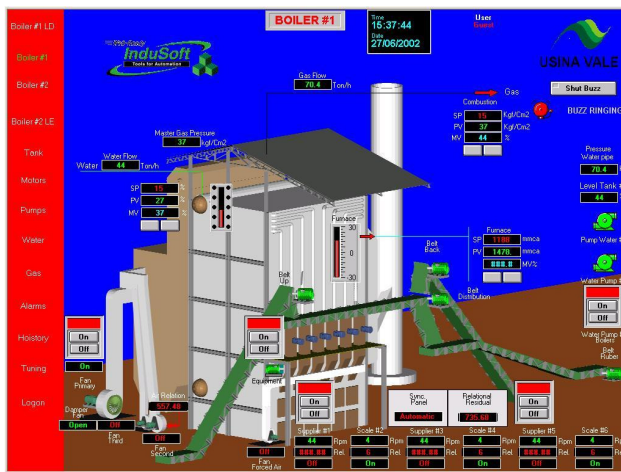


InduSoft Web Studio

User's Guide and Technical Reference Manual for InduSoft Web Studio Version 6.1 SP6



www.InduSoft.com
info@indusoft.com

Copyright © 2003-2009 by InduSoft®. All rights reserved worldwide.

No part of this publication may be reproduced or transmitted in any form or by any means without written authorization from InduSoft.

InduSoft is a registered trademark of InduSoft. CEView is a trademark of InduSoft.

The information contained within this document is subject to change without notice.

InduSoft does not assume responsibility for any errors or inaccuracies that may occur in this publication.

Windows, Windows 2000, Windows CE, Windows XP, and Windows Vista are registered trademarks of Microsoft Corporation in the United States and other countries.

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

PN: IND-UG-001



Contents

About this Publication	xi
Who Should Read this Publication.....	xii
Conventions	xii
Text Conventions.....	xii
Mouse and Selection Conventions	xiii
Windows Conventions	xiii
Additional Resources	xv
Working with Technical Support	xv
Chapter 1: What is InduSoft Web Studio?	1-1
Product Overview.....	1-1
Product Features.....	1-2
Chapter 2: Installing InduSoft Web Studio	2-1
Before You Begin	2-1
Installing IWS	2-3
Installing CEView	2-5
Uninstalling InduSoft Web Studio and CEView	2-6
Licensing Your Product.....	2-8
Product Versions	2-9
Invalid Licenses	2-10
Execution Modes.....	2-10
Protection Types	2-12
License Settings.....	2-13
Installing Hardkey Licenses for InduSoft Web Studio	2-14
Installing a New License.....	2-14
Upgrading the Current Hardkey License	2-14
Installing a SoftKey License for InduSoft Web Studio	2-16
Installing a New License.....	2-16
Upgrading the Current License.....	2-17
Installing a SoftKey License for CEView.....	2-21
Installing or Upgrading a License (Locally)	2-21
Installing or Upgrading a License (Remotely)	2-24
Chapter 3: Working with the IWS Development Environment	3-1
Navigating the Development Environment Window.....	3-1
Using the Title Bar	3-2
Using the Menu Bar	3-3
Using the Toolbars	3-5
Using the Workspace	3-6
Using the Screen/Worksheet Editor.....	3-11
Using the Database Spy	3-23
Using the Output Window	3-24
Using the Status Bar.....	3-26

Customizing the Workspace	3-28
Standard Interfaces.....	3-28
Object Properties Dialog.....	3-28
Virtual Keyboard	3-30
Fonts.....	3-31
Color Interface	3-33
Performing Common Tasks	3-37
Accessing Projects and Files.....	3-38
Using Common Buttons.....	3-39
Managing the Development Environment Windows.....	3-40
Using Select All	3-43
Cutting, Copying, Pasting Objects.....	3-43
Finding System Information.....	3-44
Searching for Tags and Screen Objects	3-45
Using the Tag Properties Toolbar	3-46
Replacing Tags.....	3-49
Testing Displays	3-50
Verifying the Application.....	3-51
Running Applications.....	3-52
Restoring Defaults	3-52
Saving Your Work.....	3-52
Printing Project Screens	3-53
Tips and Tricks.....	3-55
Configuring the Focus of the Object Properties Window on the Screen Editor	3-55
Importing a Database.....	3-56
Importing from Other InduSoft Web Studio Databases.....	3-60
Importing from OPC Server Databases.....	3-62
Importing from CSV Databases.....	3-63
Importing from ODBC Databases.....	3-64
Importing from PanelBuilder32™ Databases	3-65
Importing from RSLogix™ 5000 CSV Databases	3-67
Importing from OMRON CX Programmer Databases	3-68
Importing from PanelMate Plus™ Databases	3-70
Importing from TwinCAT PLC Databases	3-71
Development Modules	3-75
Graphics	3-75
Tasks	3-76
General Communications.....	3-76
Chapter 4: Understanding IWS Structure	4-1
Understanding the Internal Structure and Data Flow.....	4-1
Executing/Switching IWS Modules	4-3
Executing/Switching the Background Task.....	4-9
Chapter 5: Working with Tags.....	5-1
What is a Tag?	5-1

Designing a Tag	5-3
Choosing the Tag Type	5-3
Choosing a Tag Data Type.....	5-6
Sharing Tags with a Web Thin Client	5-7
Choosing Tag Properties	5-7
Creating Application Database Tags	5-13
Adding Tags to the Application Datasheet	5-13
Adding Tags “On-the-Fly”	5-14
Resetting the Tags Database	5-15
Creating Classes	5-17
Setting Tag Properties	5-19
Configuring the Parameters Tab Properties	5-19
Configuring the Alarms - Type Tab Properties	5-21
Configuring the History Tab Properties	5-23
Using Tags in Your Application	5-24
Editing Tags	5-24
From the Application Tags Datasheet	5-25
Deleting Tags	5-26
Chapter 6: Creating and Configuring a Project	6-1
Creating a New Project Application	6-1
Using a Template	6-4
Specifying a Default Screen Resolution	6-4
Sharing PC-Based Control Software Program Database Tags	6-4
Specifying Additional Project Settings	6-7
Providing Project Identification Information	6-8
Setting the Options Tab Parameters	6-9
Setting the Runtime Desktop Parameters	6-19
Setting the Communication Parameters.....	6-23
Specify Web Thin Client Parameters	6-25
Enabling Warning Messages.....	6-29
Starting Runtime Modules on the Target System	6-31
Chapter 7: Configuring Screens and Graphics	7-1
Working in an Object-Oriented Environment	7-1
Working with Screen Attributes	7-2
Specifying Background Color.....	7-7
Using Objects and Dynamics	7-8
Using the Mode Toolbar	7-8
Using the Align and Distribute Toolbar	7-10
Using Groups of Screens	7-16
Using the Web Toolbar	7-18
Using the Bitmap Toolbar	7-22
Using the Static Objects Toolbar	7-24
Using the Dynamic Properties Toolbar.....	7-31
Using the Active Objects Toolbar	7-44

Using the Library	7-130
Using Paste Link.....	7-131
Symbols Folder.....	7-131
Chapter 8: Configuring Task Worksheets.....	8-1
Configuring an Alarms Task.....	8-2
Configuring a Trend Task.....	8-14
Converting Trend History Files from Binary to Text	8-21
Converting Trend History Files from Text to Binary	8-22
Creating Batch History.....	8-22
Configuring a Recipes Task.....	8-24
Configuring a Reports Task	8-26
Configuring an ODBC Task	8-28
Configuring a Math Task.....	8-31
Configuring a Scheduler Task.....	8-33
Configuring an External Databases Task	8-35
Database Connections	8-35
Database Worksheet	8-37
Chapter 9: Event Settings.....	9-1
Configuring the Events Settings	9-2
Chapter 10: Communication	10-1
Configuring a Driver	10-2
Configuring the Driver Worksheets.....	10-10
Executing the Driver	10-19
Configuring OPC	10-20
Configuring an OPC Client	10-20
OPC Troubleshooting	10-23
Configuring an OPC Server.....	10-24
Configuring TCP/IP	10-25
Configuring the Client	10-25
Setting Custom Parameters	10-25
Configuring the TCP/IP Server	10-26
Configuring DDE	10-27
Configuring DDE Client	10-28
Configuring the DDE Server	10-29
Chapter 11: Configuring a Security System.....	11-1
Entering a Password	11-1
Defining Groups	11-2
Defining Users.....	11-7
Remote Security System	11-14
Setting the Security Access Level	11-16
Logging On/Off.....	11-17
Chapter 12: Testing and Debugging Your Application	12-1
Testing Your Application	12-1
Debugging Applications from the Database Spy	12-2

Debugging Applications from the Output Window	12-4
Using the LogWin Module (NT and CE).....	12-7
Using Remote Tools.....	12-9
Using Remote Database Spy	12-10
Using Remote LogWin.....	12-11
Chapter 13: Configuring a Web Solution	13-1
Configuring the Application for Different Architectures	13-2
Architecture 1: Web server and Web Thin Clients in the same network.....	13-2
Architecture 2: Web server and Web Thin Clients in the same network; Web server and data server in different stations	13-4
Architecture 3: Redundant servers and Web Thin Client stations in the same network.....	13-6
Architecture 4: Web server and Web Thin Clients in different networks.....	13-8
Installing and Registering the ISSymbol Control Layer	13-11
How It Works	13-13
Testing the Application.....	13-14
Installing Secure Viewer as Alternative to Web Browser.....	13-17
Chapter 14: Managing Applications Remotely.....	14-1
Downloading the Application.....	14-1
Configuring the Target Station.....	14-1
Configuring the Development Station.....	14-3
Monitoring/Managing Applications from the Remote Station.....	14-6
Configuring Windows CE to Automatically Run an Application	14-6
Chapter 15: Scripting Languages: IWS and VBScript	15-1
Working with the IWS Scripting Language, Expressions, and Functions	15-1
Using Tags	15-1
Specifying Data Types	15-1
Accessing the Tags Database.....	15-2
Arithmetic Operators.....	15-2
Logic Operators	15-3
Using Functions	15-4
Overview of VBScript	15-5
VBScript in IWS.....	15-6
Global Procedures	15-8
Graphic Module - Graphics Script	15-11
Graphic Module - Screen Script	15-13
Graphic Module - Command Dynamic	15-15
Graphic Module - ActiveX Events.....	15-17
Background Task - Startup Script.....	15-19
Background Task - Script Groups	15-20
Language Reference.....	15-22
Constants.....	15-22
Errors	15-25
Functions	15-28
Keywords	15-29
Methods	15-29

Objects and Collections	15–29
Operators	15–30
Properties	15–31
Statements.....	15–32
Tips and Tricks.....	15–33
VBScript Editor - IntelliSense	15–33
VBScript compared to VBA	15–35
Screen Events	15–36
MsgBox() and InputBox() functions	15–37
Support for ActiveX objects	15–37
Logical Operator NOT	15–37
Boolean Tags and Boolean Variables	15–38
Windows CE Support	15–39
Scope and Lifetime of Variables.....	15–40
Declaring Variables	15–40
Creating Constants.....	15–40
Precedence of VBScript Operators	15–41
Using Conditional Statements	15–42
Looping Through Code.....	15–44
VBScript Procedures	15–47
Chapter 16: Using the Translation Tool/Editor	16–1
Project Settings for Automatic Translation.....	16–2
Configuring Object Properties for Screen Objects	16–3
Translation Editor.....	16–4
Editing Worksheets.....	16–5
Saving Your Worksheets.....	16–8
Executing the Translation Functions.....	16–9
Using the SetTranslationFile() Function	16–9
Using the Ext() Function.....	16–11
Closing the Translation Editor.....	16–12
Chapter 17: IWS Database Interface.....	17–1
General Concepts	17–2
SQL Relational Databases	17–2
History Format	17–4
Primary and Secondary Databases.....	17–5
Default Database	17–6
Linking the Database Through a Remote DB Provider	17–7
Configuring Database Settings	17–8
Database Configuration Dialog.....	17–9
Studio Database Gateway	17–13
Advanced Settings.....	17–15
Database Troubleshooting.....	17–25
Database FAQ	17–25
Database Appendix A: Using ODBC Databases	17–29

Database Appendix B: Using Microsoft SQL Server 17–30

Database Appendix C: Using ORACLE databases 17–32

Database Appendix D: Using Microsoft Access Databases 17–34

Database Appendix E: Using SQL Server CE 17–35

Database Appendix F: Using Sybase 17–36

Database Appendix G: Using Microsoft Excel 17–36

Database Appendix H: Using MySQL..... 17–40

Chapter 18: Troubleshooting 18–1

 Before Contacting Technical Support 18–2

 Verifying Your Application 18–4

 Common Errors 18–5

 Database & Security System..... 18–5

 Graphics 18–5

 Tasks 18–7

 Communication..... 18–8

 General Troubleshooting 18–10

Appendix A. InduSoft Web Studio Functions A-1

 Function Prototypes and Descriptions A-12

 Log Message Functions A-13

 Arithmetic Functions..... A-14

 Statistical Functions A-30

 Logarithmic Functions A-35

 Logical Functions A-38

 String Functions A-42

 Date and Time Functions A-73

 Trigonometric Functions..... A-81

 Opening and Closing Windows Functions A-89

 Security Functions..... A-94

 Module Activity Functions A-104

 File Functions A-138

 Graphic Functions A-162

 Translation Functions..... A-172

 Multimedia Functions A-174

 System Information Functions A-175

 Tags Database Functions A-208

 Loop Function A-212

 ODBC Functions A-213

 Mail Functions A-242

 Dial-Up Functions..... A-248

 ActiveX Functions A-271

 Event Logger Functions A-274

 FTP Functions A-276

 DB/ERP Functions A-281

Index 1



About this Publication

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

The information in this publication is organized into the following chapters:

- **This chapter:** Describes the purpose, content, and organization of the *InduSoft Web Studio 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 InduSoft Web Studio
 - Explains how to contact a technical support representative
- **Chapter 1. What is InduSoft Web Studio?** Provides a high-level overview of the product's uses, features, and functions.
- **Chapter 2. Installing InduSoft Web Studio:** Provides step-by-step instructions for installing, licensing, starting, and uninstalling InduSoft Web Studio and CEView.
- **Chapter 3. Working with the IWS Development Environment:** Describes the InduSoft Web Studio interface (or development environment), and explains some basic skills and techniques you must understand before creating a new application.
- **Chapter 4. Understanding IWS Structure:** Describes the internal structure of InduSoft Web Studio, including how data flows through the runtime modules and how these modules are executed.
- **Chapter 5. 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 applications.
- **Chapter 6. Creating and Configuring a Project:** Provides step-by-step instructions for creating and configuring a new project application.
- **Chapter 7. Configuring Screens and Graphics:** Explains how to use the different InduSoft Web Studio development tools to create your application screens and graphics.
- **Chapter 8. Configuring Task Worksheets:** Explains how to create and configure the different InduSoft Web Studio task worksheets for your project applications.
- **Chapter 9. Event Settings.** Describes the logging and event-retrieval features.
- **Chapter 10. Communication:** Describes how to configure InduSoft Web Studio to read and write your application tag variables to or from a device's memory. The information includes instructions for configuring drivers and OPC, TCP/IP, and DDE (Dynamic Data Exchange) communication.
- **Chapter 11. Configuring a Security System:** Explains how to set-up and manage a security system for your applications.
- **Chapter 12. Testing and Debugging Your Application:** Discusses how to test and debug applications 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.
- **Chapter 13. Configuring a Web Solution:** Explains how configure and run your application on the Web.

- **Chapter 14. Managing Applications Remotely:** Explains how to download, monitor, and debug applications from a remote runtime workstation.
- **Chapter 15. Scripting Languages: IWS and VBScript:** Describes InduSoft Web Studio's built-in scripting language, as well as the support for VBScript in IWS.
- **Chapter 16. Using the Translation Editor:** Explains how to use the InduSoft Web Studio Translation Editor to translate the text in your applications from one language to another.
- **Chapter 17. IWS Database Interface:** Explains how to connect InduSoft Web Studio to compatible databases.
- **Chapter 18. Troubleshooting:** Provides instructions for verifying applications, describes some common development errors, and explains what to do if you need to contact a support representative.
- **Appendix A. Studio Functions:** Contains tables and information describing the different functions available with InduSoft Web Studio and CEView.

Who Should Read this Publication

This *User Guide and Technical Reference* is a comprehensive document designed to provide useful information for both novice and advanced InduSoft Web Studio users.

- **New Users:** This publication uses a step-by-step, hands-on approach to the application 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 product applications.

 **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.

Conventions

This section describes the text, mouse, and Windows conventions used throughout all InduSoft publications.

Text Conventions

The following text conventions and formatting techniques are used in this publication to help you quickly identify certain kinds of information:

- File names and screen/message text are indicated using **bold, monospace** text (for example: `\DRunStartup.exe`).

When you are required to provide information, such as a file name, the entry is enclosed in angle brackets `<>` and indicated using **bold, italic monospace** text (for example, `<Project folder>\Redist\CEView\<Processor Type>\`).

In this example, you must type the project folder name and processor type to complete the entry.

- Buttons, menu options, and keyboard keys are indicated using a **narrow bold** typeface (for example, “Press the **Enter** key.”)
- Text requiring special emphasis (such as warnings, new terms, or product names) is indicated using *italics* to draw your attention to the item (for example, “*Do not* save the file.”)
- In addition, this publication segregates some text into **Tip**, **Note**, and **Caution** boxes to help you identify information quickly and easily.
 - **Tips** (⇒) provide useful information to save development time or to improve application performance.
 - **Notes** (⌘) provide information related to the surrounding text (usually the paragraph just preceding the note) to help you further understand a concept or to provide supplemental information.
 - **Cautions** (⚠) provide information necessary to prevent errors that can cause problems when running the application, and may result in damage.

Mouse and Selection Conventions

Because most PCs used for application development run a version of Windows with a mouse, this publication assumes you are using a mouse. Generally, a PC mouse is configured so that the left mouse button is the primary button and the right mouse button is the secondary button.

This publication uses the following mouse and selection conventions:

- **Double-click** means to quickly click twice on an object with the left mouse button.
- **Right-click** means to click once on an object with the right mouse button.
- **Click** and **Select** means to click once on an object with the left mouse button. In general, you *click* on buttons and *select* from lists.



Select also means you should use your pointing device to highlight or specify an item on the computer screen. Selecting an object with a touch screen 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 objects 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 object will move with the mouse cursor.
- Instructions to select multiple menu bar and/or menu options use arrows to illustrate the selection sequence. For example, if you see the following . . .
Select **Start** → **Programs** → **InduSoft Web Studio** → **Register** to register your product.
. . . you should click the Windows **Start** button, select the **Programs** option, select the **InduSoft Web Studio** option, and select the **Register** option.

Windows Conventions

This publication uses the following Windows conventions:

- **Dialog boxes** (or *dialogs*) are windows that allow you to enter information.
- **Text boxes** (or *fields*) are areas in a dialog where you can type in text.

- **Radio buttons**  are white circles in which a black dot appears or disappears when you click on the button. The dot indicates the option or function is *enabled* (active) and a clear circle indicates the option or function is *disabled* (inactive).
- **Check-boxes** are white squares in which a check () appears or disappears when you click on it. A check indicates the option or function is *enabled* (active) and a clear box indicates the option or function is *disabled* (inactive).
- **Buttons** are boxes containing text or graphics that perform an action within the program. Buttons appear “pressed” when you click on them.

Button Type	Description
	<p>Click to perform an action.</p> <p>For example, click Save to save your project or click Cancel to cancel the current operation.</p>
	<p>Click a button containing text with ellipses (...) to open a related dialog.</p> <p>For example, click the Options button to open the Options dialog.</p>
	<p>Click to perform an action.</p> <p>For example, click the Open Library button to open the Symbol Library dialog.</p>
	<p>Some buttons are <i>toggles</i>, which means that clicking the button turns a particular feature, mode, or display on or off.</p> <p>For example, click the Output Window button to switch between turning the Output window (LogWin) display on and off in the development environment window.</p>

Button Examples Table

- **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.
- **Interface** refers to the entire InduSoft Web Studio window (*development environment*).
- **Dockable windows** are windows that you can drag to an edge of the interface and merge with that edge.
- **Toolbars** are dockable windows containing only buttons and text boxes.

Additional Resources

For more information about InduSoft Web Studio, the following resources are available:

Resource	Description
Related Publications	<ul style="list-style-type: none"> • <i>InduSoft Web Studio Getting Started Guide</i>: Designed for first-time users, this publication contains information about the basic functions of InduSoft Web Studio. This publication is provided in the Documentation folder on the IWS CD-ROM or from the Help menu located on the main menu bar. • <i>Individual Driver User Guides</i>: Explain how to configure individual InduSoft drivers, according to their unique protocol characteristics. One customized user guide is included with each InduSoft driver. These publications are provided in the DRV subdirectory of the InduSoft Web Studio folder on the IWS CD-ROM or from the Help menu located on the main menu bar. • <i>InduSoft Web Studio Frequently Asked Questions (FAQ)</i>: Lists the most commonly asked questions and answers about InduSoft Web Studio.
InduSoft Web Site	Visit www.InduSoft.com for information about InduSoft products, available downloads and demos, product FAQ, driver information, training opportunities, distributors, and the latest InduSoft news.

Information Resources Table

Working with Technical Support

InduSoft's expert support engineers are committed to resolving your issues and questions as quickly and accurately as possible. Our technical support centers are located in Austin, Texas and Sao Paulo, Brazil.

Support Office	Contact Information
Austin, Texas	Telephone: 877-INDUSOFT (877-463-8763) Fax: 512-349-0375 Email: support@indusoft.com
Saõ Paulo, Brazil	Telephone: +55 11 5505-5676 Fax: +55 11 5505-5676 ext. 13 Email: support@indusoft.com.br

InduSoft Support Offices Table

Your technical support options include:

- Electronic support
- Telephone support
- Product version updates
- Access to the InduSoft technical support Web page (<http://www.InduSoft.com>)
- FaxBack system

When requesting technical support, please have the following information available:

Name of Submitter: _____

Contact Information: _____

Preferred contact method: E-mail Telephone Mobile Phone Other

Industrial Sector (For example Automotive, Pharmaceutical, Manufacturing): _____

Number of supervisor stations: One Two Three Other

Field Equipment (For Example, Allen Bradley Data Highway Plus):

Manufacturer: _____

Model: _____

Amount: _____

Protocol: _____

InduSoft Driver(s): _____

Computer Environment:

Topic	Sub-Item	Characteristics
Hardware	32-bit Workstation	
	Hardware	_____
	Vendor	_____
	Model number	_____
	Processor	_____
	Clock	_____
	RAM Memory	_____
Other information you think engineering should know about the hardware:		

Customer Information Form (continued)

Computer Environment (cont.):

Topic	Sub-Item	Characteristics
Software	Operating System	
	Hardware	
	Type	_____
	Version	_____
	Service Pack	_____
	Language	_____
Other information you think engineering should know about the software:		

Topic	Sub-Item	Characteristics
Software	Microsoft Internet Explorer	
	Hardware	_____
	Software	_____
	Version	_____
	Service Pack	_____
	Language	_____
Other information you think engineering should know about the Microsoft Internet Explorer installation:		

Topic	Sub-Item	Characteristics
Software	InduSoft	
	Hardware	_____
	Software	_____
	Version	_____
	Service Pack	_____
Other information you think engineering should know about the Microsoft Internet Explorer installation:		

Computer Environment (cont.):

Topic	Sub-Item	Characteristics
Software	CEView	
	Hardware	
	Software	
	Version	
	Service Pack	

Other information you think engineering should know about the CEView installation:

Problem Description:

Customer Information Form



Chapter 1: What is InduSoft Web Studio?

InduSoft Web Studio (or *IWS*) 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) applications for:

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

Product Overview

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

These applications consist of animated operator-interface screens, configurable *PLC* (programmable logic controller) drivers and other controllable I/O devices, an application tags database, and optional modules such as alarm monitors, logic, trend charts, recipes, schedulers, and a security system. IWS applications 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 an application, you can run it on your development workstation or download the application 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 application and then react to, display, store and upload the data.

The InduSoft Web Studio product consists of:

- *Development system* software that runs on a desktop, laptop or industrial PC running Windows® 2K/XP/Vista
- *Runtime system* software that runs on an operator interface workstation running Windows 2K/XP/Vista or Windows CE

 **Note:**

The runtime system software (CEView) for the Windows CE operating system is usually 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:
 - Shortcut menus, which can be accessed 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 *dynamics* (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
 - Dynamic properties such as bar graphs, color, resizing, position, hide/unhide, rotation, command, hyperlink, and text input/output
 - Online and historical alarm list displays
 - Online and historical trending
 - Alignment and distribution tools
 - Background bitmap layer creation and editing
 - Graphics importation
 - ActiveX object containers
- Online remote management and configuration
- Microsoft DNA architecture compliance, with full OPC and XML support
- Web interface enabled, which exports application screens to a “thin” client through the Internet/intranet and by exchanging data online through the TCP/IP protocol
- Symbol library with more than 100 symbols and dynamic 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
 - Online system and network diagnostics
- Powerful and flexible *Tags 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 an application 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 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
- 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 application, which means modular worksheets and screens can be merged easily to other applications
- Full integration with PC-based control programs (imports tag databases) such as ISaGRAF, SteepleChase, Think&Do, and ASAP
- 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 application responsibility. However, IWS does not support some features in certain product types (such as *CEView*). You can review the **TargetVersions.pdf** document on the InduSoft Web Studio CD-ROM for detailed information about the limitations of each product-type limitations.



Chapter 2: Installing InduSoft Web Studio

This chapter explains how to install, license, run, and uninstall InduSoft Web Studio (IWS) and CEView™.

You can install IWS from the InduSoft Web Studio CD-ROM or create 3.5-inch installation diskettes. For Windows CE applications, you can use IWS to download CEView (runtime software) to the Windows CE HMI by serial or TCP/IP link.

The IWS installation program automatically creates the necessary directories, copies files to your hard drive, and creates the **InduSoft Web Studio** icon in your *Desktop* folder.

Notes:

If you will be using Windows CE:

- You use IWS to download CEView (the runtime software) to the Windows CE HMI using a serial or TCP/IP link.
- When you install InduSoft Web Studio on Windows 2K/XP/Vista computers, IWS stores the CEView runtime files in the following folder:

`<InduSoft Web Studio Folder>\Redist\CEView\<Processor Type>\`

Where:

- `<Installation Folder>` is the installation directory chosen during the installation (`C:\Program Files\InduSoft Web Studio` is the default installation directory).
- `<Processor Type>` is the processor platform. InduSoft provides a CEView runtime for most processor platforms supported by the WinCE operating system.


