boolean subtype                            |
| <code>vbVariant</code>    | 12    | Variant (used only for arrays of variants) |
| <code>vbDataObject</code> | 13    | Data access object                         |
| <code>vbDecimal</code>    | 14    | Decimal subtype                            |
| <code>vbByte</code>       | 17    | Byte subtype                               |
| <code>vbArray</code>      | 8192  | Array                                      |

## 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

| Function Names           |                |                          |                          |
|--------------------------|----------------|--------------------------|--------------------------|
| Abs                      | Array          | Asc                      | Atn                      |
| CBool                    | CByte          | CCur                     | CDate                    |
| CDbl                     | Chr            | Clnt                     | CLng                     |
| Conversions              | Cos            | CreateObject             | CSng                     |
| CStr                     | Date           | DateAdd                  | DateDiff                 |
| DatePart                 | DateSerial     | DateValue                | Day                      |
| Derived Math             | Escape         | Eval                     | Exp                      |
| Filter                   | FormatCurrency | FormatDateTime           | FormatNumber             |
| FormatPercent            | GetLocale      | GetObject                | GetRef                   |
| Hex                      | Hour           | InputBox                 | InStr                    |
| InStrRev                 | Int, Fix       | IsArray                  | IsDate                   |
| IsEmpty                  | IsNull         | IsNumeric                | IsObject                 |
| Join                     | LBound         | LCase                    | Left                     |
| Len                      | LoadPicture    | Log                      | LTrim; RTrim; and Trim   |
| Maths                    | Mid            | Minute                   | Month                    |
| MonthName                | MsgBox         | Now                      | Oct                      |
| Replace                  | RGB            | Right                    | Rnd                      |
| Round                    | ScriptEngine   | ScriptEngineBuildVersion | ScriptEngineMajorVersion |
| ScriptEngineMinorVersion | Second         | SetLocale                | Sgn                      |
| Sin                      | Space          | Split                    | Sqr                      |
| StrComp                  | String         | StrReverse               | Tan                      |
| Time                     | Timer          | TimeSerial               | TimeValue                |
| TypeName                 | UBound         | UCase                    | Unescape                 |
| VarType                  | Weekday        | WeekdayName              | Year                     |

## 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

| Error Number | Description                                                   |
|--------------|---------------------------------------------------------------|
| 5            | Invalid procedure call or argument                            |
| 6            | Overflow                                                      |
| 7            | Out of memory                                                 |
| 9            | Subscript out of range                                        |
| 10           | This array is fixed or temporarily locked                     |
| 11           | Division by zero                                              |
| 13           | Type mismatch                                                 |
| 14           | Out of string space                                           |
| 17           | Can't perform requested operation                             |
| 28           | Out of stack space                                            |
| 35           | Sub or function not defined                                   |
| 48           | Error in loading DLL                                          |
| 51           | Internal error                                                |
| 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                                         |

| Error Number | Description                        |
|--------------|------------------------------------|
| 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 '='                                       |
| 1012         | Expected 'If'                                      |
| 1013         | Expected 'To'                                      |
| 1014         | Expected 'End'                                     |
| 1015         | Expected 'Function'                                |
| 1016         | Expected 'Sub'                                     |
| 1017         | Expected 'Then'                                    |
| 1018         | Expected 'Wend'                                    |
| 1019         | Expected 'Loop'                                    |
| 1020         | Expected 'Next'                                    |
| 1021         | Expected 'Case'                                    |
| 1022         | Expected 'Select'                                  |
| 1023         | Expected expression                                |
| 1024         | Expected statement                                 |
| 1025         | Expected end of statement                          |
| 1026         | Expected integer constant                          |
| 1027         | Expected 'While' or 'Until'                        |
| 1028         | Expected 'While,' 'Until,' or end of statement     |
| 1029         | Expected 'With'                                    |
| 1030         | Identifier too long                                |
| 1037         | Invalid use of 'Me' keyword                        |
| 1038         | 'loop' without 'do'                                |
| 1039         | Invalid 'exit' statement                           |
| 1040         | Invalid 'for' loop control variable                |
| 1041         | Name redefined                                     |
| 1042         | Must be first statement on the line                |
| 1044         | Cannot use parentheses when calling a Sub          |
| 1045         | Expected literal constant                          |
| 1046         | Expected 'In'                                      |
| 1047         | Expected 'Class'                                   |
| 1048         | Must be defined inside a Class                     |
| 1049         | Expected Let or Set or Get in property declaration |

| Error Number | Description                                                            |
|--------------|------------------------------------------------------------------------|
| 1050         | Expected 'Property'                                                    |
| 1051         | Number of arguments must be consistent across properties specification |
| 1052         | Cannot have multiple default property/method in a Class                |
| 1053         | Class initialize or terminate do not have arguments                    |
| 1054         | Property Set or Let must have at least one argument                    |
| 1055         | Unexpected 'Next'                                                      |
| 1057         | 'Default' specification must also specify 'Public'                     |
| 1058         | 'Default' specification can only be on Property Get                    |

## Tips & Tricks

### VBScript Editor IntelliSense

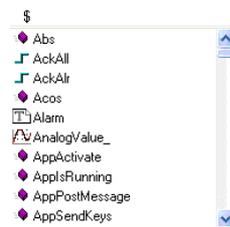
IntelliSense provides an array of options that make language references easily accessible. When coding, you do not need to leave the Code Editor or the Immediate Mode command window to perform searches on language elements. You can keep your context, find the information you need, insert language elements 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.