Before You Begin


Note:

You must have Administrator privileges for the Windows 2K/XP/Vista workstation on which you are installing (or uninstalling) InduSoft Web Studio.


Before installing the IWS software, you must do the following:

- Uninstall any older versions of IWS (or install the newer version to a different directory). Also, you cannot install the same version of IWS in two different paths on the same computer.
- System Requirements
To develop applications with IWS, you must install the following hardware and software:
 - PC-compatible computer with a Intel® Pentium IV-compatible processor, such as Centrino, AMD, Celeron, or higher
 - Windows 2000/XP/Vista or Windows 2003 Server operating system for the development system
 - Windows 2000/XP/Vista or Windows CE v3.00 operating system for runtime

 **Caution:**
InduSoft Web Studio and its remote clients are not supported any operating systems that are no longer supported by Microsoft itself.

 **Note:**
We recommend Windows XP Professional over Windows XP Home Edition/Media Center, because it includes Internet Information Services (IIS) that can be used as your application's Web server.

- Minimum of 256MB random-access memory (RAM) (512MB or higher recommended)
- MS Internet Explorer 6.0 or higher
- Minimum of 500MB free hard disk space to install the product and the application (the history files/databases will demand additional disk space)
- Ethernet adapter
- 100% IBM-compatible VGA or SVGA display adapter with 64MB Video RAM (VRAM) or higher
- Microsoft-compatible pointing device (such as a mouse, trackball, or touch-screen)
- Standard keyboard with function keys F1 through F12
- CD-ROM drive (optional – to install the system files)
- Parallel printer port (optional – to be used with Parallel Hardkey licensing method)
- USB port (optional – to be used with USB Hardkey licensing method)
- Serial COM ports and adapters (optional)

 **Notes:**

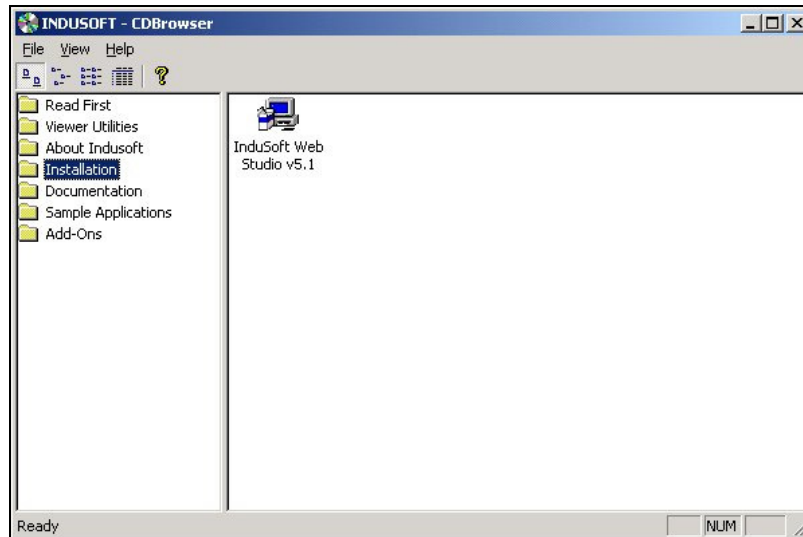
- The requirements described above are based on typical applications. Depending on your specific application, the minimum requirements may vary.
- Applications developed with InduSoft Web Studio can run under devices powered with the Windows CE operating system (Windows CE v3.0 or Windows CE .NET) such as industrial HMIs or PDAs (PocketPC). Consult your vendor for the hardware requirements when running your application under the Windows CE operating system.
- When using a Hardkey instead of a Softkey to license the product, either the parallel or the USB port must be available in the local computer.
- Some of the items listed above as optional may be mandatory depending on your application. For instance, if you need to exchange data with a PLC via a serial interface, the computer must provide a serial COM port.

Installing IWS

Use the following procedure to install IWS from the CD-ROM:

- ☑ Turn on the power to your development computer and be sure that no other programs are running.
- ☑ Insert the installation CD-ROM into the computer's CD-ROM drive.

A *CDBrowser* window should display automatically:



CDBrowser Window

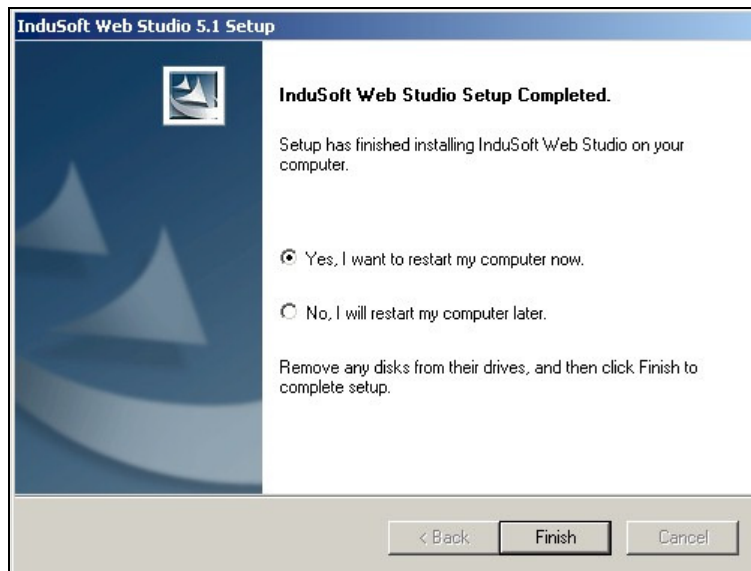
If the *InduSoft CDBrowser* window does not display, you can start the program from the Windows Explorer. Navigate to the `<D>:\Installation` directory (where `<D>` is your CD-ROM drive), and run the **Setup.exe** file (clicking the *InduSoft Web Studio* icon launches this file).

The *CDBrowser* window contains the following folders:

- **Read First:** Contains the *Readme* document (in **.pdf** format) with important information you should read before using the current product.
- **Viewer Utilities:** Contains *Microsoft PowerPoint Viewer* (needed to view the InduSoft presentations provided in **.ppt** format), *Adobe® Reader®* (needed to view the InduSoft documents provided in **.pdf** format), and *WinZip®* (for unzipping the compressed files).
- **About InduSoft:** Contains a short PowerPoint presentation about InduSoft.
- **Installation:** Contains an InduSoft Web Studio icon. Double-clicking this icon starts the installation program.
- **Documentation:** Contains all IWS documentation in **.pdf** format.
- **Sample Applications:** Contains sample applications to help you develop your own applications and provide examples for you while you use InduSoft Web Studio.
- **Add-Ons:** Contains a demo version of the *Symbol Factory ActiveX* program—an extensive symbol library that simplifies application development. Also contains the *PanelBuilder Import Wizard* for importing your existing PanelBuilder applications into IWS, and the *Users Guide for the PanelBuilder Import Wizard*.

- ✓ In the *Browser* window, double-click the *Installation* folder and then double-click the **InduSoft Web Studio** icon to start the *InduSoft Web Studio Installation Wizard*.
- ✓ A *Setup* dialog displays to inform you that the *Wizard* is loading.
- ✓ Follow the instructions provided by the *Wizard* to proceed with the installation, which includes:
 - Reading and accepting the License Agreement
 - Entering a user name and your company name
 - Choosing a destination location (*accept the default*)
 - Selecting the components to install (*accept the default*)

A *Setup Status* dialog displays while the program installs, and the *Setup Complete* dialog displays when the installation is finished:



Setup Complete Dialog

- ✓ You must restart your computer to continue, so click the **Yes, I want to restart my computer now** radio button, and then click **Finish**.
- ✓ After restarting your computer, double-click the **InduSoft Web Studio** icon on the desktop or select **Start** → **Programs** → **InduSoft Web Studio** → **InduSoft Web Studio** to start the IWS program.



Starting InduSoft Web Studio

⇒ **Tip:**

You can run the development environment under any video setting; however, to run applications on a CE platform, we recommend configuring your *Display* video settings to a resolution of 800x600 (or higher) and using 256 colors (or

higher) for a more pleasing environment. Application resolution (screen size) is independent of the operating system resolution.

 **Note:**

Microsoft *.NET Framework 1.1* is automatically installed, starting with IWS v.6 Service Pack 3. See Using ODBC Databases, in *Chapter 17: IWS Database Interface*, for more information.

Installing CEView

 **Note:**

The runtime system software (CEView) for the Windows CE operating system is usually 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.

When you install InduSoft Web Studio on Windows 2000/XP/Vista, IWS stores the CEView runtime files in the following folder:

<InduSoft Web Studio Folder>\Redist\<OS Version>\<Processor Type>

Where:

- **<InduSoft Web Studio Folder>** is the installation directory chosen during the installation (**C:\Program Files\InduSoft Web Studio v6.1** is the default installation directory).
- **<OS Version>** is the operating system version where CEView will be installed. The *CEView* subfolder stores the files for WinCE v3.0. The *WinCE 4.0* folder stores the files for WinCE v4.0. The *WinCE 4.1* folder stores the files for WinCE v4.1 and so on.
- **<Processor Type>** is the processor platform. InduSoft provides a CEView runtime for most processor platforms supported by the WinCE operating system.

To install CEView, use the following steps:

- Power-on the WinCE device, and the *Remote Agent* dialog should launch automatically.

If the dialog does not display, copy the **CEServer.exe** file from the **\InduSoft Web Studio v6.1\Redist\CEView\<Processor Type>\BIN** directory on the Win2K/XP/Vista computer where you installed IWS, paste the file into the **\<non-volatile>** folder of your WinCE device, and run the file.

 **Note:**

There are different ways to copy a file into a WinCE device (for example, you can map a shared folder from the Win2K/XP/Vista computer in the WinCE device or you can use ActiveSync). If you need assistance copying this file into the WinCE device, contact InduSoft technical support.

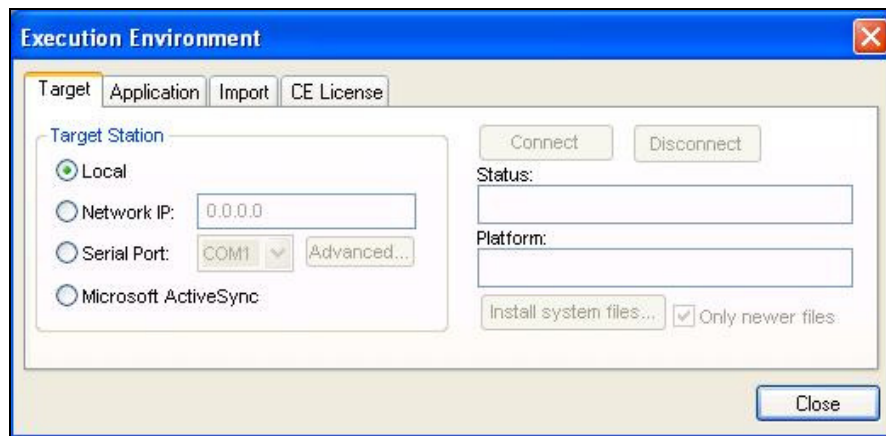
- After executing the **CEServer.exe** file, the *Remote Agent* dialog launches in the WinCE device.

- ✓ Click the **Setup** button in the *Remote Agent* dialog and configure the link (Serial or TCP/IP) to connect the WinCE device to the Win2K/XP/Vista computer.

Note:

InduSoft recommends using the TCP/IP link between the WinCE device and your Win2K/XP/Vista computer to download and upload files.

- ✓ Start *InduSoft Web Studio* on the Win2K/XP/Vista computer.
- ✓ Select **Project** → **Execution Environment** from the main menu bar.
- ✓ When the *Execution Environment* dialog displays, select a connection type (**Network IP**, **Serial Port** or **Microsoft ActiveSync**) and configure its settings (for example, IP Address or COM Port).



Execution Environment Dialog

- ✓ Click the **Connect** button to connect InduSoft Web Studio to the WinCE device.
- ✓ Click the **Install System Files** button from the *Execution Environment* dialog (**Target** tab) to download the CEView files to the WinCE device.

Uninstalling InduSoft Web Studio and CEView

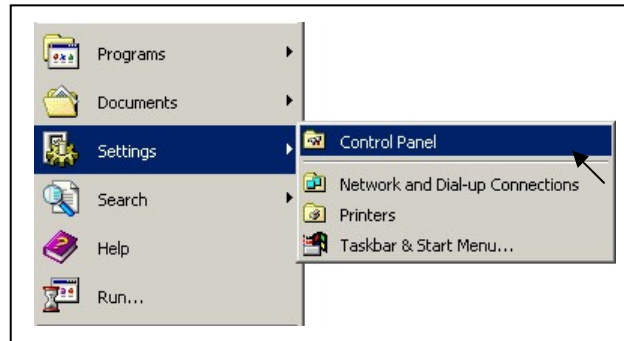
If you find it necessary to remove InduSoft Web Studio from your system, follow these instructions:

Caution:

Before starting the uninstall procedure, be sure to back-up any files you may find useful later into the ...**InduSoft Web Studio v6.1**\ folder.

Also, be certain that you have a current (or newer) version of the *InduSoft Web Studio* installation CD-ROM or diskettes so you can re-install the software later if necessary.

- ✓ From the Windows task bar, select **Start > Settings > Control Panel** to open the *Control Panel*.

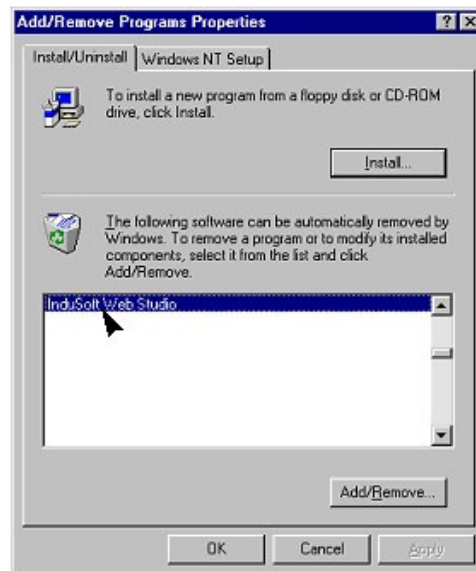


Opening the Control Panel

- ✓ Double-click on the **Add/Remove Programs** icon in the *Control Panel* window:



- ✓ When the *Add/Remove Programs Properties* dialog displays, select **InduSoft Web Studio** from the list and click the **Add/Remove** button.



Removing InduSoft Web Studio

- ✓ When the *Confirm File Deletion* dialog displays, click the **Yes** button.

The *Uninstall Shield Wizard* and the *Remove Programs from Your Computer* dialogs display.



Removing Programs: Progress Screen

- ✓ When the *Uninstall successfully completed* message displays, and the **OK** button becomes active, click the **OK** button.
- ✓ Verify that InduSoft Web Studio is no longer listed in the *Add/Remove Programs Properties* dialog.
- ✓ Click the **Close** button (**X**), to close the *Add/Remove Programs Properties* dialog, and then close the *Control Panel* window.
- ✓ Open the *Windows Explorer* and browse to the directory containing the InduSoft Web Studio directory.
- ✓ Verify that all of the InduSoft Web Studio files and folders were deleted. (You must manually delete any that remain.)

Note:

The uninstall tool cannot automatically delete files you created or modified in the `\InduSoft Web Studio v6.1\Projects\` or in the `C:\Documents and Settings\\My Documents\InduSoft Web Studio v6.1 Projects` folder.

Licensing Your Product

This section explains how to license your *InduSoft Web Studio (IWS)* and *CEView* products.

Notes:

- These instructions are valid for InduSoft Web Studio v5.1 or later.
- CEView runtime files for each platform are stored in the development station during IWS installation. You can use the IWS remote management tools to download *CEView* runtime files to the WinCE device.

Product Versions

IWS and CEView should both have the same version number, which uses the following syntax:

X.Y+SPWW (for example, InduSoft Web Studio v6.1+SP5 and CEView v6.1+SP5)

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 productEach 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.

⚠ Caution:

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

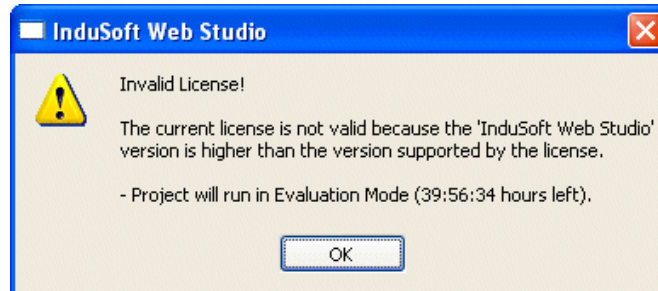
For example, you cannot execute version 6.1 applications using IWS version 6.0 but you can execute version 6.0 applications with IWS version 6.1.

⚠ IMPORTANT!

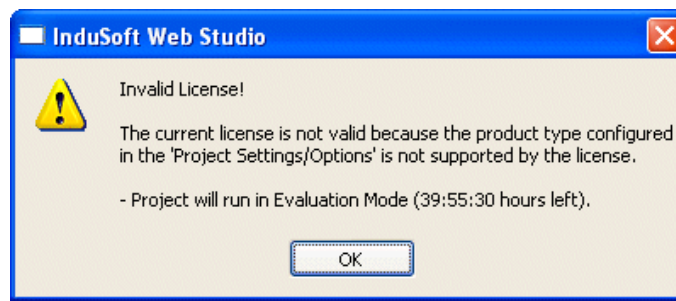
We issue each license for a specific *Family* version and *Sub-version* (X.Y), and the license is valid for that version (including Service Packs) only. However, a license is not valid for a newer *Family* version or *Sub-version* of the product. Any time you install a new version of IWS or CEView, you must upgrade the current license to the new version being installed. If you are installing a Service Pack only, you are not required to upgrade your license.

Invalid Licenses

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



Version of software is higher than the license version



Selected Target System is not supported by the license

Execution Modes

InduSoft Web Studio and CEView support the following execution modes:

Execution Mode	InduSoft Web Studio	CEView
Evaluation Mode	✓	✗
Demo Mode	✓	✓
Licensed for Engineering Only	✓	✗
Licensed for Runtime Only	✓	✓
Licensed for Engineering + Runtime	✓	✗

✓ = Apply; ✗ = Does NOT Apply

- Evaluation Mode:** Enables all of the product's engineering and runtime features. The first time you install IWS on a computer, the product runs for 40 (forty) 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 the product has an evaluation period that is independent of every other InduSoft Web Studio 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:** Enables you to download and upload applications to remote stations, and to run applications for testing and/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 application settings in **Demo Mode**.

The following menu options are available in **Demo Mode**:

File	View	Project		Tools	Help
Open Project Exit	Toolbars Status Bar Zoom Library	Settings Status Run Application Stop Application	Send Project to Target Execution Environment Logon	Register Controls Convert Resolution Verify Application System Information	All Options

- **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.

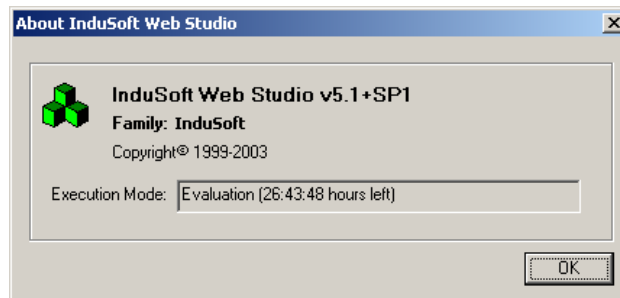
IMPORTANT!

Every license sets restrictions, such as which *Product Types* are supported for that license. Consult your software vendor about which product types are available and which features are enabled for each type.

Notes:

The IWS Execution Environment tools are always available so you can upload or download files from or to remote stations (such as remote WinCE devices) using the *Execution Environment* dialog (**Project > Execution Environment**).

You can select **Help** → **About** from the main menu bar to see which execution mode you are running. The About InduSoft Web Studio screen contains information about the current execution mode. If you are running *Evaluation* mode, the amount of time you have remaining displays in the **Execution Mode** field.



Verifying Version, Execution Mode, and Remaining Evaluation Time

Protection Types

InduSoft Web Studio and CEView support the following protection types:

Protection Type	InduSoft Web Studio	CEView
Hardkey	✓	✗
Softkey	✓	✓

✓ = Apply; ✗ = Does NOT Apply

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

The IWS license resides in the hardkey, and you cannot share this license simultaneously with more than one other copy of IWS in the network. If you connect the hardkey to another computer, you will be transferring the license to that computer.

Using the hardkey does not prevent you from connecting another device (such as a printer) to the computer's parallel 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 IWS 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 IWS or CEView license on your computer or WinCE device.

 **Note:**

When you use a softkey, IWS records the license in the computer's (or WinCE 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:

- **Product Type:** Specifies which features and restrictions are enabled for the application (such as maximum number of tags supported, maximum number of drivers running simultaneously, and so on). Consult your software vendor about which product types are available and which features are enabled for each type.
- **Execution Mode:** Specifies the following options
 - **Engineering Only:** Configures and runs the application for testing during development only. You cannot use this license as a long-term, runtime license.
 - **Runtime Only:** Runs the application for unlimited time. You cannot use this license to develop or modify the application.
 - **Engineering + Runtime:** Configures, modifies, and runs the application for an unlimited time.
- **Number of Web Thin Clients:** Specifies how many Web Thin Clients are supported simultaneously by the server. You can connect one or more Web Thin Clients to the server simultaneously (for an additional charge), but the license installed on the server must support these additional Web Thin Clients.

Options for adding Web Thin Clients are as follows:

- **Server for InduSoft Web Studio (Win2K/XP/Vista):** Connects 0 (*No Web Thin Clients*), 1, 2, 4, 8, 16, 32, 64, 128, or 256 Web Thin Clients to the server simultaneously.
- **Server for CEView (WinCE):** Connects 0 (*No Web Thin Clients*), 1, 2, 4, or 8 Web Thin Clients to the server simultaneously.
- **Version Supported:** When you generate a license, the license will specify the product version it supports.

Installing Hardkey Licenses for InduSoft Web Studio

This section explains how to install and upgrade a hardkey license on a Windows 2K/XP/Vista computer.

Installing a New License

To install a new IWS license, use the following steps:

- ✓ Install InduSoft Web Studio on your computer using the instructions provided earlier in this chapter.
- ✓ Connect the hardkey to the parallel port (LPT1) or USB interface on the computer where you installed IWS.


⚠ Caution:

Be careful when you install or remove a hardkey from your computer. We strongly recommend that you turn off the computer and disconnect it from the power supply before installing or removing the hardkey.

- ✓ Run InduSoft Web Studio.

Upgrading the Current Hardkey License

To upgrade your current IWS license, perform the following steps:

- ✓ Close all InduSoft Web Studio development and runtime modules.
- ✓ Ensure the hardkey is connected to the parallel port (LPT1) or USB interface on the computer where you installed IWS.
- ✓ Select **Start > Programs > InduSoft Web Studio >  Register to execute the Register module.**
- ✓ When the *Protection Manager* dialog displays, enable the **Hardkey** button in the **Protection Type** section and then click the **Check** button.



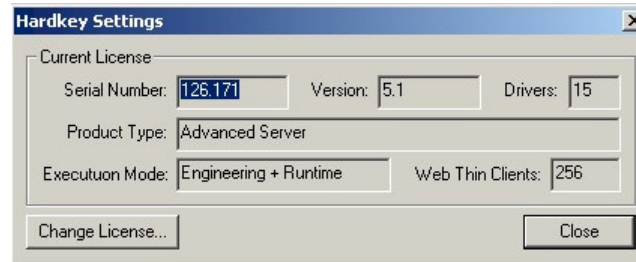
Protection Manager: Select Hardkey

- 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

- ✓ Click the **Change License** button to open the *Change License – Hardkey* dialog:



Change License Dialog

- ✓ Copy the code from the **Site Code** text box and send it to your software vendor.
- ✓ 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 (*Product Type*, *Execution Mode*, or *Number of Web 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 SoftKey License for InduSoft Web Studio


This section explains how to install and upgrade a SoftKey license for IWS (on Win 2K/XP/Vista).

 **Note:**

You must have Administrator privileges for the Windows 2K/XP/Vista workstation on which you are installing or modifying a softkey license.

Installing a New License

To install a new IWS softkey license, use the following steps:

- ✓ Install InduSoft Web Studio on your computer using instructions provided earlier in this chapter.
- ✓ Execute the *Register* module by selecting **Start** → **Programs** → **InduSoft Web Studio** →  **Register**.

Click (*enable*) the **Softkey** radio button in the *Protection Type* group, and then click the **Check** button.



Protection Manager: Softkey

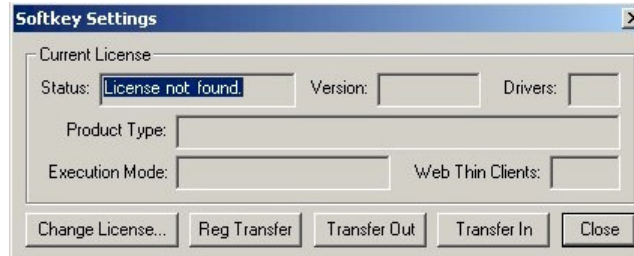
 **Note:**

If you have a hardkey license installed on your computer, the *Register* dialog displays a **Warning: this will change the protection method of the software. Continue anyway?** message.

To continue with the softkey installation, click **Yes**.

The *Softkey Settings* dialog displays.

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



Checking the Softkey Settings

- ✓ Click the **Change License** button on the *Softkey Settings* dialog.
- ✓ 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.

- ✓ Close the *Register* module and run *InduSoft Web Studio*.

Upgrading the Current License

To upgrade your current IWS license, use the following steps:

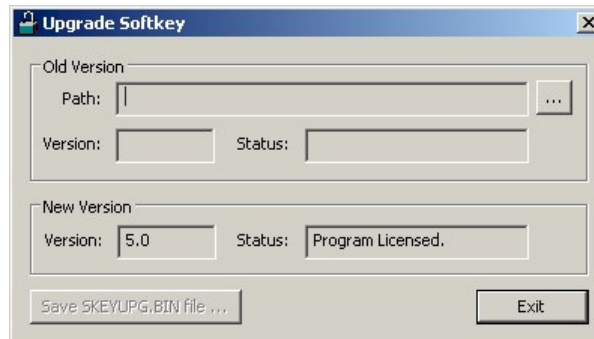
- ☑ Before upgrading a softkey license, you must re-install InduSoft Web Studio on the same computer where you installed the previous license.

Caution:


Do not uninstall InduSoft Web Studio before getting an upgraded license from your software vendor or you will lose your current license and it will not be possible to upgrade. (You will have to order a new license.)

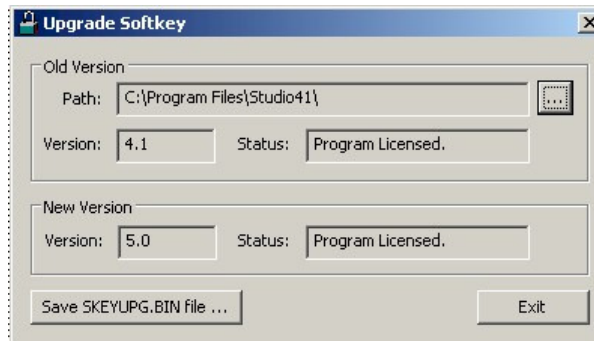
- ☑ From the newly installed version of InduSoft Web Studio, open the \BIN subfolder and run the **SKEYUPG.exe** program.

The *Upgrade Softkey* dialog displays as shown:




Upgrade Softkey Dialog

- ☑ In the *Upgrade Softkey* dialog, click the  button and when the *Browse* dialog displays, select the path (location) where the previous (already licensed) version of IWS was installed. The specified path information automatically displays in the **Path** text box in the **Old Version** section.



Finding the Previously Installed Softkey

- ☑ Click the **Save SKEYUPG.BIN file** button to save the information necessary to generate the upgraded license.
- ☑ Copy the **SKEYUPG.BIN** file from the directory where you executed the **SKEYUPG.exe** program and send it to your software vendor.
- ☑ Your software vendor will send you a *Site Key* matching the information saved in the **SKEYUPG.BIN** file.

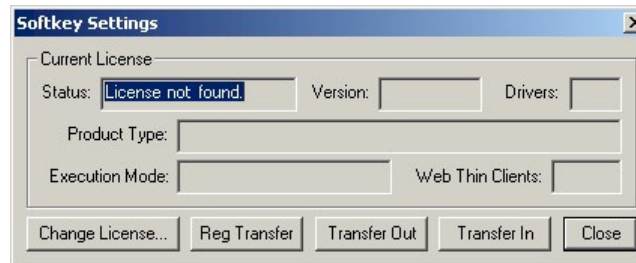
- ✓ Execute the *Register* module ( **Register**) for the newly installed version of InduSoft Web Studio (where you executed the **SKEYUPG.exe** program to generate the **SKEYUPG.BIN** file).
- ✓ When the *Protection Manager* dialog displays, click (*enable*) the **Softkey** button in the **Protection Type** section and click the **Check** button.



Protection Manger: Select Softkey

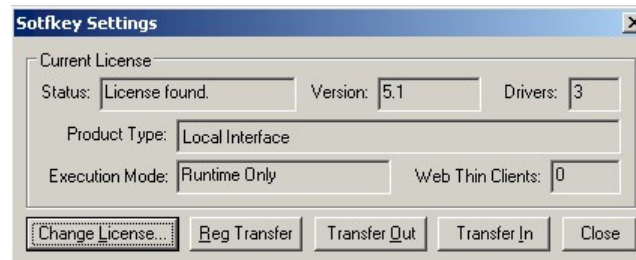
The *Softkey Settings* dialog displays.

- If you have not previously installed a license on your computer, the **Status** text box displays a “**License not found**” message:



Softkey Settings Dialog

- If you already have a valid InduSoft Web Studio Softkey license installed, the current license settings display:



Softkey Settings: License Found

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

- ✓ In the *Softkey Settings* dialog, click the **Change License** button to open the *Change License – Softkey* dialog:



Change License: Softkey

- ✓ Type the site key sent by your software vendor into the **Site Key** field of the *Change License – Softkey* dialog.
- ✓ 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 valid, an error message displays. If this happens, verify that you entered the site key correctly. If you typed the site key correctly and still receive an error message, contact your software vendor for further assistance.

- ✓ Close the *Register* module and run *InduSoft Web Studio*.

Installing a SoftKey License for CEView

There are two ways to register a CEView license on your WinCE device:

- **Locally:** Using the Remote Agent from the WinCE device as the interface.
- **Remotely:** Using InduSoft Web Studio to send the license to the WinCE device.

 **Note:**

You can purchase some WinCE devices with the CEView license already loaded. Consult your software vendor about this possibility.

Installing or Upgrading a License (Locally)

To install a new (or upgrade an existing) CEView softkey license (locally), use the following procedure:

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

 **Note:**

The **\<Non-Volatile Folder>** path can vary with each WinCE device manufacturer.

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

```
<InduSoft Web Studio Path>\Redist\<WinCE version>\<Platform>\BIN\  
CEServer.exe
```

Where:

- **<InduSoft Web Studio Path>** is the directory where you installed IWS (for example, **C:\Program Files\InduSoft Web Studio v6.1**).
- **<WinCE version>** indicates the operating system version (for example, CEView for Windows CE v3.0, WinCE 4.0 for Windows CE v4.0, WinCE 4.1 for Windows CE v4.1, and so forth).
- **<Platform>** is the WinCE device processor type (for example, **x86**).

 **Note:**

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

⇒ **Tips:**

- There are two ways to download the **Remote Agent** program (**CEServer.exe**) to a WinCE device:
 - You can use the Microsoft *ActiveSync*[®] utility to download/upload files from a Win2K/XP/Vista station to a WinCE 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 WinCE devices:


```
net use [<Local Name>] [Remote Name] [/user:<UserName>]
```

After executing this command successfully, open a *Command Prompt* window and use a **COPY** command to copy files to the WinCE device.
- We *strongly* recommend that you configure the WinCE device to execute the **Remote Agent** program automatically when you power on the WinCE device. See the WinCE 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 WinCE device, you can run it manually from the **\<Non-Volatile Folder>**.



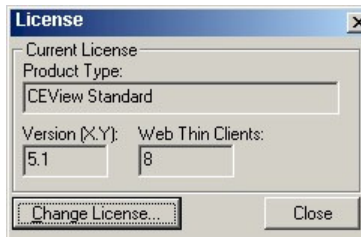
Remote Agent Dialog

- ☑ 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** text box) 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:



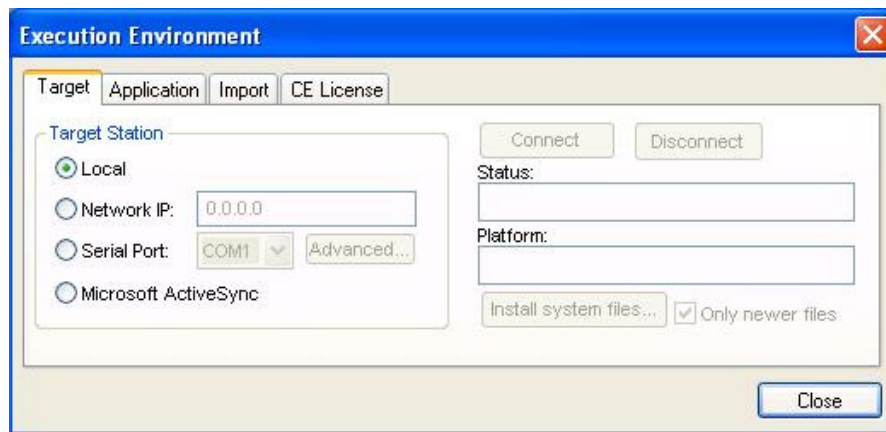
Successful Site Key Installation

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 License (Remotely)

To install a new (or upgrade an existing) *CEView* SoftKey license (remotely), use the following procedure:

- ✓ Execute the three first steps described in the previous section.
- ✓ In the *Setup* dialog, specify the **Device Connection** type by clicking (*enabling*) the **Serial Port** or **TCP/IP** button. (If you enable **Serial Port**, you also must select a port from the combo-box list). Click **OK** to close the dialog.
- ✓ Run InduSoft Web Studio on the Win2K/XP/Vista station and connect this station to the WinCE device using either a serial or TCP/IP link.
- ✓ From the IWS main menu bar, select **Project > Execution Environment** to open the *Execution Environment* dialog:

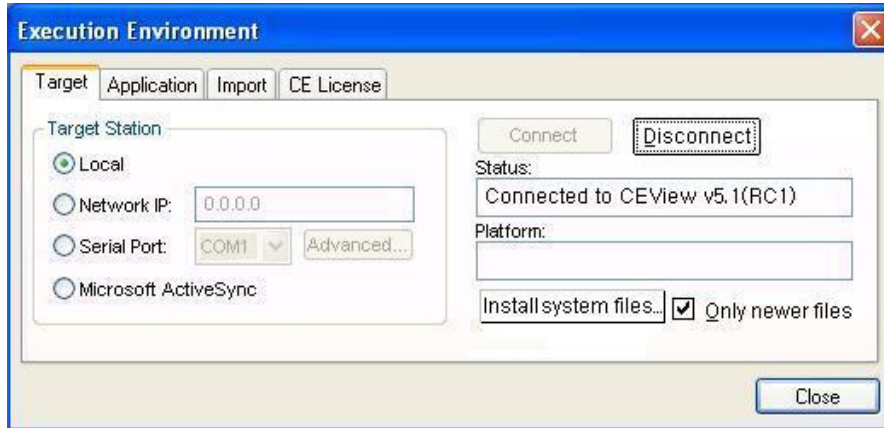


Execution Environment Dialog

- ✓ Specify a target station by clicking one of the following radio buttons in the *Target Station*:
 - **Local**
 - **Network IP** and type the IP address into the field provided
 - **Serial Port** and select a port from the combo-box list provided
 - **Microsoft ActiveSync**
- ✓ When the **Connect** button becomes active, click the button to connect to the WinCE device on which the **Remote Agent** is running. (If you select **Network IP**, you must also enter the IP address in the text box provided.)

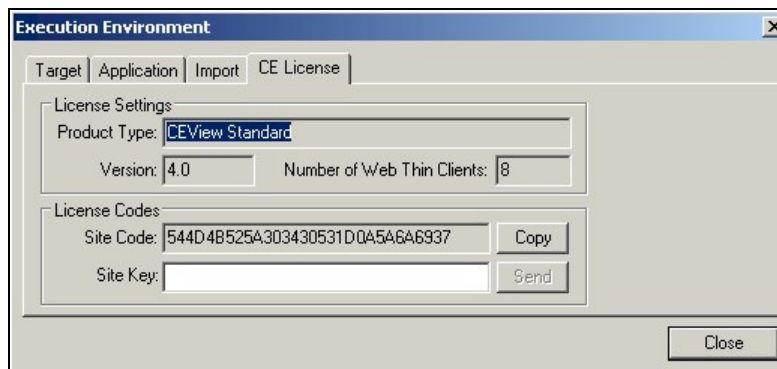
⇒ **Tip:**
TCP/IP links provide better communication performance than serial links.

The **Status** field must display the following message:
Connected to CEView <CEView Version>



Connecting to CEView

- ☑ Select the **CE License** tab to see which license settings are currently installed on your WinCE device.



CEView License Settings

- ☑ From the **License Codes** section of the *Execution Environment* dialog, 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 matches 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 WinCE device.

The **Remote Agent** program will attempt to install the new license using the site key sent from InduSoft Web Studio. If the site key is accepted (validated), the following message displays:



Successful Site Key Installation

**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 WinCE device, be sure to save its registry settings. If you do not save these settings, you will lose the license after rebooting the device.

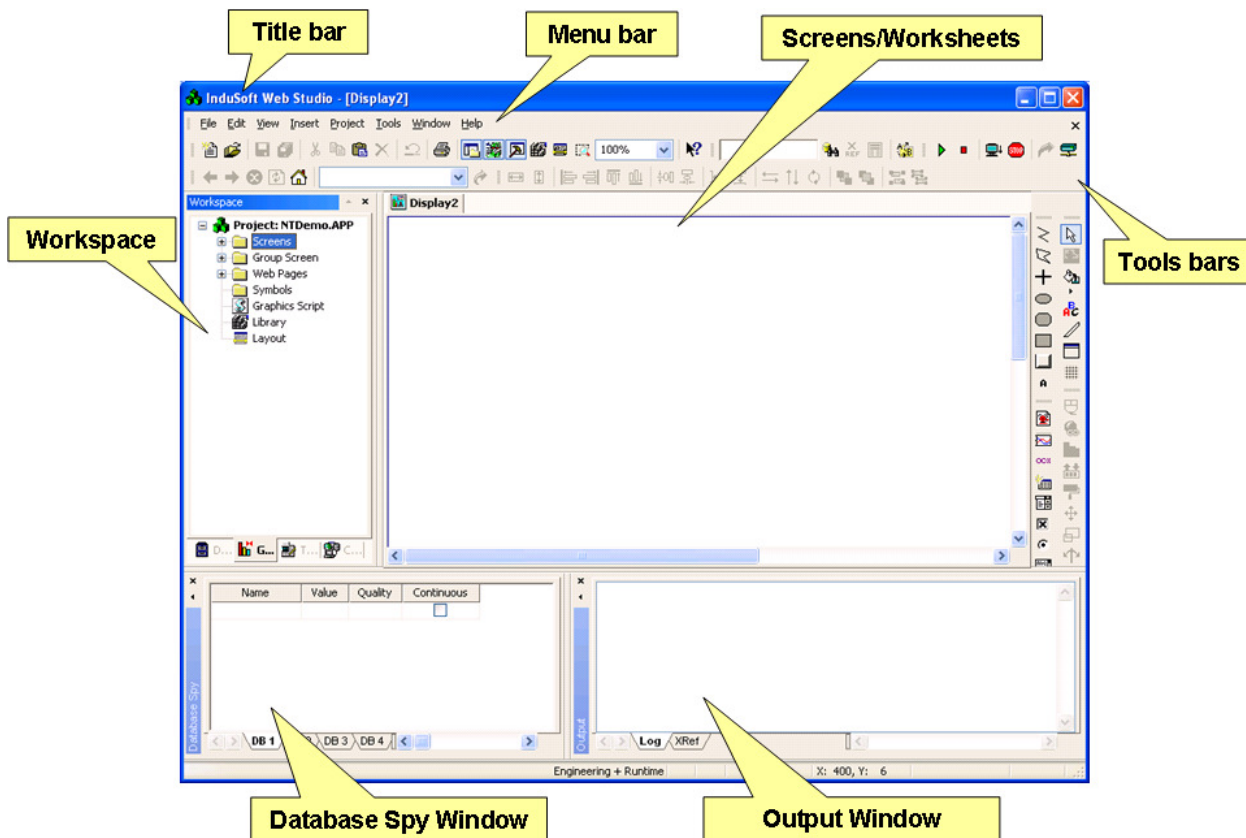
Chapter 3: Working with the IWS Development Environment

This chapter provides an overview of the InduSoft Web Studio development environment. This information is organized into the following sections:

- Navigating the Development Environment Window
- Using the Screen/Worksheet Editor
- Using the Database Spy
- Using the Output Window
- Customizing the Workspace
- Development Modules
- General Communications

Navigating the Development Environment Window

InduSoft Web Studio uses standard, Windows-like tools and interfaces to provide an integrated and user-friendly *development environment*:



This development environment consists of the following features:

- Title bar
- Menu bar
- Toolbars
- Workspace
- Screen/Worksheet Editor
- Database Spy window
- Output window (LogWin)
- Status bar

 **Note:**

Other IWS tools, such as the *Symbol Library* and the *Translation Editor* also contain some of these features (such as menu bars).




Using the Title Bar

The title bar is located along the top of the development environment window and it contains the InduSoft icon, the product name, and the name of the active screen or worksheet (if any).



Example Title Bar

The title bar also contains the following buttons (from left to right):

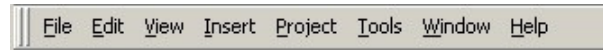
- **Minimize** button (): Click to minimize the development environment window.
- **Restore Down/Maximize** button (): Click to toggle the development environment window between two sizes:
 - **Restore Down** reduces the window to its original (*default*) size
 - **Maximize** enlarges the window to fill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

 **Note:**

Closing the development system does not close the runtime system.

Using the Menu Bar

The *menu bar* is located just below the title bar.



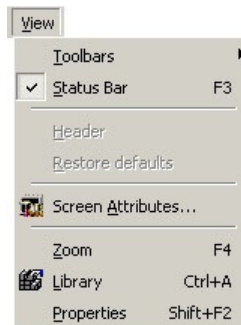
Menu Bar

The menu bar provides access to the following menus and their menu options:

- **File:** Contains options that enable you to manage application files.
- **Edit:** Contains options that enable you to manage screens and worksheets.
- **View:** Contains options that enable you to manage visible tools and provides shortcuts to dialogs you open most frequently.
- **Insert:** Contains options that enable you to configure application tags, tag classes, documents, drivers, users, security settings, screens, and ActiveX objects.
- **Project:** Contains options that enable you to execute applications locally and remotely, and provides links used to configure general application settings.
- **Tools:** Contains options that provide links to auxiliary development tools.
- **Window:** Contains options that enable you to manage open screens and worksheets.
- **Help:** Contains options that provide links to information about the InduSoft Web Studio product and InduSoft.


When you click on the menu name, a pulldown menu displays containing a variety of related options.

For example, when you click **View**, the following pulldown menu displays:



View Pulldown Menu

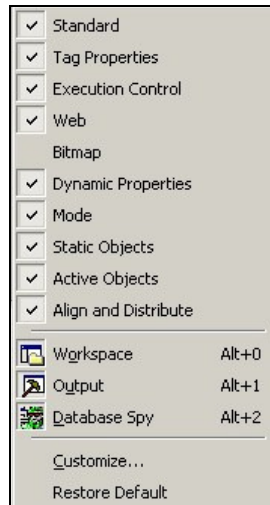
Notes:

- Individual menu options are discussed in detail throughout this publication whenever their use relates to performing a specific task.
- In many cases, menu option functionality is duplicated using buttons on the various toolbars. For example, you can click the  button or select **File** → **Print** to open the *Print* dialog.

PULLDOWN MENU FEATURES

Certain features on a pulldown menu provide a clue as to what you can expect when you select a menu item.

- **Ellipses** (): Indicates a dialog or window will open when you select this option. For example, when you select the **New** option, the *New* dialog displays.
- **Arrows** (): Indicates a cascade menu (a second pulldown menu) will display when you select this option. For example, when you select the **Toolbars** option, the following cascade menu displays:



Toolbars Cascade Menu

- **Keyboard Combinations and Function Keys:** Indicates an alternate method (shortcut) for selecting that option from the pulldown menu. For example,
 - Pressing the **Ctrl** and **p** keys simultaneously is the same as selecting the **Print** option from the **File** menu.
 - Pressing the **Alt** and **1** keys simultaneously opens/closes the *Output* window.
 - Pressing the **F4** function key opens the *Zoom* window.

- **Toggle buttons** (or): Click the button to alternate between turning the feature on or off .

For example, if you click once on the **Standard** button shown in the figure above, you will turn off the *Standard* toolbar so it will not display in the development environment. Click the button again, and the toolbar will display. Similarly,

clicking the **Workspace** button), toggles the *Workspace* display on and off .

- **Option Name only:** Indicates that when you click the option name, IWS will perform the task immediately or put you in the right mode to perform the task. For example, if you select **File** → **Save**, IWS immediately saves the active application screen.

DOCKING THE MENU BAR

The menu bar is a *dockable* feature, which means you can move it to another location in the development environment window.

To move and dock the menu bar:

- ☑ Click on the menu bar and drag it to a new location.
- ☑ Release the mouse button to attach or *dock* the menu bar to its new position.

Using the Toolbars

InduSoft Web Studio provides several toolbars that enable you to perform different tasks within the program. This section describes the function and default location of each toolbar.

- The following toolbars contain general purpose tools, and they are located across the top of the *Workspace*, just below the menu bar by default:
 - *Standard*
 - *Tag Properties*
 - *Execution Control*
 - *Web*
 - *Align and Distribute*
- The following toolbars contain screen editing tools, and they are located along the right side of the development environment window by default:
 - *Mode*
 - *Static Objects*
 - *Active Objects*
 - *Dynamic Properties*
 - *Bitmap*

Notes:

- The *Bitmap* toolbar is hidden by default. Select **View** → **Toolbars** → **Bitmap** from the main menu bar to display the *Bitmap* toolbar.
- Detailed instructions for using these toolbars and toolbar options are provided throughout this publication when you will use them to perform a specific task.

DOCKING A TOOLBAR

All toolbars are *dockable*, which means you can move them to another location in the development environment window. To move and dock any of the toolbars:

- ☑ Click on the toolbar and drag it to a new location.
- ☑ Release the mouse button to attach or *dock* the toolbar to its new position.

Tip:

Position your cursor on a button and the status bar (located at the bottom of the IWS interface) will provide a brief description of that button.

Using the Workspace

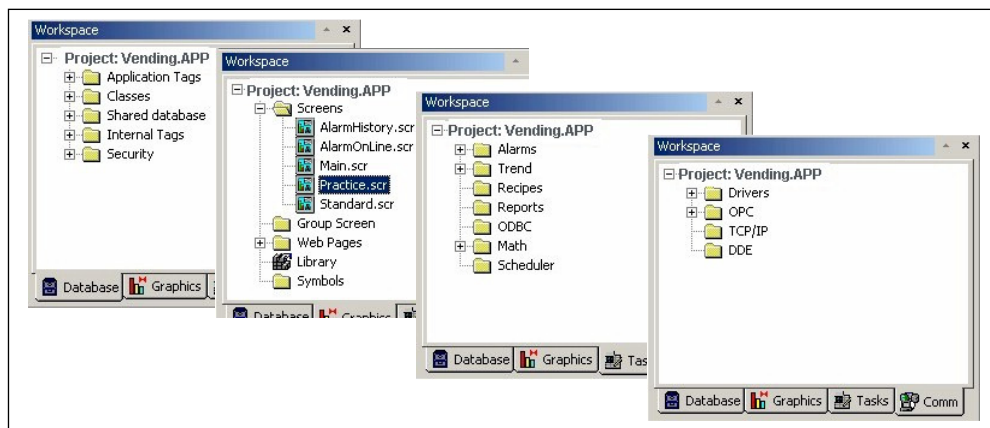
The *Workspace* window is a user-friendly interface that enables you to find any application component (such as tags, screens, worksheets, and so forth) quickly and easily.

You can resize, move, or hide the *Workspace* window to suit your development style.



The *Workspace* contains four tabs:

- **Database:** Provides access to all database tags and security-system components configured for the current application.
- **Graphics:** Provides access to all screens and symbols in the application.
- **Tasks:** Provides access to all task worksheets in the application.
- **Comm:** Provides access to all drivers configured to establish communication with another device or software program using available protocols

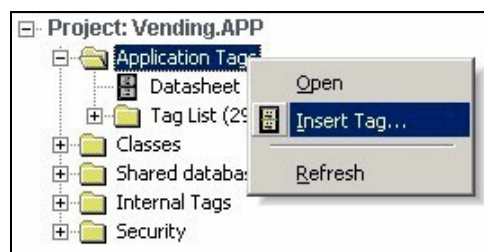
On each tab, the application components (represented by an icon and unique description) are organized into a *tree-view* display similar to the following:



Workspace Tabs

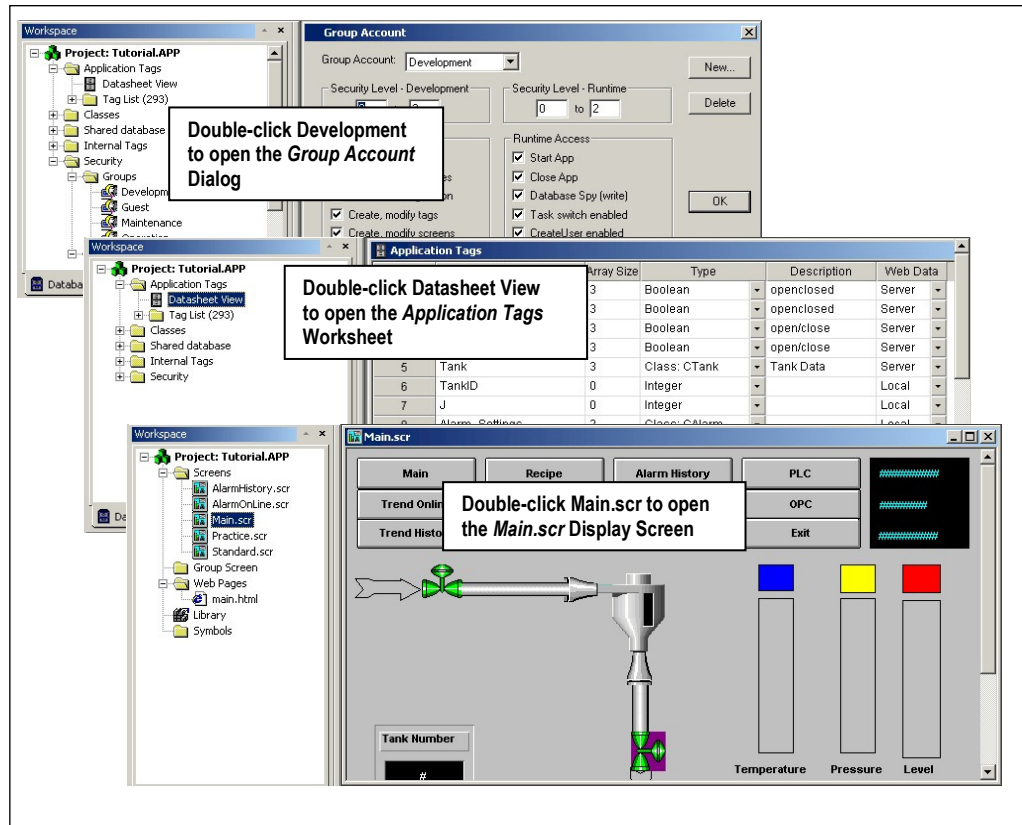
Click the  button or double-click the folder to view the contents of (*expand*) the folder or click the  button to close the folder.

If you right-click on any component in the *Workspace*, a menu displays with options related to that component. For example, the following pop-up menu enables you to **Open** the *Application Tags* database, **Insert** (create) a new tag, or **Refresh** the current view of the *Application Tags* database:



Right-Click to Open a Pop-Up Menu

If you double-click on the button or name of any component in a folder, a related IWS dialog, worksheet, or screen displays so you can edit that component. For example:



The rest of this section describes each of the *Workspace* tabs.

DATABASE TAB

Select the **Database** tab to access all of the database tags and security-system components that are available for the current application. This tab contains the following folders:

- **Application Tags** contains tags you created during application development (such as screen tags or tags that read from/write to field equipment).

⇒ **Tip:**

You can sort the data in the Application Tags sheet and/or insert/remove additional columns to/from the sheet by right-clicking on it and choosing the applicable option from the pop-up menu.

- **Classes** contain 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 software program and then imported into the IWS *Tags* database.