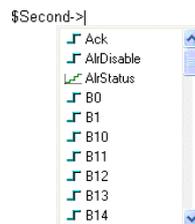
When you type the \$ character on any VBScript interface, a list box will automatically open with the list of all tags available for the current project as well as all functions from the Built-in Scripting Language.



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:



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:

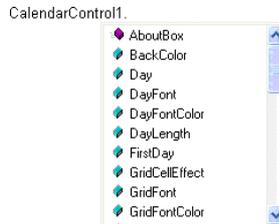


The items are displayed in alphabetic order, and each item has an icon to identify its main type, as follows:

| Icon | Type        |
|------|-------------|
|      | Boolean Tag |
|      | Integer Tag |
|      | Real Tag    |
|      | String Tag  |
|      | Class Tag   |

| Icon                                                                              | Type                                          |
|-----------------------------------------------------------------------------------|-----------------------------------------------|
|  | Function from the Built-in Scripting Language |

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:

| Icon                                                                              | Type                             |
|-----------------------------------------------------------------------------------|----------------------------------|
|  | Property from the ActiveX object |
|  | Method from the ActiveX object   |

### 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.

```
$FileCopy(

FileCopy(strSourceFile, strTargetFile))
```

### 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.

| Category                     | Feature/Keyword          |
|------------------------------|--------------------------|
| Declarations                 | Class                    |
| Miscellaneous                | Eval                     |
|                              | Execute                  |
| Objects                      | RegExp                   |
| Script Engine Identification | ScriptEngine             |
|                              | ScriptEngineBuildVersion |

| Category | Feature/Keyword          |
|----------|--------------------------|
|          | ScriptEngineMajorVersion |

The following table lists VBA features that VBScript does not have.

| Category                | Omitted Feature/Keyword                                                  |
|-------------------------|--------------------------------------------------------------------------|
| Array Handling          | Option Base                                                              |
|                         | Declaring arrays with lower bound <> 0                                   |
| Collection              | Add, Count, Item, Remove                                                 |
|                         | Access to collections using ! character                                  |
| Conditional Compilation | #Const                                                                   |
|                         | #If...Then...#Else                                                       |
| Control Flow            | DoEvents                                                                 |
|                         | GoSub...Return, GoTo                                                     |
|                         | On Error GoTo                                                            |
|                         | On...GoSub, On...GoTo                                                    |
|                         | Line numbers, Line labels                                                |
| Conversion              | CVar, CDate                                                              |
|                         | Str, Val                                                                 |
| Data Types              | All intrinsic data types except Variant                                  |
|                         | Type...End Type                                                          |
| Date/Time               | Date statement, Time statement                                           |
| DDE                     | LinkExecute, LinkPoke, LinkRequest, LinkSend                             |
| Debugging               | Debug.Print                                                              |
|                         | End, Stop                                                                |
| Declaration             | Declare (for declaring DLLs)                                             |
|                         | Optional                                                                 |
|                         | ParamArray                                                               |
|                         | Static                                                                   |
| Error Handling          | Erl                                                                      |
|                         | Error                                                                    |
|                         | Resume, Resume Next                                                      |
| File Input/Output       | All traditional Basic file I/O                                           |
| Financial               | All financial functions                                                  |
| Object Manipulation     | TypeOf                                                                   |
| Objects                 | Clipboard                                                                |
|                         | Collection                                                               |
| Operators               | Like                                                                     |
| Options                 | Deftype                                                                  |
|                         | Option Base                                                              |
|                         | Option Compare                                                           |
|                         | Option Private Module                                                    |
| Select Case             | Expressions containing the <b>Is</b> keyword or any comparison operators |
|                         | Expressions containing a range of values using the <b>To</b> keyword     |
| Strings                 | Fixed-length strings                                                     |

| Category      | Omitted Feature/Keyword   |
|---------------|---------------------------|
|               | LSet, RSet                |
|               | Mid Statement             |
|               | StrConv                   |
| Using Objects | Collection access using ! |