For example you can import *SteepleChase* tags into IWS so IWS can read/write data from a *SteepleChase* PC-based control product.

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

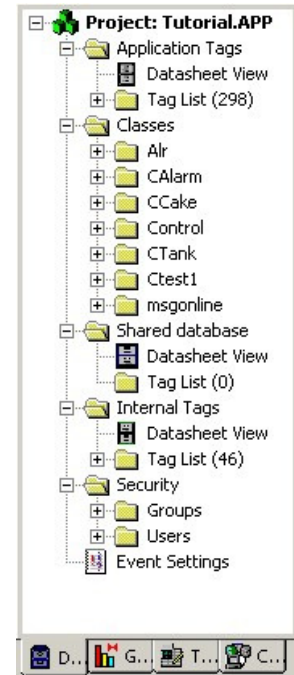
All internal tags are *read-only*, which means you cannot add, edit, or remove these tags from the database.

- **Security:** Contains all of the group and individual user security accounts configured for the current application.
- **Global Procedures:** Contains VBScript functions and sub-routines that can be called by any other script in the application.
- **Event Settings:** Contains logging and event-retrieval features.

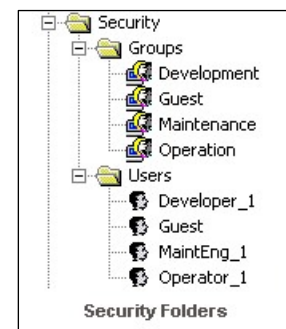
You can view, add, and edit tags in the *Application Tags*, *Classes*, *Shared Database*, or *System Tags* folders as follows:

- Double-click **Datasheet View** to open a *Tags* worksheet, which you use to create or modify tags for your application.
- Open a *Tag List* or *Member List* folder and double-click on any of the existing tag names to open a *Properties* dialog. You can use the parameters on this dialog to modify the tag's current properties.

To modify an existing *Security* account, open the *Groups* or *Users* folder and double-click on a group or user name. When a *Group Account/User Account* dialog displays, use the parameters on the dialog to change the existing account properties.



Tag Folders



Security Folders

Notes:

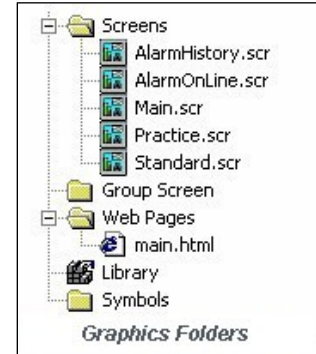
- *Chapter 5: Working with Tags* provides instructions for creating and editing tags in the IWS Tags database.
- *Chapter 11: Configuring a Security System* provides instructions for configuring a security system for your application.

GRAPHICS TAB

Select the **Graphics** tab to access all of the screens, Web pages, Library objects, and symbols in the application.

This tab contains the following folders:

- **Screens:** Contains all of the display screens created for the current application.
- **Group Screen:** Contains the entire screen groups (individual screens combined into manageable groups) created for the current application. (**Note:** Screen groups are not available for Windows CE.)
- **Web Pages:** Contains all the Web pages (screens saved in HTML format) created for the application.
- **Symbols:** Contains all of the user-defined symbols, which can be groups of images and/or text. You can create custom symbols for the application and save them into this folder.
- **Graphics Script:** Contains predefined functions that are executed when certain screen actions occur, such as when the Web Thin Client is launched on a remote station.
- **Library:** Contains the library of common symbols and graphics provided with InduSoft Web Studio. Double-click the **Library** button to open the IWS *Symbol Library* utility, consisting of a list pane (containing all of the symbol groups) and a display screen.
- **Layout:** Displays all screens currently open in the Screen Editor and allows you to visualize how the screens fit together during runtime.



To open a screen, Web page, or symbol for editing, double-click the appropriate button.

Note:

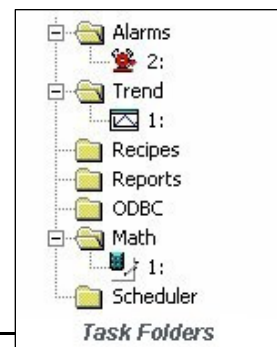
Specific instructions for editing screens, Web pages, or symbols are provided later in this chapter (“Using the Screen/Worksheet Editor” on page 3–11) and in *Chapter 7: Configuring Screens and Graphics*.

TASKS TAB

Select the **Tasks** tab to access all task worksheets in the current application.

This tab contains the following task folders:

- **Alarms:** Contains the *Alarm* worksheets used to configure alarm groups and the tags related to each alarm group in the application. You also use the Alarm task to define the alarm messages generated by IWS.
- **Trend:** Contains the *Trend* worksheets used to configure history groups that store trend curves for the application. 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. IWS



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 application 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 IWS and any database supporting the ODBC interface.
- **Math:** Contains the *Math* worksheets used to configure and implement additional routines to work with different IWS tasks. IWS 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 application.
- **Script:** Contains the Startup Script and other VBScript 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.
- **External Databases:** Contains the *DB* worksheets that communicate with external databases using the standard ADO.NET interface (as an alternative to ODBC).

To open *Task* worksheets for editing, double-click the task button and the worksheet will display in the *Screen/Display* window.

 **Note:**

Detailed instructions for editing worksheets are provided in *Chapter 8: Configuring Task Worksheets*.

COMMUNICATIONS TAB

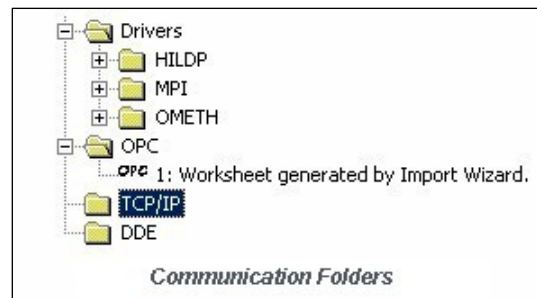
Select the **Communications** tab to access all of the drivers configured for your application. Drivers are used to establish communication with other devices or software programs using available protocols.

This tab contains the following folders:

- **Drivers:** Contains the *Driver* worksheets used to configure a communication interface (or interfaces) between the project application 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:** Contains the *OPC* worksheets used to configure OPC interfaces between the application and an OPC Server. An InduSoft Web Studio OPC Client module enables the IWS system to communicate with any device that implements an OPC



Server by implementing the OPC standard described in the *OLE for Process Control Data Access Standard Version 2.0* document published by the OPC Foundation.

- **TCP/IP:** Contains the *TCP/IP* worksheets used to configure TCP/IP Client interfaces for other InduSoft stations.

IWS TCP/IP Client and Server modules enable two or more applications to keep their databases synchronized using the TCP/IP protocol to provide communication between applications.

- **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 applications. IWS provides interfaces that run as clients or as servers.

 **Note:**

By default, the DDE Client module from IWS 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 <ApplicationName>.APP file (you can use Notepad to edit this file):

[Options]

DDEANSI=1

To open worksheets in the *Drivers*, *OPC*, *TCP/IP*, or *DDE* folders for editing, double-click the worksheet button and the worksheet will display in the *Screen/Worksheet* window.

 **Note:**

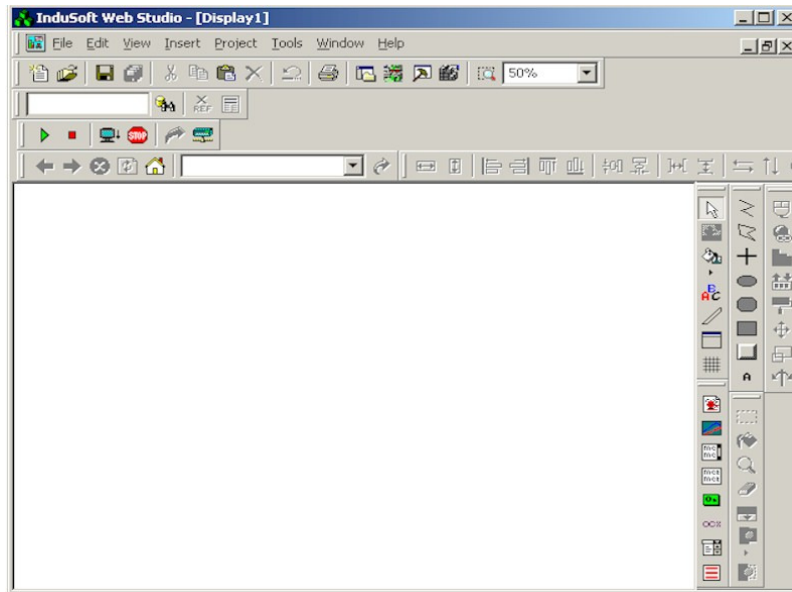
Detailed instructions for editing communication worksheets are provided in *Chapter 10: Communication*.

Using the Screen/Worksheet Editor

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

Other screen editor features include:

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




Screen/Workspace Editor

The following toolbars contain general purpose tools that enable you to perform different tasks within the program.


- **Standard toolbar:** Provides icons (shortcuts) that duplicate functionality found on the *File*, *Edit*, and *View* menus.




Standard Toolbar

- **New button** (): Click to open the *New* dialog and create new applications (projects) or files as part of your open application.

Note:
Using the **New** button is the same as selecting **File** → **New** from the menu bar or typing the **Ctrl+N** key combination.


- **Open Project button** (): Click to locate and open an InduSoft Web Studio application.

Note:
Using the **Open Project** icon is the same as selecting **File** → **Open Project** from the menu bar or opening the InduSoft Web Studio folder from *Windows Explorer* and double-clicking on the project name.

- **Save button** (): Click to save any active screens or worksheets.


**Notes:**

- Using the **Save** icon is the same as selecting **File** → **Save** from the menu bar or typing the **Ctrl+S** key combination.
- The **Save** function becomes available only when you modify the active file.

- **Save All** button (): Click to save all open screens or worksheets.


**Note:**

- Using the **Save All** icon is the same as selecting **File** → **Save All** from the menu bar.
- The **Save All** function becomes available only when you modify a screen or worksheet.

- **Cut** button (): Click to remove a selected object from the screen/worksheet and store it on the clipboard, replacing any previously stored selections on the clipboard. Use in combination with the **Paste** button.


**Note:**

Using the **Cut** icon is the same as selecting **Edit** → **Cut** from the menu bar or typing the **Ctrl + X** key combination.

- **Paste** button (): Click to place the contents of the clipboard into the upper left corner of the active screen. You can *Paste* a cut or copied object multiple times to a several screens/worksheets.


**Note:**

Using the **Paste** icon is the same as selecting **Edit** → **Paste** from the menu bar or typing the **Ctrl + V** combination.

- **Copy** button (): Click to duplicate a selected object and store it on the clipboard. You can then use the **Paste** button to move the copied object to another location on the same screen or to another screen.


**Note**


Using the **Copy** icon is the same as selecting **Edit** → **Copy** from the menu bar or typing the **Ctrl+C** key combination.


- **Delete** button (): Click to delete the selection. If you accidentally delete an object, you can restore it using the **Undo** button.


**Note:**


Using the **Delete** icon is the same as selecting **Edit** → **Delete** from the menu bar or typing the **Ctrl+X** key combination.


- **Undo** button (


 **Note:**
Using the **Undo** icon is the same as selecting **Edit** → **Undo** from the menu bar or typing the **Ctrl+Z** key combination.


- **Print** button (

 **Note:**
Using the **Print** icon is the same as selecting **File** → **Print** from the menu bar or typing the **Ctrl+P** key combination.


- **Workspace Window** button (


 **Note:**
Using the **Workspace Window** icon is the same as selecting **View** → **Toolbars** → **Workspace** from the menu bar or typing the **Alt+ 0** key combination.

- **Output Window** button (


 **Notes:**


 - Using the **Output Window** icon is the same as selecting **View** → **Toolbars** → **Output** from the menu bar or typing the **Alt+1** key combination.
 - See also “Using the Output Window” on page 3–24.

- **Database Spy Window** button (


 **Notes:**

 - Using the **Database Spy Window** icon is the same as selecting **View** → **Toolbars** → **Database Spy** from the menu bar or typing the **Alt+2** key combination.
 - See also “Using the Database Spy” on page 3–23.

- **Library** button (

 **Note:**

 - Using the **Library** icon is the same as selecting **View** → **Library** from the menu bar.
 - See also *Chapter 7: Configuring Screens and Graphics*.

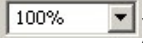

- **Layout** button (- * Modify the **Screen Attributes**: Right-click on the screen displayed on the *Layout* tab and use the alignment options or the **Screen Attributes** link to modify the screen position. You can also click and drag the screen to change its position (Top and Left) or resize it (Width and Height).
- * Visualize how the screens fit together during the 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.


- **Zoom** button (): Select the zoom scale from the combo-box.
- **Context Sensitive Help** button (

Tag Properties Toolbar

- **Tagname** text box (): Type a name into the text box to create a new tag for your application. The **Cross Reference** and **Tag Properties** buttons will reference this tag name for their actions.
- **Object Finder** button (): Click to open the *Object Finder* dialog, which lists all *Tags* and *Functions* currently configured for the application. See “Using the Object Finder” on page 3–47.
- **Cross Reference** button (): Click to search all application 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. See “Using the X-ref Option” on page 3–48.

- **Tag Properties** button (

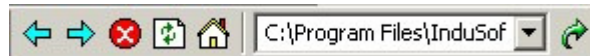
Execution Control Toolbar

- **Test Display** button (

Note:

For more information, see *Chapter 12: Testing and Debugging Your Application* and *Chapter 14: Managing Applications Remotely*.


- *Web* toolbar: Allows you to open and navigate HTML files.





Web Toolbar

Note:

You must install Internet Explorer v4.1 (or higher) before you can use any of the tools on the *Web* toolbar. See *Chapter 7: Configuring Screens and Graphics*, and *Chapter 13: Configuring a Web Solution* for more information.

- **Back** button (3–16


- **Forward** button (): Type a Web page URL address into the text box to open (download) that page to your *Internet Explorer* Web browser.
- **Go** button (- **Align and Distribute** toolbar: Allows you to edit screen objects. You can resize, align, flip, rotate, space, and group objects.




Align and Distribute Toolbar

 **Note:**

See “Using the Align and Distribute Toolbar” on page 7–10 in “Chapter 7: Configuring Screens and Graphics” for a more detailed description and examples of these buttons.

- **Resize height** button (InduSoft Web Studio v6.1 SSP4

- **Resize width** button (3–18

The following toolbars contain screen-editing tools. These toolbars are located along the right side of the interface window by default and they are enabled only while you are editing graphic screens:


- **Mode toolbar:** Allows you to edit your screens.



Mode Toolbar

Note:


See “Using the Mode Toolbar” on page 7–8 in *Chapter 7: Configuring Screens and Graphics* for a more detailed description.






- **Selection** button (

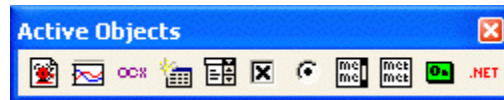
Static Objects Toolbar

Note:

See “Using the Static Objects Toolbar” on page 7–24 in *Chapter 7: Configuring Screens and Graphics* for a more detailed description.











- **Open Polygon** button (InduSoft Web Studio v6.1 SSP4


- **Ellipse** button (): Click to draw ellipses, chords, arcs, and rings.
 - **Rounded Rectangle** button (): Click to draw rounded rectangles (empty or filled).
 - **Rectangle** button (): Click to create rectangles.
 - **Button** button (): Click to create custom-sized buttons.
 - **Text** button (): Click to create text objects.
- **Active Objects** toolbar: Allows you to create dynamic objects. Active objects typically require more parameters than static objects and provide embedded dynamics.



Active Objects Toolbar

Note:
See “Using the Active Objects Toolbar” on page 7–44 in *Chapter 7: Configuring Screens and Graphics* for a more detailed description and examples.

- **Alarm/Event Control Object** button (): Click to add an Alarm/Event Control Object to your application screen.
- **Trend Control** button (): Click to display data points (values) from different data sources in a graphic format.
- **ActiveX Control** button (): Click to open the *Insert ActiveX Control* dialog, which you can use this dialog place ActiveX components on your screen.
- **Grid** button (): Click to specify an area of the screen to create a new **Grid** object.
- **Combo-Box** button (): Click to select a single label from a combo-box list of labels.
- **Check Box** button (): Click to create a check-box object on your screen.
- **Radio Button** (): Click to create a radio button object on your screen.
- **List Box Object** button (): Click to create a list box object on your screen. Generally, when you execute an application, the *active* list box object displays a list of messages.
- **Smart Message Objects** button (): Click to create one or more smart message objects, which you can use to display messages and graphics when you execute the application.
- **Pushbuttons** button (): Click to create a pushbutton object using the *Command* dynamic object property with an object or pre-configured pushbuttons.









- **.NET Control** button (): Click to open the *.NET Framework Components* dialog, which you can use to place *.NET Framework* components on your screen.
- *Dynamic Properties* toolbar: Allows you to apply dynamics to objects or a group of objects. Dynamics enable you to modify object properties on the fly (during runtime) according to tag values. Some dynamics also enable you to execute commands or insert values (set points) to the tags.



Dynamic Properties Toolbar

 **Note:**

See “Using the Dynamic Properties Toolbar” on page 7–31 in *Chapter 7: Configuring Screens and Graphics* for a more detailed description.






- **Command** button (): Click to add the command property to a selected object or group of objects. The command property enables you to click on the object or press a pre-defined key to execute the command at runtime.
- **Hyperlink** button (): Click to add the hyperlink property to a selected object or group of objects. Applying this property allows you to click on the object(s) during execution to launch the default browser and load the specified URL.
- **Bargraph** button (): Click to add bar graph properties to a selected object, and then double-click on the object to open the *Object Properties* dialog.
- **Text I/O** button (): Click to add the dynamic input or output text property to a selected text object. Applying the **Text I/O** property allows you to insert and display tag values in real time if you are using the keyboard or on-screen keypad to run an application.
- **Colors** button (): Click to add the color change property to a selected object. The **Colors** dynamic allows you to specify up to four **Change Limit** colors.
- **Position** button (): Click to specify when and where to display an object, using the specified tag values.
- **Resize** button (): Click to increase or decrease the size of a selected object or symbol.
- **Dynamic Rotation** button (): Click to rotate a line.
- *Bitmap* toolbar: Allows you to access the bitmap screen editor tools. (This toolbar is available only when the *Background Picture* layer is active. You enable the *Background Picture* layer in the *Screen Attributes* dialog.)



Bitmap Toolbar



 **Note:**

- The *Bitmap* toolbar is hidden by default.
- See “Using the Bitmap Toolbar” on page 7–22 in *Chapter 7: Configuring Screens and Graphics* for a more detailed description.

- **Select Area** button (): Click to select an area within the *Bitmap Screen Editor*.
- **Flood Fill** button (): Click the **Flood Fill** button, and then click on the screen to paint the surrounding area with the color you specified with the **Fill Color** button.
- **Pixel Editing** button (): Click to open an *Edit Image* dialog, where you can draw detailed bitmaps, pixel by pixel.
- **Erase Area** button (): Click to remove a selected area from the screen.
- **Change Colors** button (): Click to change the transparent fill color for a selected area.

 **Note:**

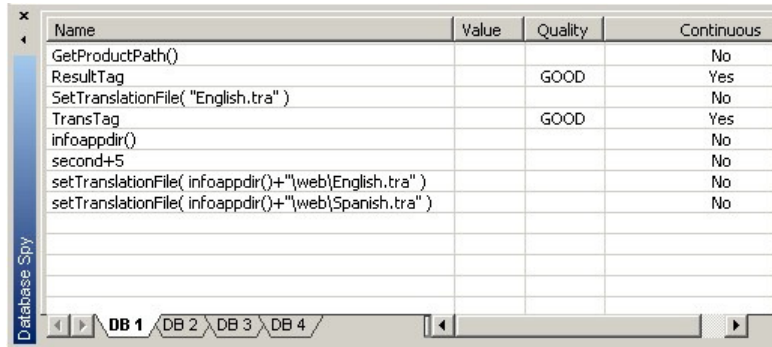
Before you can use this button, you should have already specified a fill color (**Fill Color** button), selected a transparent color (**Select Transparent Color** button), and defined the area to fill (**Select Area** button).

- **Select Transparent Color** button (): Click to specify a transparent color (referenced by the Change Colors button).
- **Toggle Transparent Color** button (): Click to cause the color selected using the **Select Transparent Color** button to become transparent for bitmaps selected in the *Bitmap Screen Editor*.

Using the Database Spy

The *Database Spy* window (located under the *Workspace* by *default*) is an IWS debugging tool that allows you to:


- Monitor and force values to database tags
- Execute and test functions
- Execute and test math expressions



Name	Value	Quality	Continuous
GetProductPath()			No
ResultTag		GOOD	Yes
SetTranslationFile("English.tra")			No
TransTag		GOOD	Yes
infoappdir()			No
second+5			No
setTranslationFile(infoappdir()+"\web\English.tra")			No
setTranslationFile(infoappdir()+"\web\Spanish.tra")			No

Sample Database Spy Window

The *Database Spy* window consists of the following elements:

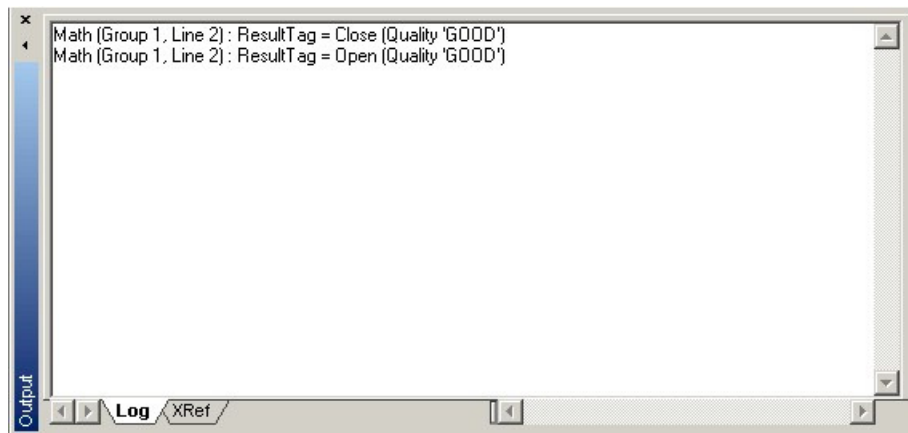
- **Hide Docked Window** button (✕): Click to toggle between opening and closing the window. (Alternatively, you can use the  **Database Spy** button on the **View → Toolbars** menu or **Alt+2** to open and close this window.)
- **Contract/Expand** button (▾): Click to toggle between contracting and expanding the *Database Spy* window.
- **DB tabs**: Click each tab to access a spreadsheet with the following components
 - **Name**: Displays tag names, function names, and equations
 - **Value**: Displays returned values and equation results
 - **Quality**: Displays a quality evaluation (**Good** or **Bad**) of the tag or function source
 - **Continuous**: Displays whether IWS is re-evaluating the tag, function, or equation continuously
- **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 spreadsheet.

Notes:

- 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.
- Detailed instructions for using the *Database Spy* are provided in *Chapter 12: Testing and Debugging Your Application*.

Using the Output Window

The *Output* window (located next to the *Database Spy* by default) is another IWS debugging tool that is similar in function to a *LogWin* module.




Sample Output Window

Though it functions on a smaller scale than the *LogWin*, the *Output* window enables you to monitor components in your application directly from the development environment. (For a description of the *LogWin* module, see *Chapter 12: Testing and Debugging Your Application*.)

The *Output* window contains the following elements:

- **XRef** tab: Use this tab to search for and display the location (path, file name, column, and row) of every instance where a specific tag is being used within your application. (See also “Using the X-ref Option” on page 3–48.)
- **Hide Docked Window** button (☒): Click to open or close the window.

Alternatively, you can select **View** → **Toolbars** from the menu bar and click the  **Output** button or press **Alt+1** to open/close the window.

- **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.

Notes:

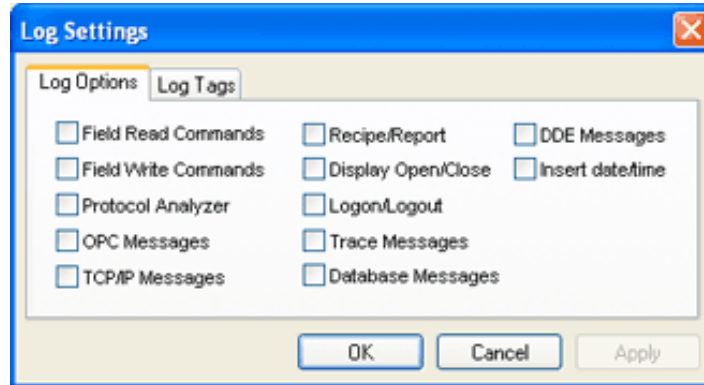
- The *Output* window is *dockable*, which means you can move it to another location in the development environment. Click on the title bar and drag the window to a new location. Release the mouse button to attach or *dock* the window to its new location.
- Detailed instructions for using the *Output* window are provided in *Chapter 12: Testing and Debugging Your Application*.

CONFIGURING THE LOG TO SHOW ADDITIONAL INFORMATION

Note: By default, the log shows only debugging and error messages — that is, messages indicating that your IWS application 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:

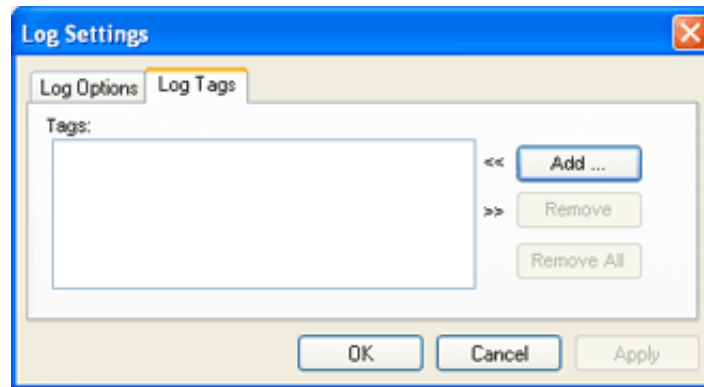
1. Right-click anywhere in the *Output* window and choose Settings from the pop-up menu. The *Log Settings* dialog is displayed.
2. In the *Log Options* tab of the dialog, click (enable) the specific types of messages that you want the log to show.