## 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:

| Event                   | Order of execution                                                                                                                                                                                                  |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| When opening the screen | <ul style="list-style-type: none"> <li>Screen_OnOpen() sub-routine from the Screen Script interface (VBScript language)</li> <li>On Open from the Screen Logic interface (Built-in Scripting Language)</li> </ul>   |
| When closing the screen | <ul style="list-style-type: none"> <li>On Close from the Screen Logic interface (Built-in Scripting Language)</li> <li>Screen_OnClose() sub-routine from the Screen Script interface (VBScript language)</li> </ul> |

## 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.

```
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:

```
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:

```
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:

```
Dim DegreesFahrenheit
```

You declare multiple variables by separating each variable name with a comma. For example:

```
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:

```
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 variables in VBScript are handled differently than Boolean tags are handled in IWS. The Boolean states of FALSE and TRUE have the same meaning, but the numeric values of TRUE are different, as shown in the table below.

| Boolean State | Numeric Value in... |                   |
|---------------|---------------------|-------------------|
|               | Project Tag         | VBScript Variable |
| FALSE         | 0                   | 0                 |
| TRUE          | 1                   | -1                |

 **Note:** In VBScript, **False** and **True** are also reserved as [keywords](#).

This difference in how Booleans are handled can seriously affect runtime behavior if you use VBScript in your project. Logical and arithmetic operations — especially the [NOT operator](#) — could change tag values in unexpected ways.

You can change this behavior, if necessary, by changing your project's runtime settings.

### Changing How VBScript Handles Boolean Tags

You can change the way VBScript handles Boolean tags by activating "VB Boolean" mode. To do this, edit your project file (*project\_name.app*) to change the following setting:

```
[Script]
VBBoolean=value
```

If **VBBoolean** is set to 0, then the project will behave as described above: all VBScript functions and operations will read/write a value of -1 for TRUE to Boolean tags. This is the default setting for projects created with InduSoft Web Studio v6.1+SP3 or earlier and then updated to v6.1+SP4, in order to maintain backward compatibility.

If **VBBoolean** is set to 1, then VBScript — as it is implemented within IWS — will read/write a value of 1 for TRUE to Boolean tags. (This does not affect Integer or Real tags.) This is the default setting for projects created with InduSoft Web Studio v6.1+SP4 or later.

 **Note:** Be careful when defining a custom property on a [Linked Symbol](#) using the **#Label:@Pointer** syntax. For example:

```
'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
End If
```

### 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:

```
[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.

| Arithmetic         | Comparison      | Logical |
|--------------------|-----------------|---------|
| Negation (-)       | Equality (=)    | Not     |
| Exponentiation (^) | Inequality (<>) | And     |

| Arithmetic                         | Comparison                    | Logical |
|------------------------------------|-------------------------------|---------|
| Multiplication and division (*, /) | Less than (<)                 | Or      |
| Integer division (\)               | Greater than (>)              | Xor     |
| Modulus arithmetic (Mod)           | Less than or equal to (<=)    | Eqv     |
| Addition and subtraction (+, -)    | Greater than or equal to (>=) | Imp     |
| String concatenation (&)           | Is                            | &       |

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...

**result = NOT expression**

| If expression is... | Then result is... |
|---------------------|-------------------|
| -3                  | 2                 |
| -2                  | 1                 |
| -1                  | 0                 |
| 0                   | -1                |
| 1                   | -2                |
| 2                   | -3                |
| 3                   | -4                |

 **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...

**result = NOT expression**

| If expression is... | Then result is... |
|---------------------|-------------------|
| 0                   | 1                 |
| ≠0                  | 0                 |

## 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:

```
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:

```
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**:

```
Sub AlertUser(value)
 If value = 0 Then
 AlertLabel.ForeColor = vbRed
 AlertLabel.Font.Bold = True
 AlertLabel.Font.Italic = True
 Else
 AlertLabel.ForeColor = vbBlack
 AlertLabel.Font.Bold = False
 AlertLabel.Font.Italic = False
 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:

```
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
```

```
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:

```
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.

```
Sub ChkFirstWhile()
 Dim counter, myNum
 counter = 0
 myNum = 20
 Do While myNum > 10
 myNum = myNum - 1
 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.

```
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.

```
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.

```
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:

```
//Access the value of the property Day from the CalendarControl1 ActiveX object
CalendarControl1.Day
```

```
//Triggers the method AboutBox from the CalendarControl1 ActiveX object
CalendarControl1.AboutBox
```

## Windows Embedded Support

CEView also supports VBScript. The hardware manufacturer of the Microsoft Windows Embedded device must enable the support for VBScript on it, so CEView will be able to execute the scripts configured in VBScript language on the device.

The **MsgBox()** and **InputBox()** functions can be specifically enabled/disabled by the hardware manufacturer when the image for the Windows Embedded device is created.

If you are not sure if the image loaded on your device supports VBScript, please consult the hardware manufacturer.