Log Settings — Options Tab

- **Field Read Commands** and **Field Write Commands** checkboxes: Show any read and/or write commands that are sent to connected devices.
- **Protocol Analyzer** checkbox: Show messages generated by configured *device drivers*.
- **OPC Messages** checkbox: Show messages generated by *OPC* communications.
- **TCP/IP Messages** checkbox: Show messages generated by *TCP/IP* communications.
- **Recipe/Report** checkbox: Show messages generated by the *Recipe* and *Report* tasks.
- **Display Open/Close** checkbox: Display a message whenever a *screen* is opened or closed.
- **Logon/Logout** checkbox: Display a message whenever a user logs on or logs out. (For more information, please see *Security*.)
- **Trace Messages** checkbox: Show messages generated by the `Trace()` function. This function is used to generate customized messages from within your IWS application.
- **Database Messages** checkbox: Show messages generated by the *ODBC* and *ADO.NET* database interfaces.
- **DDE Messages** checkbox: Show messages generated by *DDE* communications.
- **Insert date/time** checkbox: Timestamp each message.

- In the *Log Tags* tab of the dialog, click the **Add...** button to browse for *application tags*.



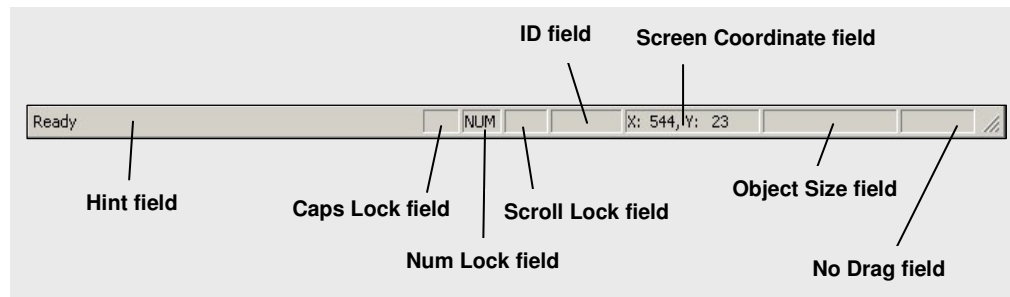
Log Settings — Tags Tab

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.

Using the Status Bar

The status bar (located along the bottom of the IWS window) contains fields used to identify toolbar buttons and provide information about the active screen (if any).



Example Status Bar

The fields are as follows (from left to right):


- **Hint field:** Provides a short description of any toolbar button or display object touched by the cursor.
- **Caps Lock field:** Indicates whether the keyboard Caps Lock is on (*CAP*) or off (*empty*).
- **Num Lock field:** Indicates whether the keyboard Num Lock is on (*NUM*) or off (*empty*).
- **Scroll Lock field:** Indicates whether the keyboard Scroll Lock is on (*SCRL*) or off (*empty*).
- **ID field:** Displays the ID number of a selected screen object.
- **Screen Coordinate field:** Displays the current location of the cursor (or pointer) on the active screen. Where: *X* is the number of pixels from the left edge of the screen and *Y* is the number of pixels from the top of the screen.
- **Object Size field:** Displays the size (in pixels) of a selected object, where *W* is the width and *H* is the height.
- **No DRAG field:** Indicates whether dragging is disabled (*No Drag*) or enabled (*empty*) in the active screen. You can toggle the *No Drag* feature by pressing **Ctrl+D**.

**Note:**

Use the **Ctrl+D** shortcut to enable/disable the *No Drag* feature when you edit the screen. You can use the *No Drag* feature to avoid moving objects on a screen when you are changing their properties.

Customizing the Workspace

InduSoft Web Studio allows you to customize the development environment:

- **Hide Docked Window** button (✖): Click to toggle between opening and closing the window. (Alternatively, you can use the  **Database Spy** button on the **View → Toolbars** menu or **Alt+2** to open and close this window.)
- **Contract/Expand** button (⏏): Click to toggle between contracting and expanding the *Database Spy* window.

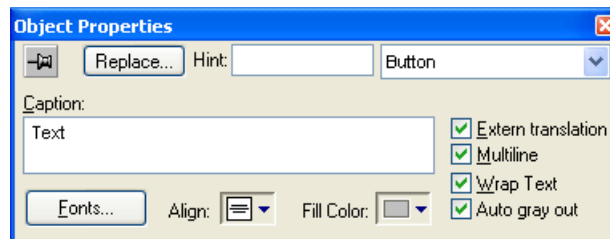
Standard Interfaces

The standard interfaces discussed in this section consist of basic interfaces that you use frequently as you work within the InduSoft Web Studio development environment. These include:

- Object Properties dialog
- Virtual Keyboard
- Fonts
- Color Interface


Object Properties Dialog


The **Object Properties** dialog is the interface which allows you to configure the settings for the objects and dynamics designed with the Screen Editor of InduSoft Web Studio. It can be launched just by double-clicking on the object or by selecting the object (clicking once on it) and selecting the menu option **View > Properties** (Shift+F2):



Object Properties Dialog

The content of this dialog depends on the object (or group of objects) selected by the user. However, the following interfaces are common for any object (or group of objects):

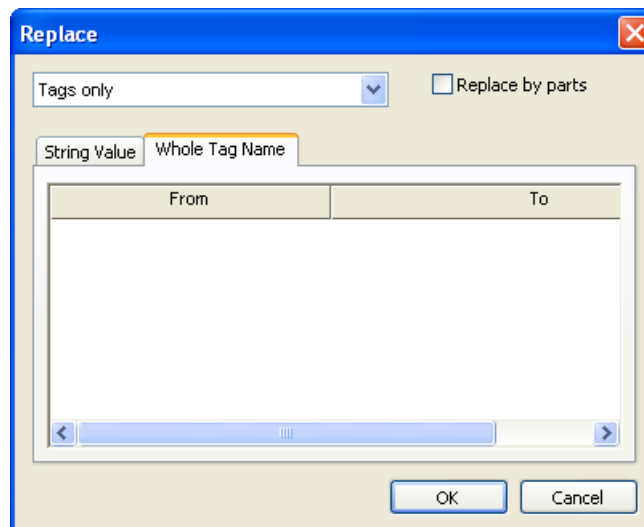
Field	Remarks	Syntax
 (Pin button)	When the pin button is released, the focus is passed to the object on the screen as soon as that object is selected. When the pin button is pressed the focus is kept on the Object Properties window, even when you click the objects on the screen.	Button
Replace	Launches the Replace Dialog , where you can replace strings, tags or properties for the selected object (or group of objects).	Button

Field	Remarks	Syntax
Hint	Tooltip displayed during the runtime, when the user keeps the mouse cursor on the object for a few seconds. This option is useful to provide hints to the operator during the runtime (quick-help). The text on the Hint field is also written to the System Tag Hint during the runtime. Therefore, you can trigger actions based on the value of this tag when the mouse cursor is moved on specific object.	Text and/or {Tag} (up to 256 chars)
	The combo-box at the right side of the dialog allows the user to select the specific object, dynamic or group of objects that must be edited.	Combo-box

Note:
The **Enable Tooltip** option must be enabled (checked) on the **Project Settings** dialog. Otherwise, the tooltips are NOT displayed during the runtime when the user points the mouse cursor on the objects. You can enable/disable this feature for the local server (**Project Settings > Runtime Desktop**) and/or for the Web Thin Client stations (**Project Settings > Web**).


REPLACE DIALOG

When pressing the **Replace** button from the Object Properties window, the Replace dialog is launched:



Object Properties: Replace Dialog

This dialog provides a fast and simple tool to replace strings (**String Value** tab), tags and/or properties for an object (or group of objects). This dialog display grids where you can edit the **To** column with the string, tag and/or property that must be replaced. The main interfaces in this dialog as described in the following table:

Field	Remarks	Syntax
	Allows you to filter the type of information that can be replaced: <ul style="list-style-type: none"> ▪ Tags only: Displays only the tags configured directly (e.g.: Second, Time, Level, and so forth). ▪ Tags + Properties: Displays both the tags configured directly and the properties configured as mnemonics (e.g.: #PumpStatus). ▪ Custom Properties: Displays only the mnemonics configured as custom properties (e.g.: #PumpStatus:). 	Button
Replace by Parts	When checked allows the user to replace each part of a tag separately: Main Tag Name, Array Index, Class Member and Tag Field. It is very useful when you want to replace a specific settings (e.g.: Array Index) for different tags configured in the same Group of Objects.	Button

⚠ Caution:

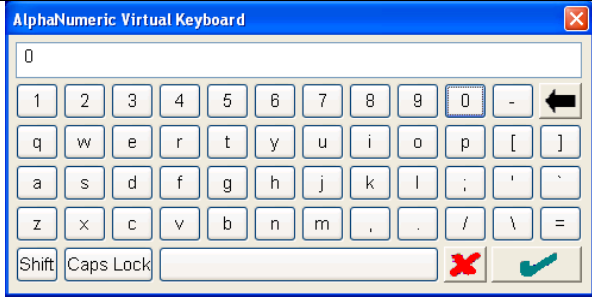
When pressing the **Replace** button after selecting the **Group of Objects** option in the Object Properties dialog, the replace feature affects all objects and dynamics which are part of the selected Group of Objects. However, when pressing **Replace** button after selecting a specific object or dynamic (e.g.: Button, Color, etc), the replace feature affects **ONLY** the selected object or dynamic.

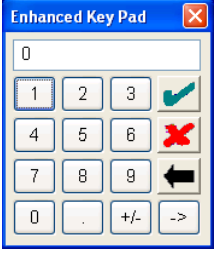
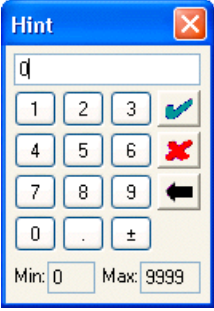
Virtual Keyboard

The IWS Virtual Keyboard (VK) allows the user to enter data (text or numbers) during the runtime using a touch-screen device, rather than a physical keyboard.

You can enable the VK for your application on the Runtime Desktop dialog (Project pulldown menu item->Settings->Runtime Desktop tab). You can designate a Default Virtual Keyboard, as well as apply a Scale (size) to all Virtual Keyboards during the runtime.

When configuring objects and dynamics that support data input, you can assign a VK to the object by the VK combo-box (either the default or a specific one). The following standard VKs are available:

Name	Picture	Description
AlphaNumeric		This VK is used to enter alphanumeric values (chars and/or numbers).

EnhKeypad		This VK is used to enter alphanumeric values (characters and/or numbers) in devices that provide a small display (e.g. PDAs). The -> button lets users scroll through different panels until they find the character they want.
Keypad		This VK is used to enter numeric values (numbers only).

There are two ways to launch a Virtual Keyboard during the runtime:

- Click on the object where a new value (data) can be entered (e.g. Text with Text I/O dynamic configured with Input Enabled).
- Execute the Keypad() built-in function.
- You can change the language of the Virtual Keyboard by the SetKeyboardLanguage() built-in function.

⇒ Tips:

- You can create new languages for any Virtual Keyboard by editing the VK<Language>.INI file from the \BIN sub-folder of IWS, where <Language> is the label of the language for the VKs (e.g. VKEN.INI = Virtual Keyboards in English; VKGE= Virtual Keyboards in German, and so forth). Consult your software vendor for more information about how to edit and configure this file.
- By default, the VK is launched close to the object on the screen associated with it. However, you can configure the following settings in the <ApplicationName>.APP file, so the VK will be always open in a fixed position:

[Keypad]

PosX=0 //Coordinate (in pixels) of the TOP coordinate where the VK must be launched.

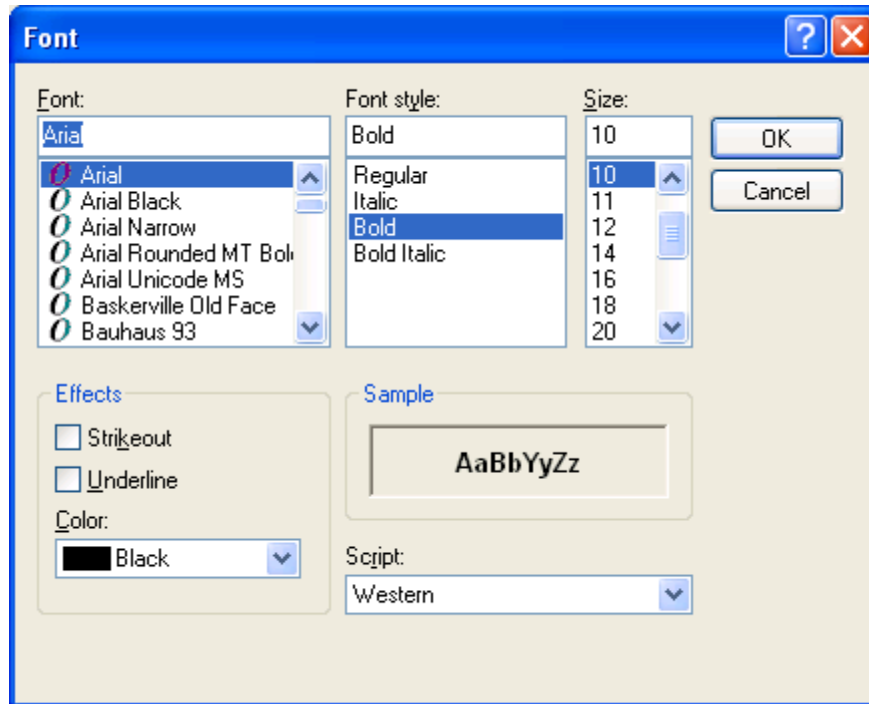
PosY=0 //Coordinate (in pixels) of the LEFT coordinate where the VK must be launched.

Fonts

IWS supports any UNICODE font available on the operating system where IWS is running. Therefore, it is possible to configure interfaces using characters for languages that do not use the standard western characters, such as Japases, Chinese, Arabic, Cyrillic, etc.

The font used on the development environment of IWS (Worksheets, Dialogs, etc.) is the default font installed by the operating system and dependent on the language of the operating system. Click the Edit Set Font menu option to select a different font for development.

When editing the objects that display text during the runtime, you can set the font that will display the text by clicking on the Fonts button in the Object Properties window. The Font button launches the standard Fonts dialog:



Font Dialog

You can set the font name, style, size, effects and script.

⇒ **Tip:**

You can change the font style of several objects simultaneously by selecting them all (press the shift key down while you click each one), and then clicking on the Fonts icon from the Mode toolbar of the screen editor.

The icon displayed to the left of the font name indicates the font technology.

Icon	Technology	Remarks
	TrueType	Outline TrueType and OpenType fonts are outline fonts that are rendered from line and curve commands. OpenType is an extension of TrueType. Both can be scaled and rotated. Both look good in all sizes and on all output devices supported by Windows. Windows provides a selection of OpenType fonts, including Arial, Courier New, Lucida Console, Times New Roman, Symbol, and Wingdings.
	OpenType	
N/A	Vector	Type 1, by Adobe Systems, Inc., is an outline font that is designed to work with PostScript printers. The outlines can be scaled and rotated. With OpenType technology, Windows fully supports Type 1 fonts. Vector fonts are supported because a number of programs still depend on them. Vector fonts are rendered from a mathematical model. They are used primarily with

		plotters. Windows supports three vector fonts: Modern, Roman and Script.
N/A	Raster	Raster fonts are supported because a number of programs still depend on them. Raster fonts are stored in files as bitmap images and are composed of a series of dots whether they are displayed on the screen and on paper.

It is strongly recommended that you use only TrueType or OpenType fonts. Fonts designed with other technologies (e.g. Courier) cannot be scaled properly and could cause issues during the runtime.

⚠ Caution:

When you design screens, the fonts you use are the ones available in the operating system of your development station. The fonts on the runtime station, however, may look different (e.g. different size in pixels), even if all settings are the same on both stations. This situation occurs more frequently when applications are run on the Windows CE operating system, where the fonts do not always match the proportions of the equivalent fonts on Windows 2K/XP/Vista. Therefore, it is important to test the graphic interfaces (screens) on the actual runtime platform during the development of the application. You should not wait until after the whole application has been developed, or it may become necessary to re-design the screens so the text objects display properly on the runtime platform.

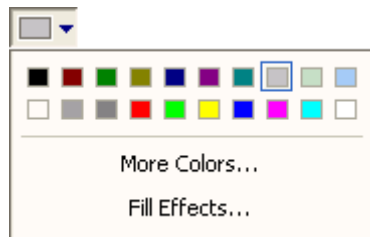
Color Interface

You can edit the color of a component with the Color interface.

- ☑ Click the icon in the toolbar:

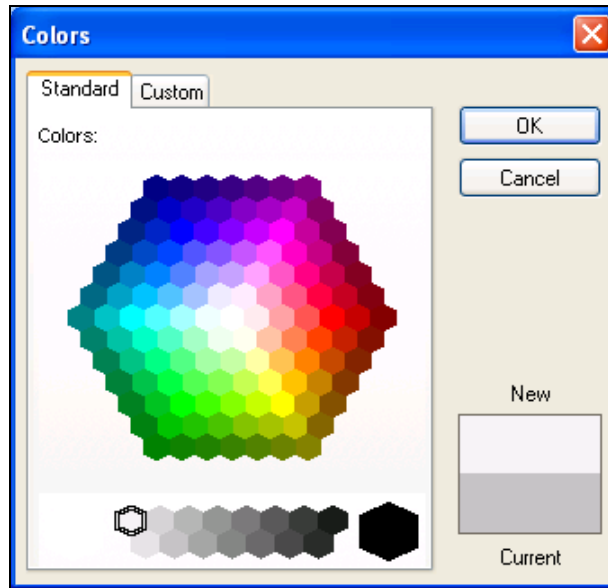


- ☑ Click the desired color from the twenty that display when the pop-up box opens:



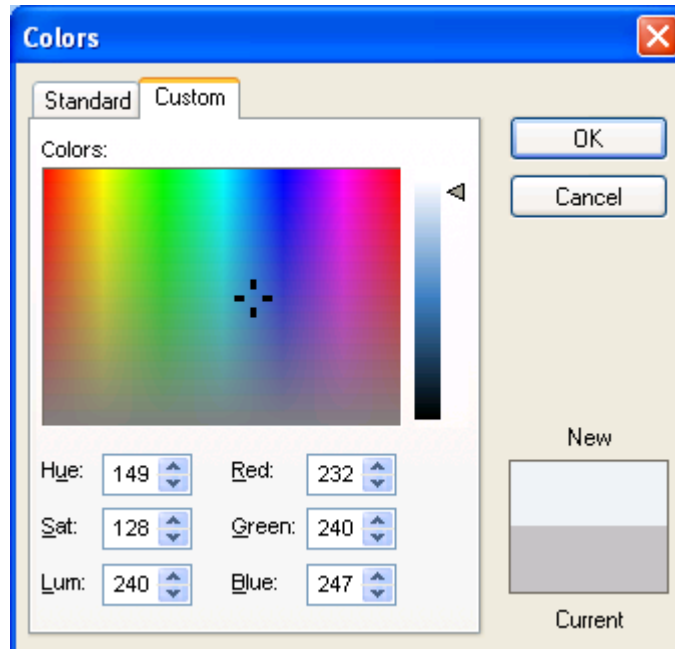
The selected color will be applied to the component that you are editing.

- ☑ Click More Colors... if you want to apply a different color. The Colors dialog will open, displaying the 143 standard colors from your operating system.



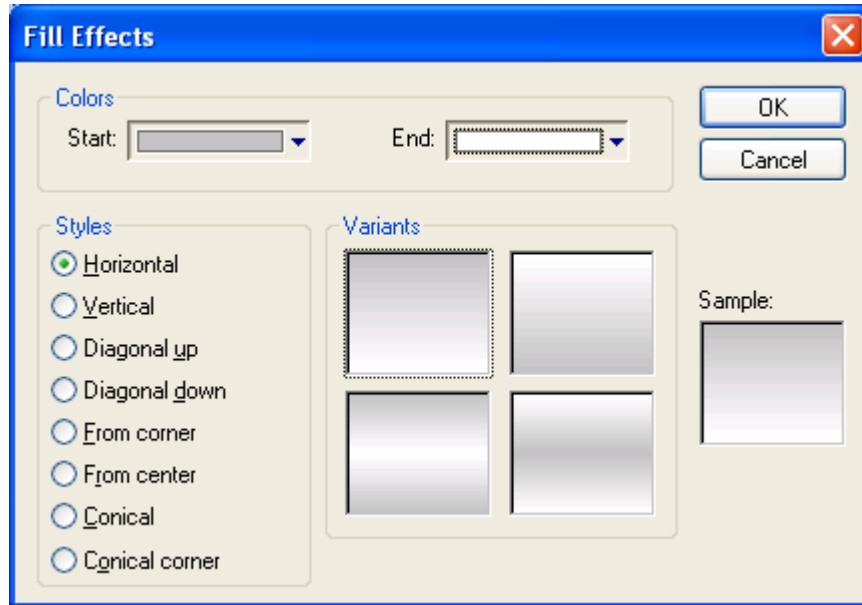
Colors Dialog

- ✓ Click the Custom tab to edit the HSL (Hue, Sat, Lum) or RGB (Red, Green, Blue) codes of any of the 143 standard colors, creating a custom color.



Custom Colors Tab

- ✓ Click the OK button to apply the selected color to the component that is being edited.
- ✓ Depending on the component that you are editing, the Fill Effects option is available from the pop-up interface (see step 2 above). Click this option to apply gradient colors with different styles and variants. The Fill Effects dialog will open.



Fill Effects Dialog


- ☑ Select two colors in the Start and End fields, select the Style, and click on the chosen Variant. Finally, click the OK button to apply the fill effect to the component which is being edited.

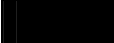

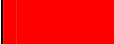

Note:

For applications for the WinCE operating system, the Fill Effects interface is available only for the Rectangle object.

Caution:

Although Fill Effect is a useful tool for enhancing the look and feel of your screens, the operating system takes a longer time to fill an object with fill effects than with plain colors. You should develop criteria for using the feature without decreasing the performance of the system, especially under the WinCE operating system.

Using the Color dynamic, , you can modify the color of a static object during the runtime. When configuring this dynamic with Type = By Color, you can set the color that will be applied in the object during the runtime, by the color code. The following table provides the code values as well as the RGB values for the most commonly used colors:

Color	Name	RGB Code			Code Value
		R (Red)	G (Green)	B (Blue)	
	Black	0	0	0	0
	Dark Red	128	0	0	128
	Red	255	0	0	255
	Pink	255	0	255	16711935

	Rose	255	153	204	13408767
	Brown	153	51	0	13209
	Orange	255	102	0	26367
	Light Orange	255	153	0	39423
	Gold	255	204	0	52479
	Tan	255	204	153	10079487
	Olive Green	51	51	0	13107
	Dark Yellow	128	128	0	32896
	Lime	153	204	0	52377
	Yellow	255	255	0	65535
	Light Yellow	255	255	153	10092543
	Dark Green	0	51	0	13056
	Green	0	128	0	32768
	Sea Green	51	153	102	6723891
	Bright Green	0	255	0	65280
	Light Green	204	255	204	13434828
	Dark Teal	0	51	102	7877376
	Teal	0	128	128	8421376
	Aqua	51	204	204	13421619
	Turquoise	0	255	255	16776960
	Light Turquoise	204	255	255	16777164
	Dark Blue	0	0	128	8388608
	Blue	0	0	255	16711680
	Light Blue	51	102	255	16737843
	Sky Blue	0	204	255	16737843
	Pale Blue	153	204	255	16764057
	Indigo	51	51	153	10040115
	Blue-Gray	102	102	153	10053222
	Violet	128	0	128	8388736
	Plum	153	51	102	6697881
	Lavender	204	153	255	16751052
	Gray-80%	51	51	51	3355443
	Gray-50%	128	128	128	8421504
	Gray-40%	150	150	150	9868950
	Gray-25%	192	192	192	12632256
	White	255	255	255	16777215

⇒ Tip:

The `RGBColor()` and `RGBComponent()` functions can be used to manipulate color codes during the runtime.

🔗 Note:

The number of colors available when developing the application depends on the color settings configured on the operating system of the development station. The number of colors available when running the application (runtime) depends on the color settings configured on the operating system of the runtime station.

Performing Common Tasks

The common tasks discussed in this section consist of basic procedures that you use frequently as you work within the InduSoft Web Studio development environment.

These tasks include:

- Accessing Projects and Files
- Using Common Buttons
- Managing the Development Environment Windows
- Changing the Screen Resolution
- Using Pop-Up Menus
- Using Select All
- Cutting, Copying, Pasting Objects
- Using the Symbols Library
- Finding System Information
- Searching for Tags and Screen Objects
- Replacing Tags
- Testing Displays
- Verifying the Application
- Running Applications
- Restoring Defaults
- Saving Your Work
- Printing Project Screens

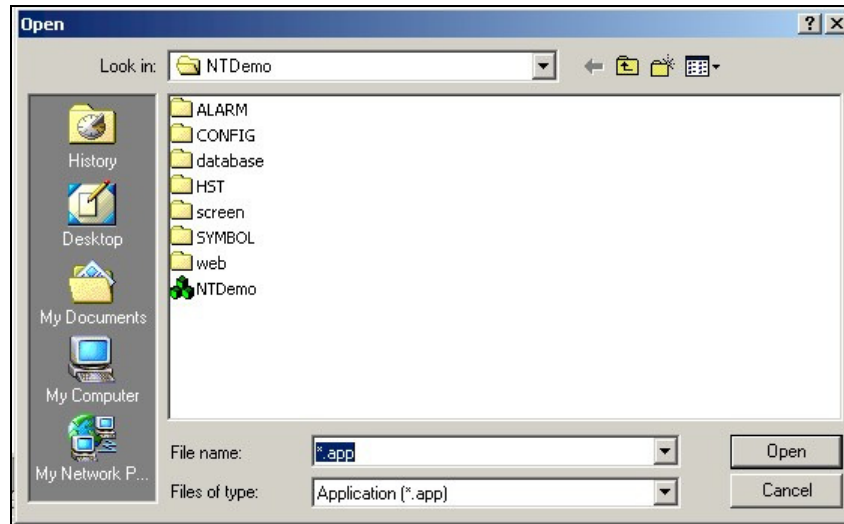
Accessing Projects and Files

To create a new project, see *Chapter 6: Creating and Configuring a Project*.

OPENING PROJECTS

To open a project, from the *Main* menu bar, select **File > Open Project**.


Selecting the **Open Project** option displays the *Open* dialog, which lists all existing folders. You can use the *Open* dialog to locate and open an InduSoft Web Studio project.



Open Dialog

Note:

Alternatively, you can open a new project using one of the following two methods:

- Click the **Open Project** button  on the *Standard* toolbar to display the *Open* dialog.
- Open *Windows Explorer*, locate the *InduSoft Web Studio* folder, and then double-click on the project name.


OPENING FILES

From the *Main* menu bar, select **File > Open File**. Selecting the **Open File** option displays the *Open* dialog (as shown in the preceding section), which lists all existing folders. To locate and open an InduSoft Web Studio application file from this dialog, click the **Files of type** combo-box button, and then click on a file name to select it from the list.

CLOSING PROJECTS

From the *Standard* toolbar, select **File → Save** to save any active screens or worksheets. The **Save** option becomes enabled (*active*) only after you modify the active file.


Note:

You can also use the **Save** button () on the *Standard* toolbar or type **Ctrl+s** to save the open, active screen/worksheet.

From the *Standard* toolbar, select **File > Save As** to save active screens or worksheets, and to specify a (new) name and location for the file.

Select the **Save As HTML** option to save the active display in HTML format. You can also select **File > Save All** from the menu bar to save all open screens or worksheets. The **Save All** option becomes enabled (*active*) only after you modify the active file.

Note:

Using **File > Save All** is the same as using the **Save All** button () on the *Standard* toolbar.


Select the **File > Save All As HTML** option to save *all* displays of the applications in HTML format. You have to close all documents before executing this command.

Select **File > Save Screen Group As HTML** to save the *Screen Group* in HTML format, making them available to the remote Web Thin Client through a Web Browser.

CLOSING FILES

From the menu bar, select **File > Close** to close the active screen or worksheet. IWS prompts you to save all unsaved changes before it closes the screen/worksheet.

Note:

Using **File > Close** is the same as using the **Exit** button () located on the title bar.

You can also select **File > Close All**. Selecting the **Close All** option closes all open screens or worksheets. IWS prompts you to save all unsaved changes before it closes the screens/worksheets.

Using Common Buttons

The following table describes buttons that typically appear on IWS dialog and windows:

Button	Purpose
OK	Click this button to execute and save all changes, and close the dialog or window.
Apply	Click this button to execute and save all changes, but leave the dialog or window open. This button enables you to see the effects of your changes before closing the dialog/window.
Cancel	Click this button to close the dialog or window immediately (discarding any changes).
Open	Click this button to open a file. Generally, this button is associated with a combo-box or list pane. You use the combo-box or list pane to specify a file and then click the Open button to open the file.

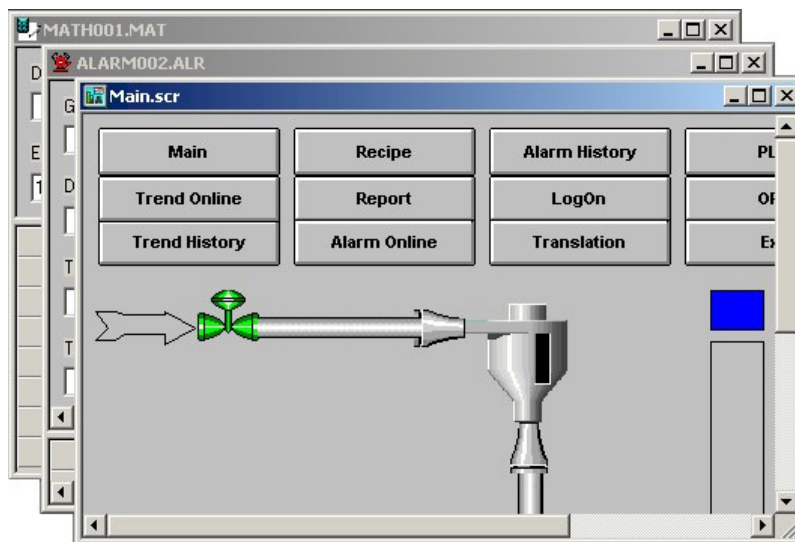
Button	Purpose
Close	Click this button to close the open file, screen, dialog, and so forth.
Browse	Click this button to open a <i>Browse</i> dialog to search for a file or folder to open.
Back	Click this button to progress to the previous screen in a sequence of screens.
Next	Click this button to progress to the next screen in a sequence of screens.
Replace	Click to open a <i>Replace</i> dialog, which enables you to change tags or strings associated with a selected screen object.
Remove	Click to remove a selected (highlighted) object from a list or a display screen.

Common Buttons Table

Managing the Development Environment Windows

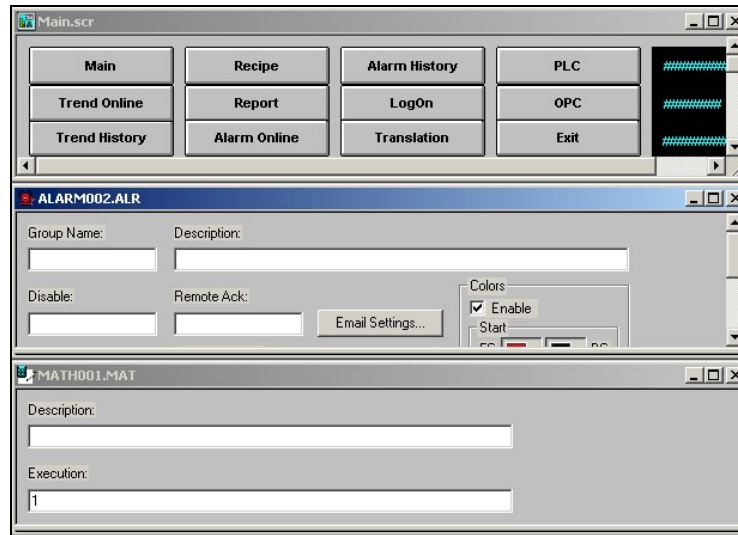
When you click the **Window** menu on the main menu bar, the following options are available to help you manage your open screens and worksheets:

- **Cascade** arranges the open screens or worksheets in a cascade within the *Screen/Worksheet* window.



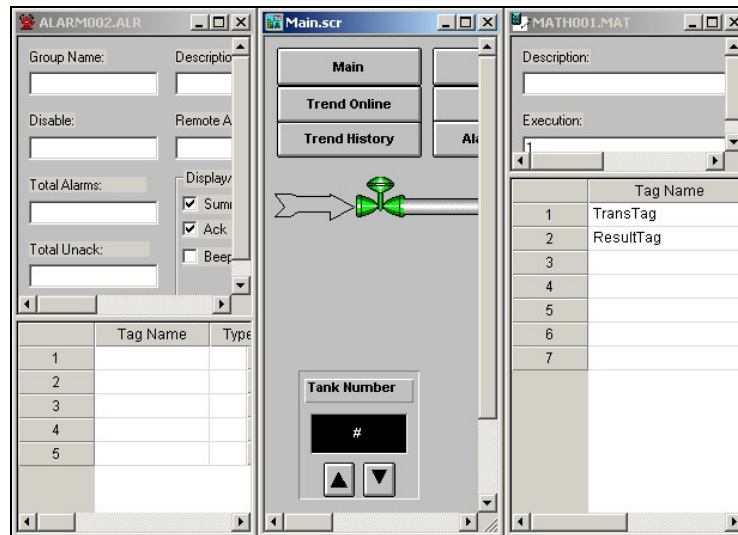
Cascading Screens

- **Tile Horizontally** arranges the open screens or worksheets:



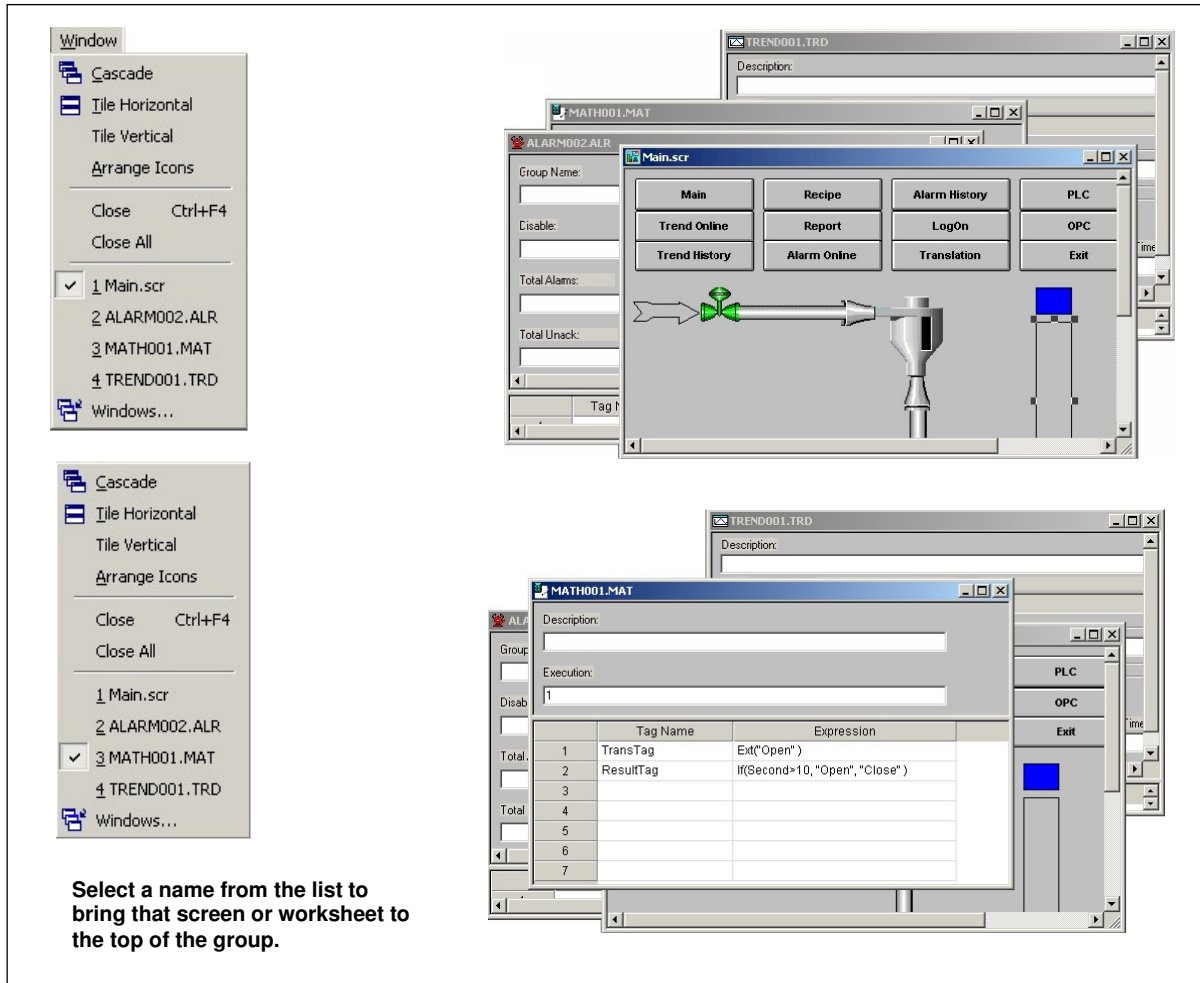
Horizontally Tiled Screens

- **Tile Vertically** arranges the open screens or worksheets:



Vertically Tiled Screens

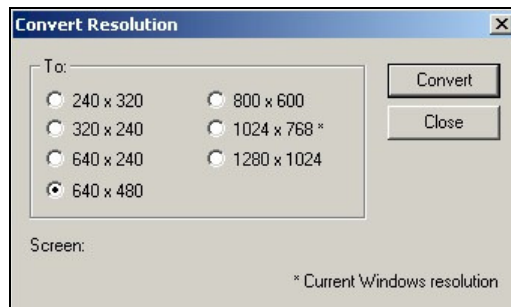
- **Arrange Icons** moves and arranges minimized screens or worksheets along the bottom of the *Screen/Worksheet* window.
- **Close** closes only the active (*selected*) screen or worksheet.
- **Close All** closes all open screens or worksheets.
- **Currently Open:** Lists all of the open screens or worksheets. You can select a name from the list to bring that screen or worksheet to the top of the group.
- **Windows...:** Contains options that enable you to manage open displays and worksheets.



List of Screens

Changing the Screen Resolution

Close all open documents and then select **Tools > Convert Resolution** to open the *Convert Resolution* dialog, which allows you to change the resolution of your application.



Convert Resolution Dialog

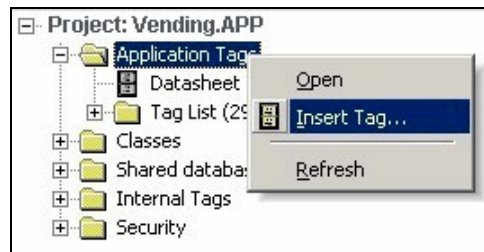
To change the application resolution:

- ☑ Click (*enable*) the radio button of the resolution you want to use to convert the application.
- ☑ Click the **Convert** button.
- ☑ Click the **Close** button to close the dialog when you are finished.

Before converting the application to the new resolution, IWS backs up all screens at their previous size in a *Backup* folder located in the *Screen* folder in your \<application>\ directory. The asterisk (*) next to the resolution denotes the current (base) resolution. (**Note:** When you select a new resolution, all screens will be scaled from this new base resolution.)

Using Pop-Up Menus

If you right-click on any component in the *Workspace*, a menu displays with options related to that component. For example, the following pop-up menu enables you to **Open** the *Application Tags* database, **Insert** (create) a new tag, or **Refresh** the current view of the *Application Tags* database:



Right-Click to Open a Pop-Up Menu


Using Select All

From the menu bar, select **Edit > Select All** to select all objects on the active screen, or press **Ctrl+A** on the keyboard.

Cutting, Copying, Pasting Objects


From the menu bar, select **Edit > Cut** to remove a selected item from a screen and store it on the Windows clipboard (replacing any previously selected objects stored on the clipboard). You use **Cut** in conjunction with the **File > Paste** option to move an object to another location on the same screen or onto a different screen.

Note:

Using the **Cut** menu option is the same as using **Cut** button  located on the *Standard* toolbar.


From the menu bar, select **Edit > Copy** to duplicate a selected item and store it on the Windows clipboard. Use **Copy** in conjunction with the **Paste** option to place a copied object in another location on the same screen or onto a different screen. You can paste a copied object multiple times.

Note:

Using the **Copy** menu option is the same as using **Copy** button  located on the *Standard* toolbar.


From the menu bar, select **Edit > Paste** to place the contents of the Windows clipboard (cut or copied objects) onto the active screen. If the clipboard contains an object, IWS will paste that object into the upper left corner of the screen. You can paste a cut or copied object multiple times.

Note:

- You can easily copy (*drag*) selected items by holding down the **Ctrl** key + left mouse button.
- Using the **Paste** menu option is the same as using **Paste** button  located on the *Standard* toolbar.

You can select **Edit > Undo** to undo 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).

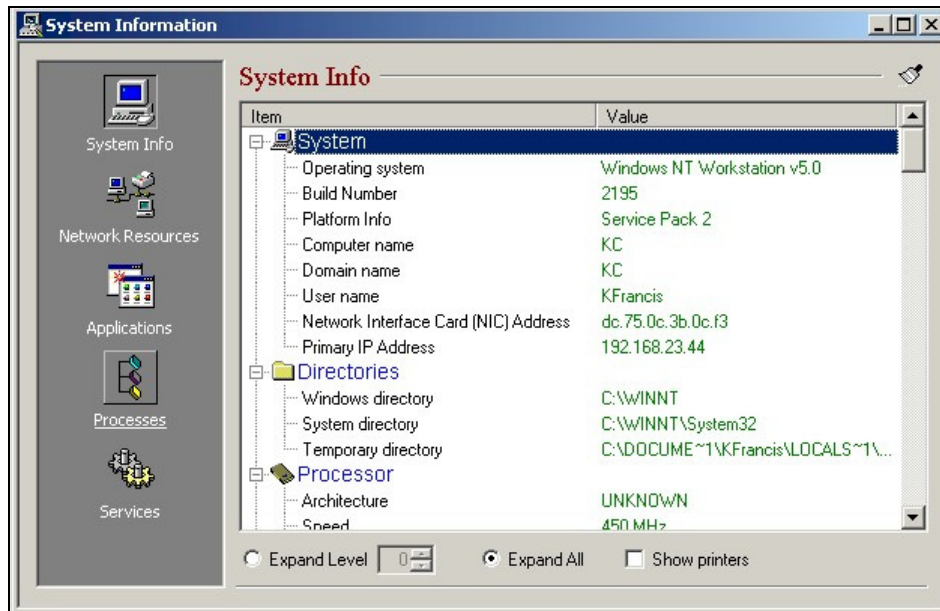
Note:

Using the **Undo** menu option is the same as using **Undo** button  located on the *Standard* toolbar.

Finding System Information

From the menu bar, select **Tools >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*).



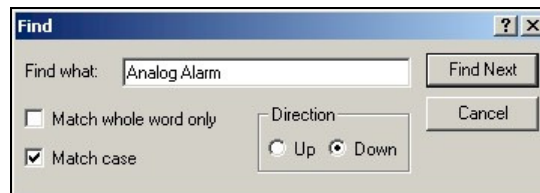
System Information Window

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.

Searching for Tags and Screen Objects

USING THE FIND OPTION

Select **Edit > Find** to open the *Find* dialog, which allows you to search for a word in the active worksheet.



Find Dialog

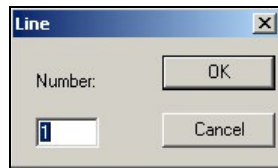
Type the text string in the **Find What** text box and then specify search parameters by clicking on one or more of the following:

- **Match whole word only:** IWS searches for the specified term, and does not include instances where the specified term is part of a larger word. For example, if you specify *back*, IWS finds only *back*, not *backward*.
- **Match case:** IWS searches for term using the specified capitalization. For example, if you specify *TankLevel*, IWS does not search for *tanklevel*.
- **Direction:** Controls which direction IWS searches the worksheet (**Up** or **Down**).

Click on the **Find Next** button to begin the search. (Click the button again to resume searching the worksheet.)

USING GO TO OPTION

From the main menu, select **Edit >Go to** to jump to a line in an open worksheet or to select an object using the object's ID number. IWS applies a sequential identification number (ID) to each object created on the screen. When you select an object, the ID number displays in the status bar. Objects are numbered according to how they are layered, starting with the number zero (the backmost object). These objects are renumbered when you bring them to front or back. When you group objects, they all become one layer.



Line Dialog

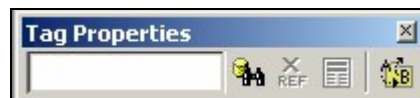
To jump to a line or an object, select **Go to**. When the *Line* dialog displays, type the line number or object ID number into the **Number** text box and click the **OK** button.

⇒ **Tip:**

If you have many superimposed objects, and it is not possible to select an object using the pointing device, you can use the **Go to** option to edit the properties of an object that is under other objects.


Using the Tag Properties Toolbar

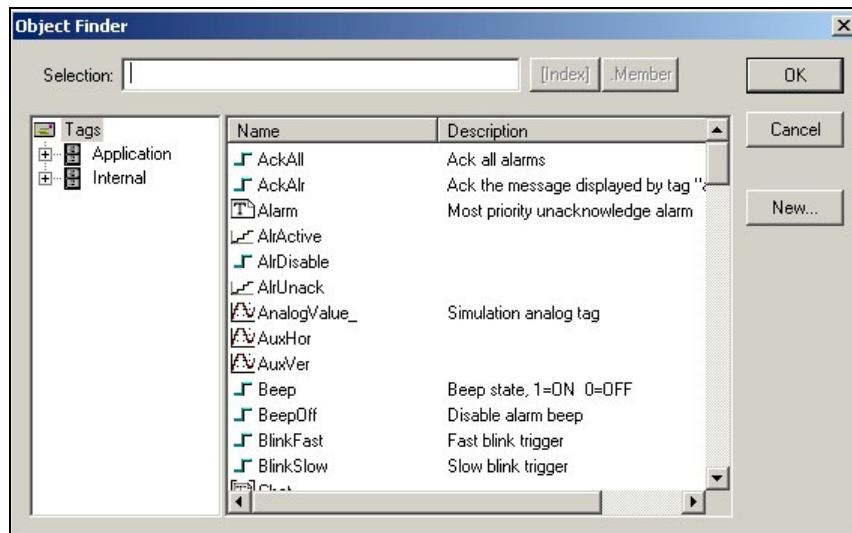
The *Tag Properties* toolbar provides a text box and several buttons (shortcuts) that enable you to create, locate, and access different tags, functions, and tag properties.



Tag Properties Toolbar

USING THE OBJECT FINDER


Click the **Object Finder** button  to open the *Object Finder* dialog, which lists all **Tags** and **Functions** currently configured for the application.

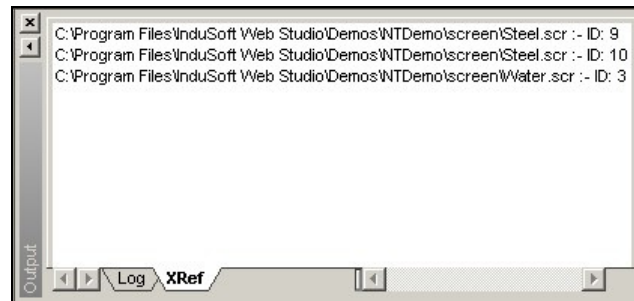


Object Finder Dialog

- To select an existing tag/function, double-click on the tag/function name, and then click **OK** to close the dialog. 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 dialog:
 - **Name**
 - **Array Size**
 - **Type** (Boolean, Integer, Real, String, Class:Control, Class:msgonline, or Class:Alr)
 - **Description**
 - **Web data** (local or server)

USING THE X-REF OPTION

Click the **Cross Reference** button  to search all application screens and worksheets for the tag noted in the **Tagname** text box. This function writes a log, detailing all the occurrences of the tag, to the *XRef* tab in the *Output* window. For example, the results of searching for a *BlinkFast* tag are as follows:

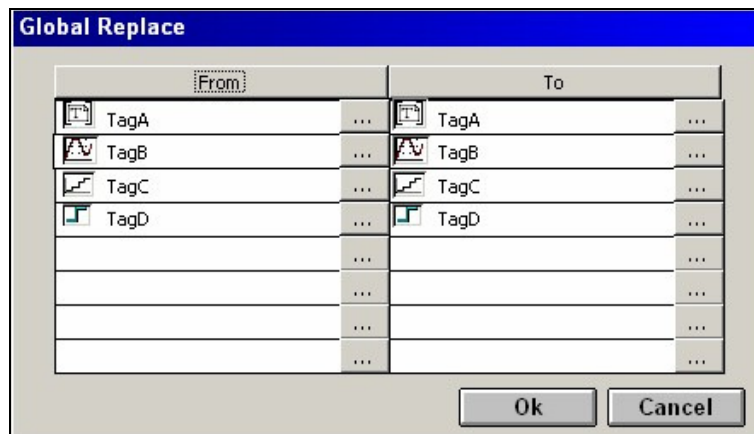


XRef Results

See also “Using the Output Window” on page 3–24.

USING THE GLOBAL TAGS REPLACE OPTION

When you select the **Global Tags Replace** button from the *Tag Properties* toolbar, the *Global Replace* dialog displays:



Global Replace Dialog

From the *Global Replace* dialog, you can replace any tag(s) from all documents (screens and worksheets) of the whole application. 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:

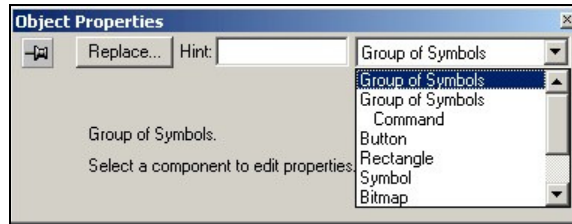
You must close all documents (screens and worksheets) before executing this command.

When changing the tag name on the *Tags Database* worksheet, IWS will ask you if you intend to replace this tag through the whole application.

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 to the current screen or worksheet in focus.

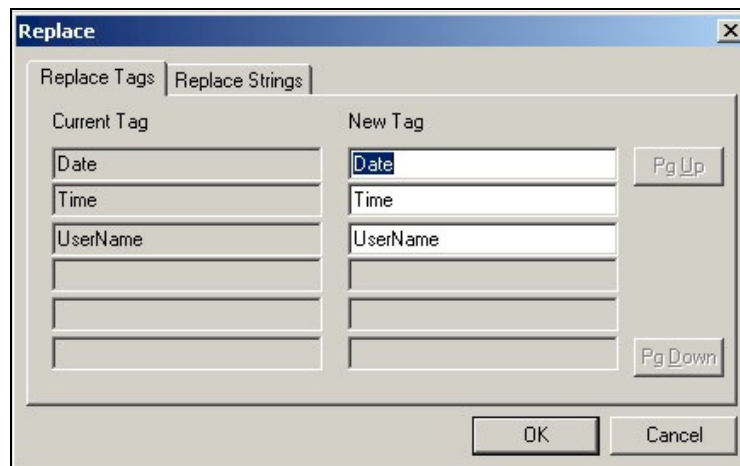
Replacing Tags

Double-click the object to open the *Object Properties* dialog. To replace tags, select the **Replace** button located on the *Object Properties* dialog.



Object Properties Dialog

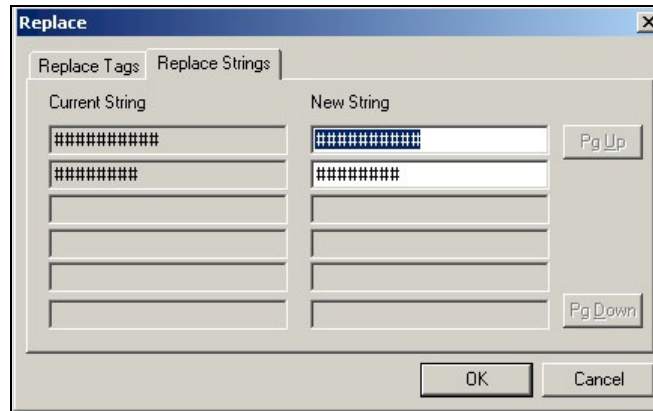
You can select one or more replacement tags by selecting the **Replace Strings** tab. Current tags used are displayed. You can type the **New Tag** to the right of each **Current Tag**.



Replace Tags Tab

You can also select one or more replacement strings by selecting the **Replace Strings** tab. Current strings used are displayed. You can type the **New String** to the right of each **Current String**.

When you are finished, click the **OK** button.




Replace Strings Tab

Testing Displays


From the menu bar, select **Project > Test Display** to activate the test display mode, which allows you to configure the application while viewing graphical dynamics online in the development environment. The test display mode does not enable you to use the *Command* or input *Text I/O* dynamics nor execute worksheets.

 **Note:**

Using the **Test Display** menu option is the same as using the  button on the *Execution Control* toolbar.

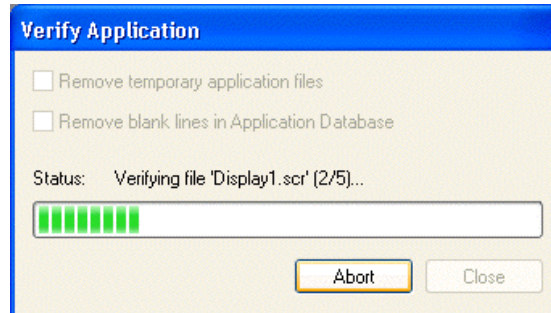
To stop the test display mode, select **Project > Stop display test**.

 **Note:**

Using the **Stop display test** menu option is the same as using the  button on the *Execution Control* toolbar.

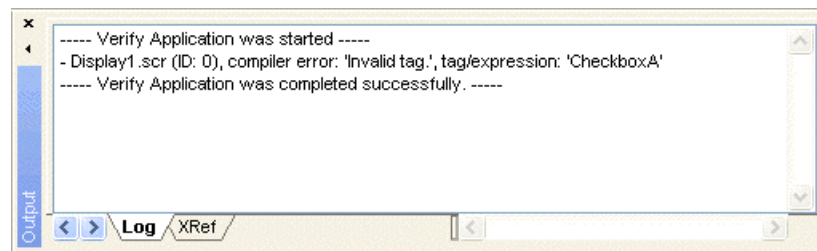
Verifying the Application

From the menu bar, select **Tools > Verify Application** to recompile all Math worksheets, expressions, and screen logic, as well as to update current HTML files using the settings configured on the **Web** tab of the *Project Settings* dialog.



Verifying an Application

If there are any tags used in the application that are not defined in the database, then IWS will indicate where (i.e., the screen or worksheet file) those tags are used.



Invalid Tag Found in Screen File

Tip:

When you save a screen or worksheet, IWS includes a pointer to the current database version. When you execute the application, IWS compares the screen or worksheet database to the current application database and if there is a mismatch, IWS recompiles the expressions.

To avoid doing these tasks during application runtime, we recommend running the **Verify Application** tool before downloading and/or finishing an application. You should also use this function when converting an application to a new version of the program.

Note:

Selecting the **Verify Application** tool will delete all temporary files (e.g., *.txt, *.mac and *.tag files) from the application folder to save disk space and clean the application before downloading it to the runtime station.


Verifying an application *does not* automatically download it to the runtime station. To download the application, you must use the **Send project to target** tool.

Running Applications

From the menu bar, select **Project > Run Application** to launch the runtime modules specified as **Automatic** on the *Project Status* dialog (**Execution Tasks** tab).

- When you start the *Viewer* module, it opens the screen(s) currently being edited.
- If you do not specify any *Automatic* tasks, InduSoft Web Studio will launch the *Viewer* and *BGTask* tasks automatically when you execute **Run Application**.
- If you are not currently editing screens in the development environment, the *Viewer* module opens the screen specified in the **Startup screen** field on the **Runtime Desktop** tab (*Project Settings* dialog).

 **Note:**

Using the **Run Application** option is the same as using the  button on the *Execution Control* toolbar.

 **Caution:**


Run Application affects the application from the target station (configured in the *Execution Environment* dialog). Be sure you know which target station is configured (local or remote) before executing the **Run Application** command.

To stop all runtime tasks, select **Project > Stop Application**.

 **Caution:**

Stop Application affects the application from the target station (configured in the *Execution Environment* dialog). Be sure you know which target station is configured (local or remote) before executing the **Stop Application** command.


 **Note:**

Using the **Stop Application** option is the same as using the  button on the *Execution Control* toolbar.

Restoring Defaults


From the menu bar, select **View > Restore Defaults** after adding or modifying the interface to return to the IWS default development environment. You will need to close and reopen IWS for the changes to take effect.

Saving Your Work

Click the **Save** button  to save any active screens or worksheets.

 **Notes:**

- Using the **Save** button is the same as selecting **File** → **Save** from the menu bar or typing the **Ctrl+S** key combination.
- The **Save** function becomes available only when you modify the active file.

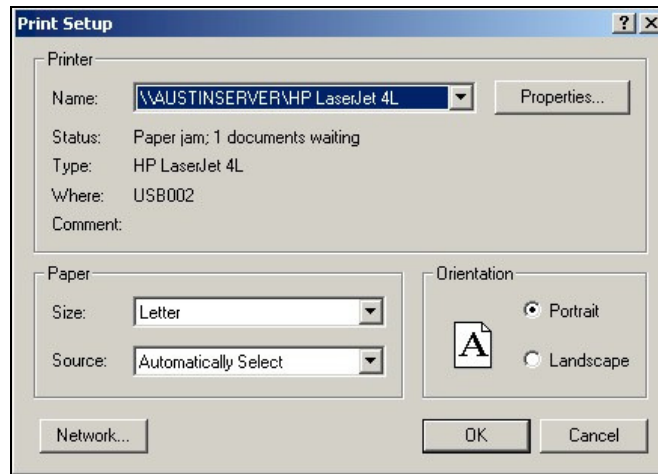
Click the **Save All** button  to save all open screens or worksheets.

Note:

- Using the **Save All** button is the same as selecting **File** → **Save All** from the menu bar.
- The **Save All** function becomes available only when you modify a screen or worksheet.

Printing Project Screens

From the menu bar, select **File** > **Print Setup** to open the *Print Setup* dialog. From this dialog you can specify which printer to use and configure different printing options, such as paper size and print orientation (portrait or landscape).



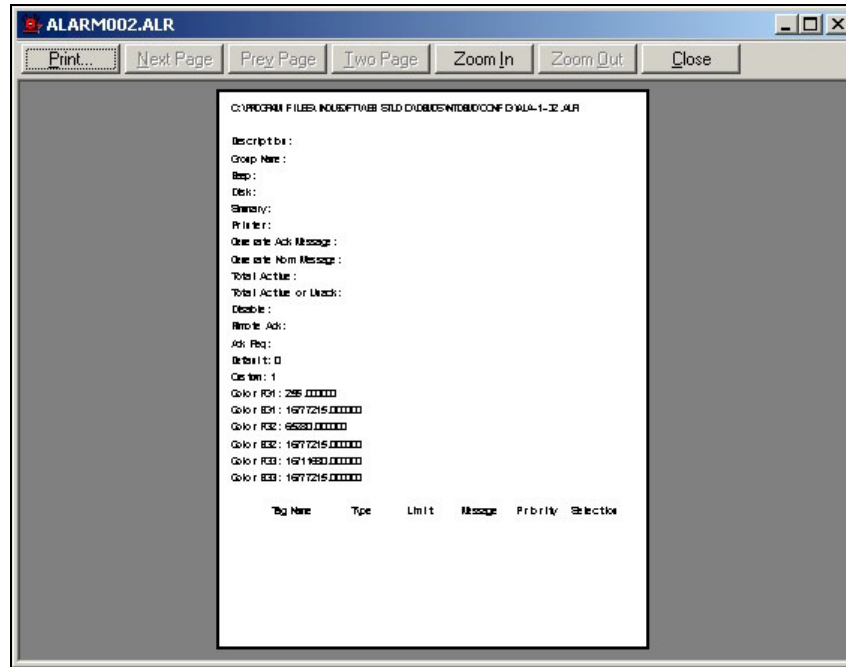
Print Setup Dialog

Note:

To specify a *default printer*:

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

To preview a screen before printing, select **File > Print Preview** to open the *Print Preview* window.



Previewing an Alarm Worksheet

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 (same as using the **File > Print** command).
- 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.
- Click **Zoom In** to check details.
- Click **Zoom Out** to change back to the default size.


Notes:

- The **Next Page**, **Prev Page**, and **Two Page** buttons become active only when you are printing more than one page.
- The **Zoom Out** button becomes active after you **Zoom In**.

- Click **Close** to close the *Print Preview* window.

You can also select **File > Print** from the menu bar open the *Print* dialog (identical to the *Print Setup* dialog discussed previously). You can use the *Print* dialog to print the active screen or worksheet.

Note:

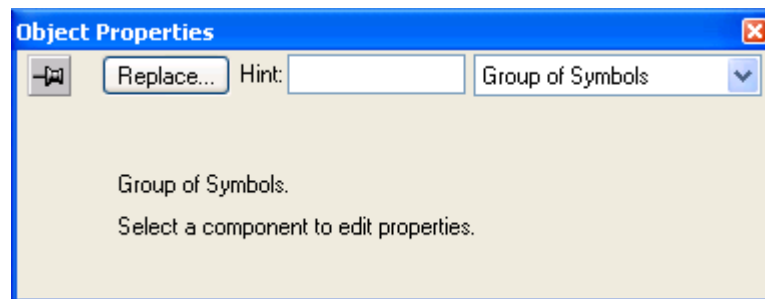
Selecting the **Print** menu option is the same as using the **Print** button  on the *Standard* toolbar.

Tips and Tricks



This section provides useful tips and tricks to help you take full advantage of the tools provided by IWS, enhancing your productivity when developing applications with IWS.

Configuring the Focus of the Object Properties Window on the Screen Editor

When you double-click any object (or group of objects) in the Screen Editor, the Object Properties window is launched, allowing you to configure the selected object's settings. The content of this dialog window varies according to the specific object/dynamic that is being edited. However, there is always a pin button in the left upper corner of this dialog window:



Object Properties Dialog

The pin button looks like this, , when it is released, and like this, , when it is pressed.



When the pin button is released, the focus is passed to the object on the screen as soon as that object is selected. Therefore, we recommend you keep this button released when you want to manipulate (copy, paste, cut or delete) the objects. Although the Object Properties window is on the top, the keyboard commands (Ctrl+C, Ctrl+V, Ctrl+X or Del) are sent directly to the objects.



When the pin button is pressed the focus is kept on the Object Properties window, even when you click the objects on the screen. We recommend you keep this button pressed when you want to modify the settings of the objects. You can click an object and type the new property value directly in the Object Properties window (it is not necessary to click on the window to bring focus to it). Also, when the pin button is pressed, the Object Properties window does not automatically close when you click on the screen.

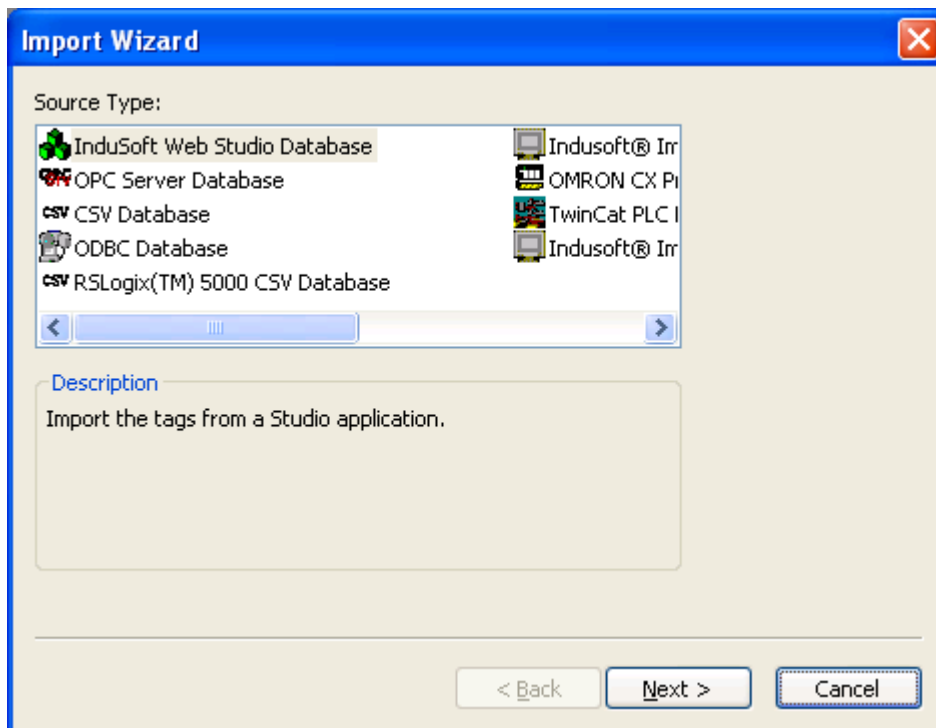
Importing a Database

The Import Wizard is a powerful tool that reduces engineering time during application development. Using the Import Wizard, you can import tags from different data sources directly to the IWS 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 select File > Import Wizard, 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:

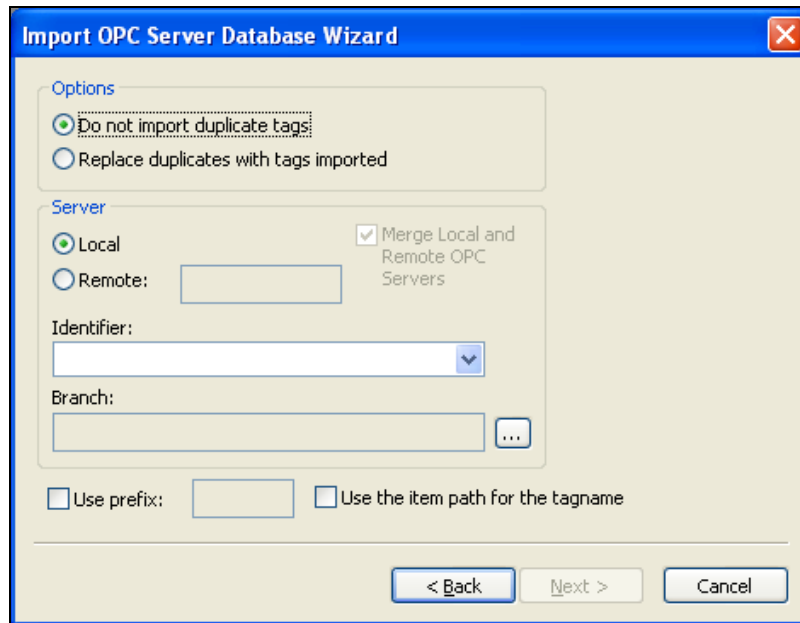
- InduSoft Web Studio Database
- OPC Server Database
- CSV Database
- ODBC Database
- PanelBuilder32™ Database
- RSLogix™ 5000 CSV Database
- PanelMate Plus™ Database
- OMRON CX Programmer Database
- TwinCAT PLC Database

STEP1: SELECT THE SOURCE TYPE



Import Wizard – Selecting the Data Source Type

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

STEP 2: CONFIGURE THE SOURCE TYPE SETTINGS*Import Wizard – Selecting the Source Type Settings*

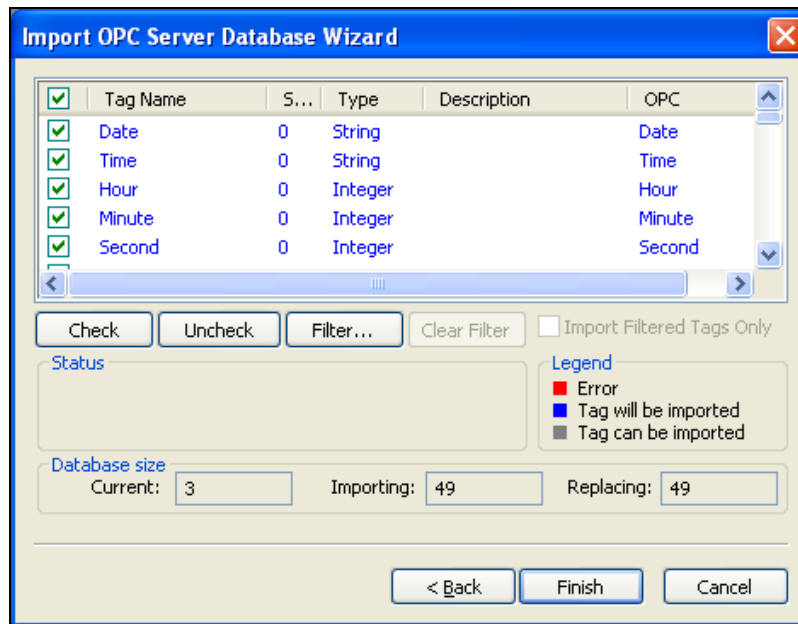
Most of the settings in the second window depend on the data Source Type selected in the first step. The screenshot above is an example of one data Source Type (OPC Server Database). The settings that are common for any data Source Type are described below:

- **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 application. 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:

To use the **Remote** option, InduSoft Web Studio must be running on the remote computer.

After configuring the settings in this dialog window, click the Next button.

STEP 3: FILTER THE TAGS

The screenshot above is an example of one data Source Type (OPC Server Database). The fields and settings that are common for all data Source Types include the following:

- **Grid:** Displays the list of tags found on the data source.

Field Name	Description
Check-box	Check to import the tag from the data source to the Tags Database of the current application.
Tag Name	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 window 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 check-box:** 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:

Color	Legend	Description
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:

Label	Description
Current	Indicates the number of tags configured in the Application Tags database of the current application
Importing	Indicates the number of tags selected to be imported
Replacing	Indicates the number of tags configured in the Application Tags database of the current application 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.

The other settings vary according to the data source selected in step one. They are described in the specific sections for each data source type that follow:

- Importing from Other InduSoft Web Studio 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 PanelMate Plus™ Databases
- Importing from TwinCAT PLC Databases

Importing from Other InduSoft Web Studio Databases

This wizard allows you to import the interfaces (tags and worksheets) of other IWS applications. When you import only the tags, rather than the whole application, 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 the runtime.

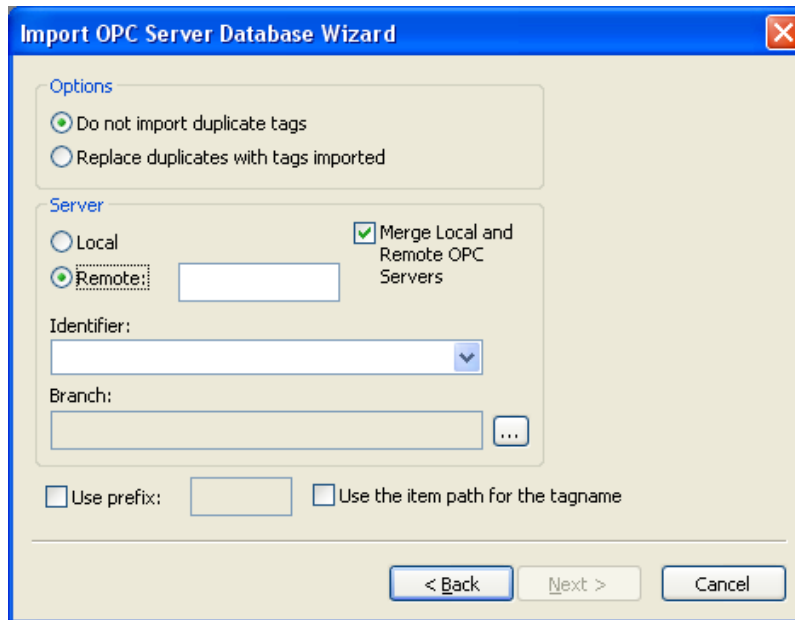
Import from InduSoft Web Studio Database Wizard

Field	Description
Import tags only	When this option is selected, the tags from the other application will be imported to the current application. The other interfaces of the application (worksheets) will not be imported.
Import the whole application	When this option is selected, the following interfaces from the other application will be imported to the current application: <ul style="list-style-type: none"> ▪ Tags Database ▪ Global Procedures ▪ Tags Database ▪ Global Procedures ▪ Screens ▪ Group Screens ▪ Web Pages ▪ Alarms ▪ Trend ▪ Recipes

	<ul style="list-style-type: none"> ▪ Reports ▪ ODBC ▪ Math ▪ Script ▪ Scheduler ▪ Drivers ▪ OPC ▪ TCP/IP ▪ DDE <p>This option is useful for merging applications and importing template application(s).</p> <p>Note: When selecting the option to Import the whole application, the following worksheets will be always imported, regardless of existing worksheets with the same number in the current application: ODBC, Math, Script, Scheduler, Drivers, OPC, TCP/IP and DDE. If there are worksheets with the same number in the current application, the worksheets imported from the other application will be inserted as additional worksheets in the current application (the number of each worksheet is automatically increased to avoid replacing files on the current application).</p>
Do not import duplicated	<p>When this option is selected, the following interfaces are not imported in case there is already an equivalent interface in the current application:</p> <ul style="list-style-type: none"> ▪ 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) ▪ Group Screens (group screens 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) ▪ Trends (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)
Replace duplicated with imported	<p>When this option is selected, the interfaces from the other application will be imported, and the interfaces from the current application with the same name (if any) will be overwritten.</p>
Local	<p>Select this option to import tags from an application stored in the local computer into the current application. To do so, click the Browse button and select the *.APP file of the application that has the tags you want to import.</p>
Remote	<p>Select this option to import tags from an application running in a remote station. Type the IP Address (or the host name) of the remote computer, in the Remote field. The tags from the current application from the remote computer will be available for import. This option is available for importing tags only, but it is not available for importing the whole application.</p>
Generate TCP/IP Client worksheet	<p>When you select Remote (instead of Local) and check this option, IWS will configure the TCP/IP Client worksheet automatically to exchange data with a remote application.</p>
Use Prefix	<p>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 suffix to differentiate the imported tags from the tags created manually.</p>

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:

Server	Description
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 check-box:** 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 check-box 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 check-box:** 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 () 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 in the section above, 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 table:

Property	Default Value
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 in the section above, 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 (Open DataBase Connectivity) 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:**

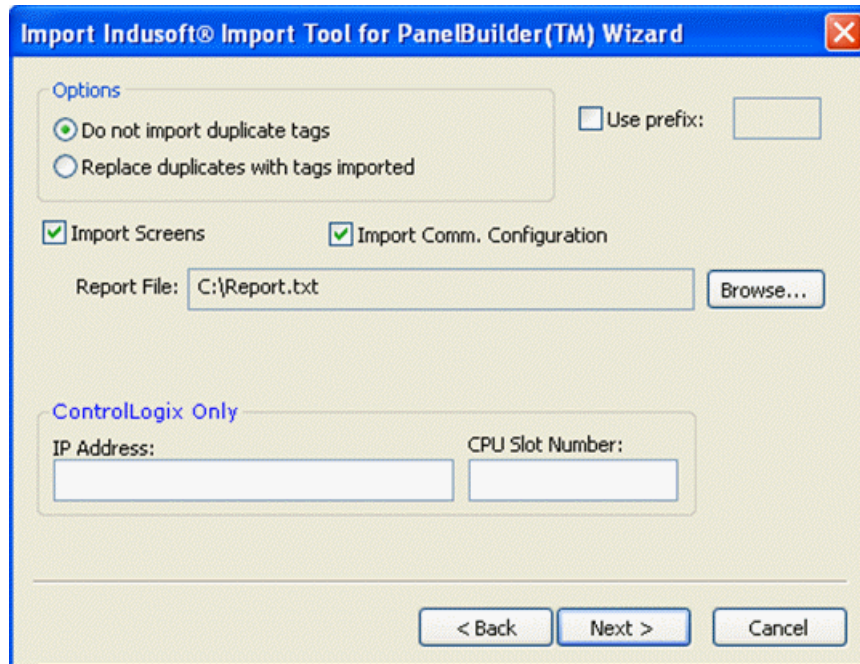
See Steps 1, 2 and 3 in the section above, for the settings and fields that are common for all Source Types.

Importing from PanelBuilder32™ Databases

 **Caution:**

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™ application (developed with PanelBuilder32™) into the IWS format and run them under any platform supported by IWS.



Import PanelBuilder™ Database Wizard

 **Note:**

See Steps 1, 2 and 3 in the section above, for the settings and fields that are common for all Source Types.

- **Import Screens:** Check this option to import the graphical screens (including their objects and dynamics) 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 an application 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 application. If the original application 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 application 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 application from the ***.PBA** format to the text (***.TXT**) format, using PanelBuilder32™, and import it into IWS.

**Note:**

IWS does not support some special characters (e.g., [] . -) in tag names. When you import your PanelBuilder 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.

Importing Data from the RSLogix 5000 CSV Database

- **PLC Options box:** Provides the following options:

Option	Description
Scope (Folder Name): <ul style="list-style-type: none"> ▪ Use Full Scope ▪ Do Not Use Scope ▪ Use Limited Scope 	<ul style="list-style-type: none"> ▪ Select Use Full Scope to import the tags using the full scope configured in the PLC program. ▪ Select Do Not Use Scope to ignore the scope of the tags configured in the PLC program. ▪ Select Use Limited Scope 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 check-box to create tags and classes for UDT tags imported from the PLC program. Uncheck this check-box 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:

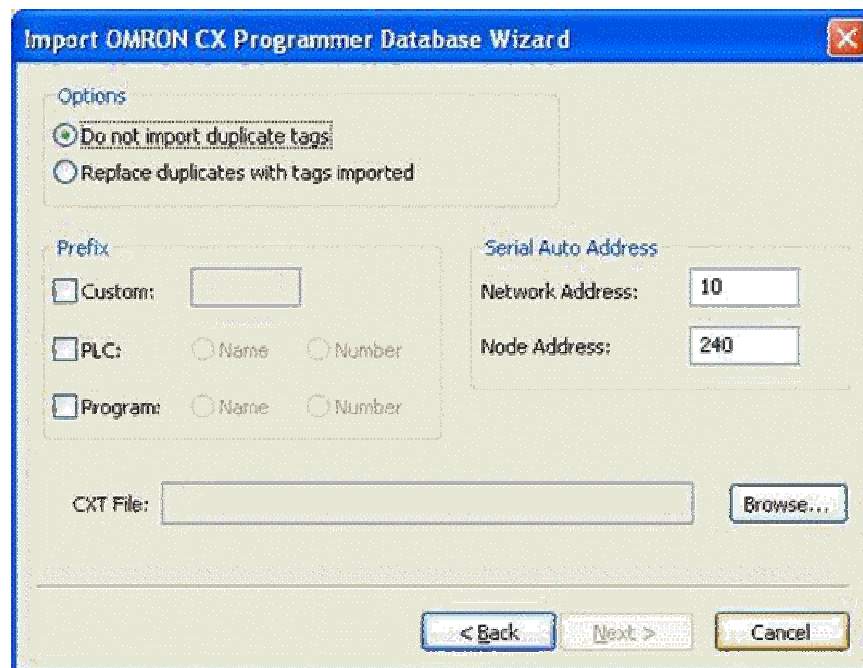
See Steps 1, 2 and 3 in the section above, for the settings and fields that are common for all Source Types.

Importing from OMRON CX Programmer Databases

Caution:

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.



Import OMRON CX Programmer Database Wizard

- **Prefix:** This box allows you to concatenate one of the following types of prefixes to the tags imported from the CX Programmer program:

Prefix	Description
--------	-------------

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 box allows you to configure the Network Address and the Initial Node Address for the PLCs configured in the product with Serial communication (if any):

Setting	Description
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:**

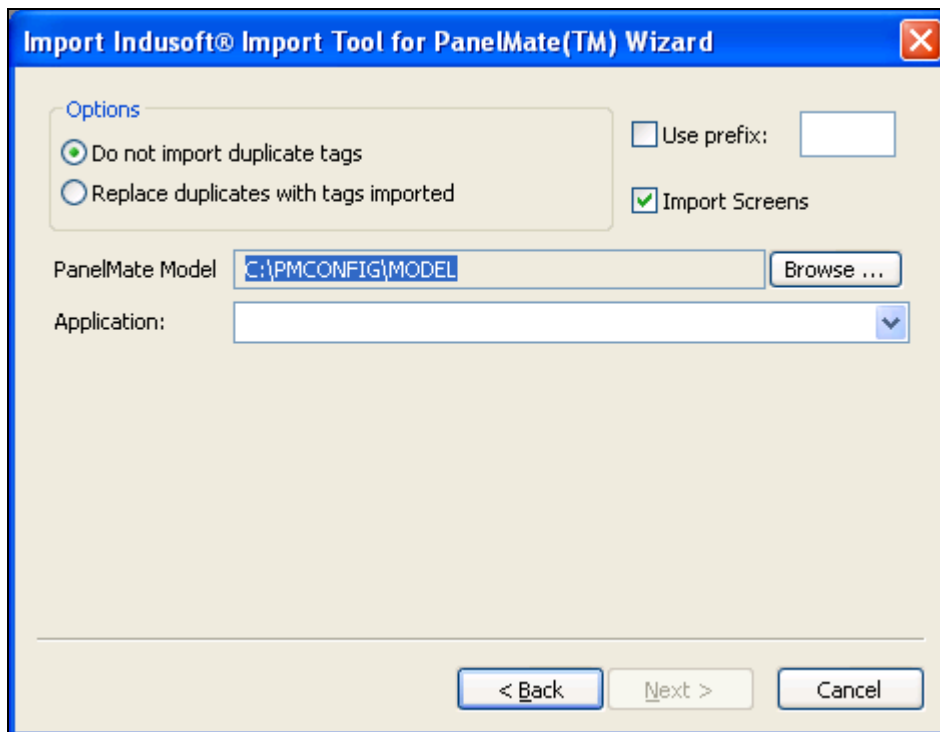
See Steps 1, 2 and 3 in the section above, for the settings and fields that are common for all Source Types.

Importing from PanelMate Plus™ Databases

⚠ Caution:

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 project created with PanelMate Plus™ software. Using this wizard, you can convert a PanelMate™ application (developed with PanelMate Plus™) into the IWS format and run it under any platform supported by IWS.



Import PanelMate Database Wizard

- **Import Screens:** Check this option to import the graphical screens (including their objects and dynamics) 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 applications available in this directory will be available in this combo-box. Select the application 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 application from the PanelMate Plus™ software into IWS.

Note:

See Steps 1, 2 and 3 in the section above, 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 application 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.

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.

EXPORTING THE SYMBOL FILE FROM TWINCAT

The TwinCAT development software automatically exports the application database to a “symbol file” every time you rebuild your TwinCAT application 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 POUs and Global Variables and then rebuild your TwinCAT application project to generate a fresh symbol file.

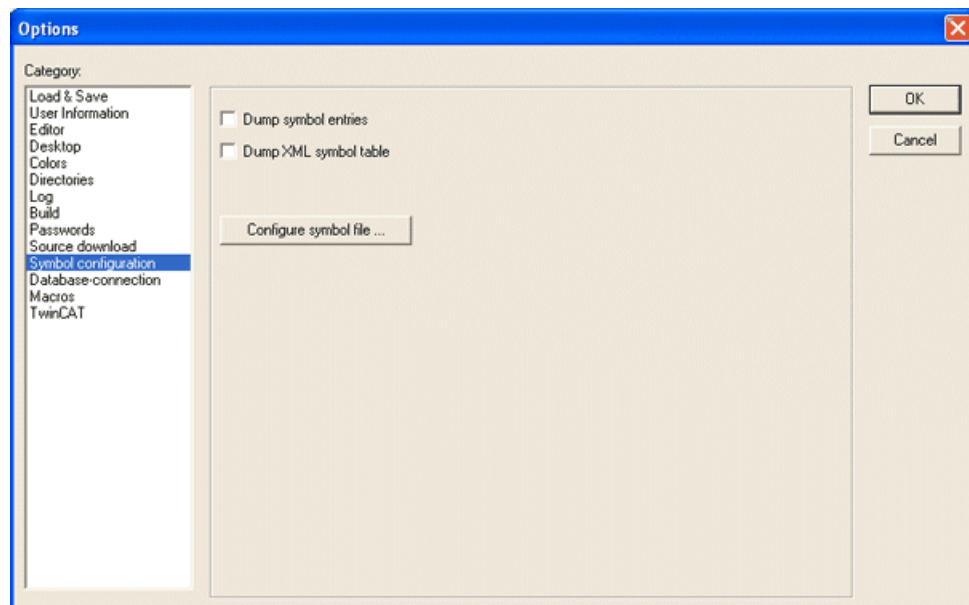
 **Note:**

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:

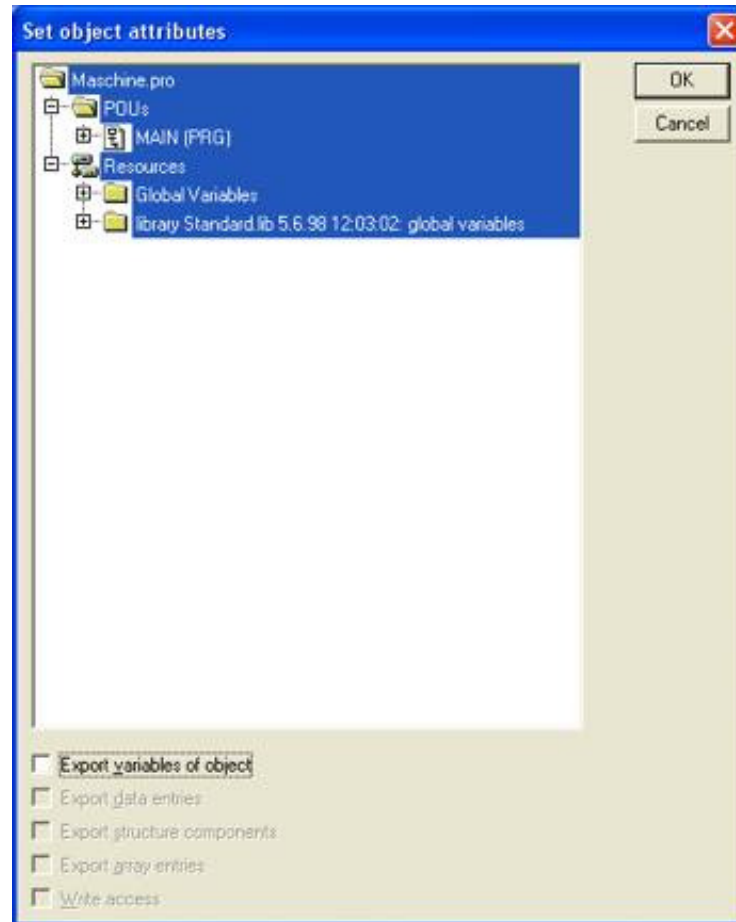
1. Open your TwinCAT application project using the TwinCAT development software.
2. Choose **Project** → **Options** from the menu bar. The *Options* window is displayed.
3. Select **Symbol configuration** from the **Category** list:



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.

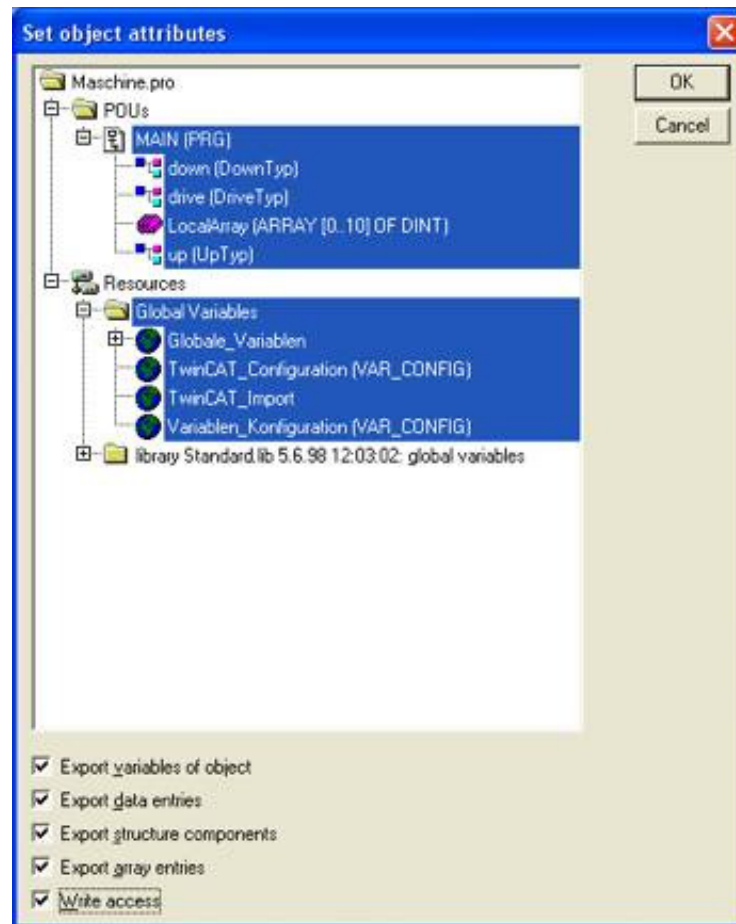
- For the sake of expediency, you should first disable the export of all objects and then reenable only the objects that you want to export to IWS — typically, the POU's and Global Variables. Select all of the objects in the tree and uncheck all options for them at the bottom of the dialog. For example:



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 POUs 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 application project, generating a symbol file that contains the desired POUs and Global Variables.

You are now ready to import the variable names into InduSoft Web Studio.

Development Modules

This section discusses an overview of graphics and tasks development modules. For more information, see *Chapter 7: Configuring Screens and Graphics* and *Chapter 8: Configuring Task Worksheets*.

Graphics

The most basic function performed by InduSoft Web Studio 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*).

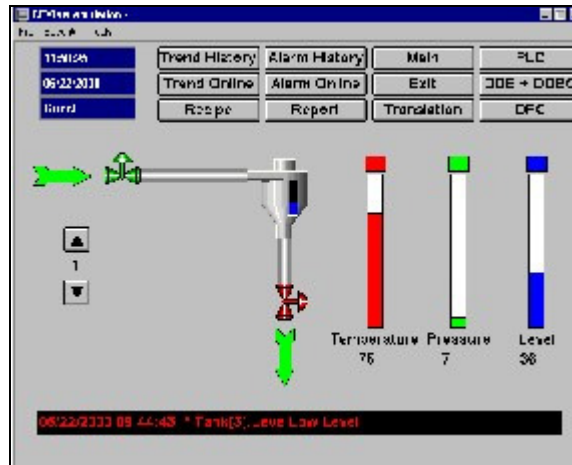
InduSoft Web Studio allows you to create applications that can monitor processes using high-resolution color screens.

The InduSoft Web Studio graphic tools consist of two modules:

- The *Screen/Worksheet Editor* on the InduSoft Web Studio desktop (used to create or import graphics)
- The application runtime *Viewer*

You can use *animation links* to create dynamic graphic objects or symbols. Animation links 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 dynamic 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 dynamic object layer can reflect the changes in the plant, giving the illusion that the screen is three-dimensional.










Sample CEView Emulation Screen

All InduSoft Web Studio configuration tasks require a Windows-compatible pointing device, such as a mouse or touch pad. You can run an application in the Viewer without a pointing device if you configure keypad or keyboard keys for all commands.

Tasks

You use the IWS **Tasks** tab to configure task-specific worksheets, each composed of a *Header* (where you define global information for the worksheet) and a *Body* (where you configure the tags and expressions used in each task).

You can configure the following task-specific worksheets:

- **Alarm Groups** : Use to define an alarm group, its characteristics, and its messages that are reported in alarm conditions. The main purpose of the alarms is to inform the operators about any problem or change of state during the process so that corrective action can be taken.
To show alarm messages on the screen, you must create the alarm object on the screen.
- **Trend Groups** : Use to define trend groups, which keep track of process variables behavior. You can store samples in a history file and show both history and online samples in a screen trend graph.
- **Recipes** : Use to read and write ASCII files from and to the hard disk, and to transfer values between files and real-time memory. Typically, you use this module to store process recipes, but these files can store any type of information such as operation logs, passwords, and so forth. You also can use this module to store data in XML format.
- **Reports** : Use to configure your own reports with system data, in either ASCII or RTF format. The main purpose of this module is to make report creation easier and more efficient.
- **Math Worksheets** : Use to implement additional routines to work with the basic functions of the InduSoft Web Studio modules. A *Math Worksheet* is a group of programming lines that are executed as one of the application *Background Tasks*. You can configure the mathematics in blocks in different worksheets.
This worksheet provides a free environment for logical routines and mathematical calculations that the project may need. For these purposes, the scripting language is simple and easy to use.
- **Scheduler** : Use to generate the time bases used in an application. The Scheduler is capable of triggering events.
- **ODBC Configuration** : Use to enable InduSoft Web Studio applications to access any database that is compatible with the ODBC protocol (such as Access, Excel, Oracle, SQL Server and so on).

 **Note:**

The ODBC interface is not available for WinCE applications.

General Communications

You can enable InduSoft Web Studio applications to communicate (exchange data values) with other applications, remote devices (such as a PLC or transmitters), and any devices that implement OPC or DDE Servers.

To enable communication, you configure the *task worksheets* provided by IWS. Instructions for configuring these worksheets are provided in *Chapter 10: Communication* in the following sections:

- **Configuring a Driver:** Explains how to configure a *Driver* worksheet to implement a communication protocol (OPC, TCP/IP, or DDE).
- **Configuring OPC:** Explains how to configure an OPC worksheet to manage communication between local or remote OPC Clients and Servers.
- **Configuring TCP/IP:** Explains how to configure a TCP/IP worksheet to manage communication between two IWS applications.
- **Configuring DDE:** Explains how to configure a DDE worksheet to manage communication between local or remote DDE Clients and Servers.

Use the **Comm** tab to access all worksheets configured to establish communication with another device or software using available protocols.

Chapter 4: Understanding IWS Structure

This chapter explains the internal structure of IWS, including how data flows through the IWS runtime environment and how the various runtime modules are executed.

🔴 IMPORTANT!

To avoid unexpected behaviors and guarantee the best performance when executing applications, we strongly recommend that you read and understand the information in this chapter before you start developing complex applications.

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 application screens and updates the screen objects.

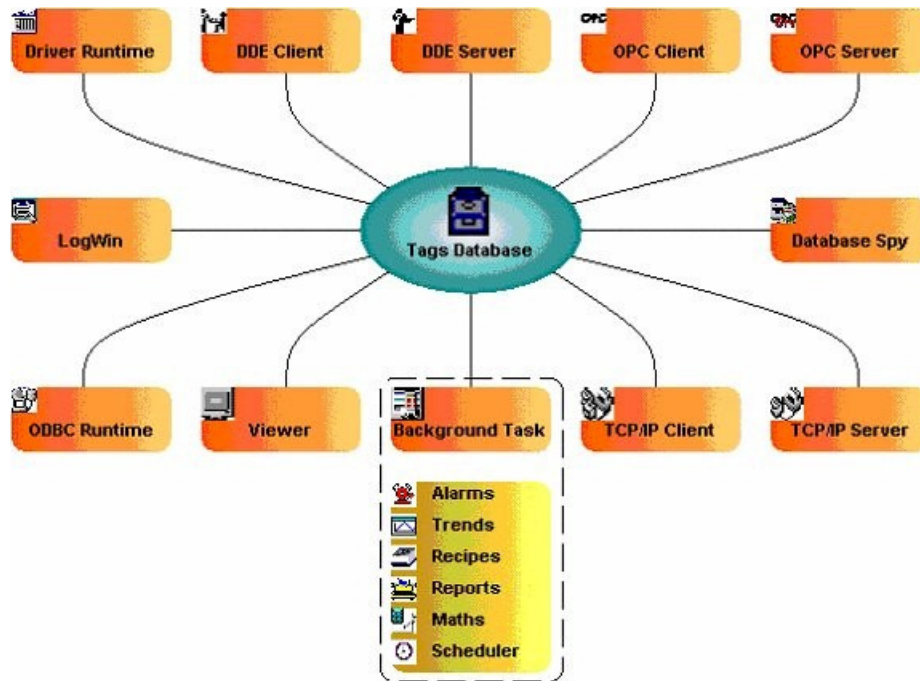
None of the preceding runtime modules exchange data directly with another module or task. Instead, runtime modules send data to and receive data from the *Tags* database, which is the “heart” of IWS.

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.

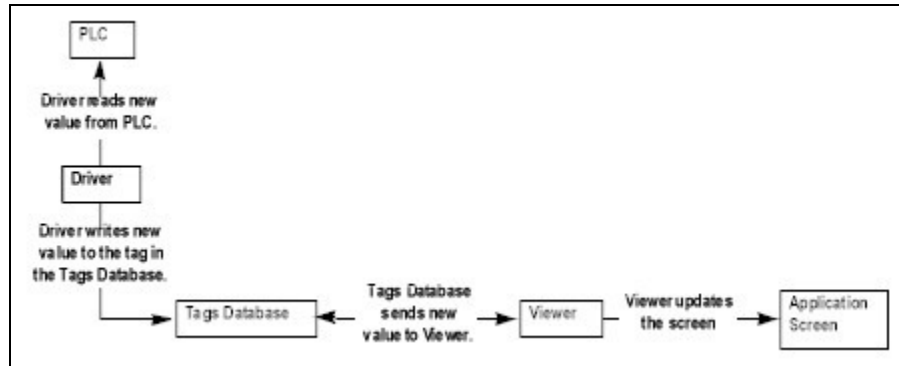
For detailed information about tags, tag values, and tag properties see *Chapter 5: Working with Tags*.



IWS 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 application 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* (programmable logic controller), the driver updates the tag associated with this value in the *Tags* database. Then, if this new information must display on the application 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.

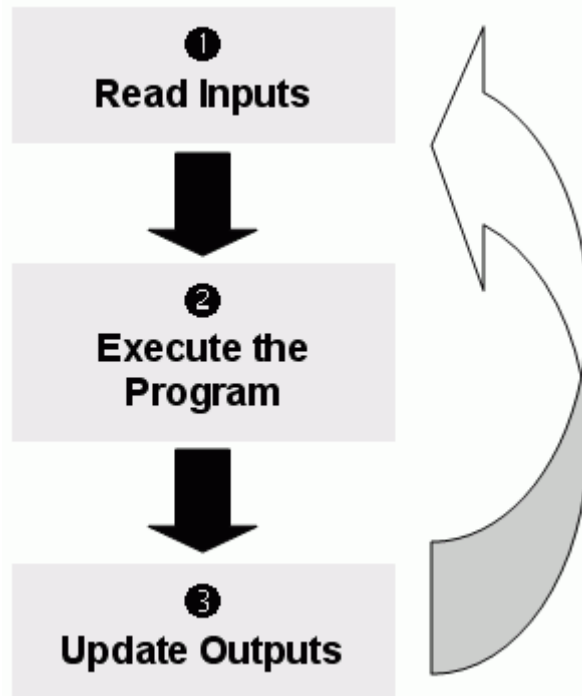
If you configure a dynamic object (such as **Text I/O**) with a function that does not require a tag (for example, **NoInputTime()**), the *Viewer* will not update the object because there is are no tags associated with that object.

IWS's architecture 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 IWS 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:



PLC Program Loop

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

IWS only has one process—**Studio Manager.exe**. When you execute a runtime application, the **Studio Manager.exe** process starts the *Tags* database and all of the runtime modules configured for the application. You can specify which modules (such as *Viewer* and *Driver*) will start during the runtime (see “Starting Runtime Modules on the Target System” on page 6–31).

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:

- ✓ From the IWS installation directory (for example, **C:\Program Files\<Installation Folder>\bin**), double-click **\BIN** to open the folder.
- ✓ 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).

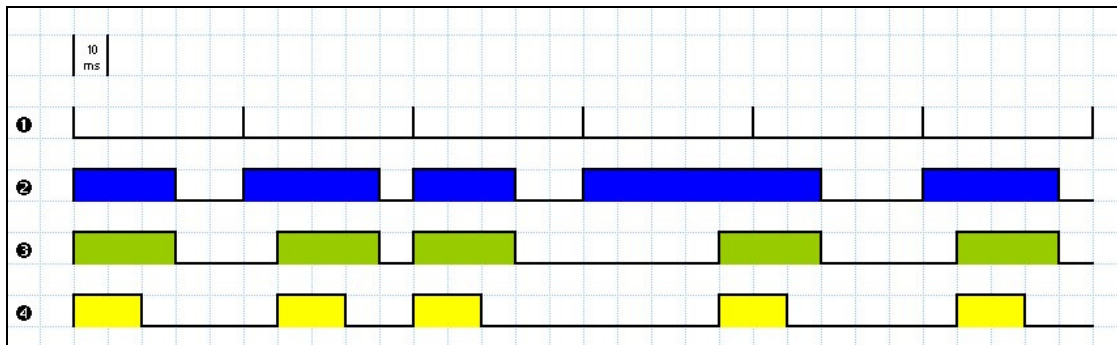
```
[Period]
DBSpy=1000
UniDDEClient=200
UniDDE=200
Driver=20
LogWin=100
UniODBCRT=100
OPCClient=20
OPCServer=20
TCPClient=100
TCPServer=100
Viewer=50

[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.
- ✓ 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 ❶ is the **Period** time period (set to 50ms for this example).
- Signal ❷ shows when the thread is active for the operating system.
- Signal ❸ is the **TimeSlice** time period (set to 30ms for this example).
- Signal ❹ shows the execution of the thread itself.

In this example, IWS generates a **Period** message every 50ms (signal ❶). 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 ❶).

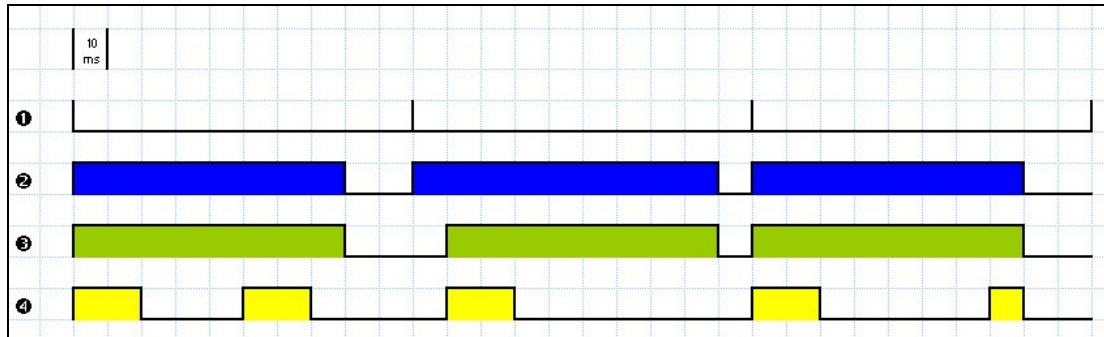
While the thread is active, the operating system is responsible for executing that thread. However, just because a thread is active does not mean the operating system will execute it immediately – the operating system may be executing other threads, for example.

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.

- Signal ❶ is the **Period** time period (set to 100ms).
- Signal ❷ shows when the thread is active for the operating system.
- Signal ❸ is the IWS **TimeSlice** time period (set to 80ms).
- Signal ❹ shows the execution of the thread itself.



Setting a Higher TimeSlice

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 an IWS **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 **C:\Documents and Settings*<user>*\Local Settings\Application Data\InduSoft Web Studio v6.1** folder (see “Executing/Switching IWS Modules” on page 4–3).

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** commands when the **Recipe ()** or **Report ()** functions are executed. 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** dynamic).

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:

- ✓ 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.
- ✓ 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.
- ✓ 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.

Chapter 5: Working with Tags

As stated in *Chapter 4*, the *Tags* database is the heart of IWS because it manages the flow of data between runtime modules and it stores all tags, tag values, and tag properties. The *Tags* database is the medium used by all modules to read or write tag values.

This chapter explains basic concepts about the tags, tag values, and tag properties used in the IWS *Tags* database—including how to create and edit tags for your projects.

Note:

We recommend that you read and understand the concepts discussed in *Chapter 4: Understanding the IWS Structure*, before you read this chapter.

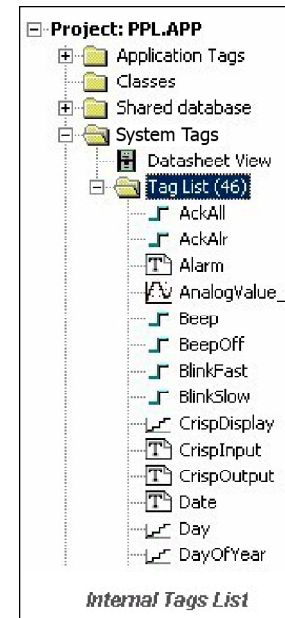
What is a Tag?

Tags are variables that are designed to receive results from expressions specified in screens and worksheets (such as communication points in field equipment, calculation results, alarm points, and so forth).

All tags are organized into one of the following categories (which are represented by folders on the **Database** tab in the *Workspace*):

- **Application Tags** are tags you create during application development.
For example:
 - Screen tags
 - Tags that read from/write to field equipment
 - Control tags
 - Auxiliary tags used to perform mathematical calculations
- **Classes** are compound tags that you create to associate a set of values (rather than a single value) with a class. Class structures permit a high-degree of encapsulation within the *Tags* database.
- **Shared Database** tags are created in a PC-based control software program and then imported into the IWS *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.



Note:

Because you must configure a project before you can share tags, instructions for this procedure are provided in *Chapter 6: Creating and Configuring a Project*. See “Sharing PC-Based Control Software Program Database Tags” on page 6–4.

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 internal 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 **Database** tab in the *Workspace*, 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 application, and you can use the same tag for more than one object or attribute.

Designing a Tag

Before you can create a tag for its intended purpose, you must understand what *types* of tags you can create and the basic elements or characteristics of a tag, including:

- Tag data types
- Shared or not
- Tag properties

After reading the information provided in this section, you should be able to create tags to properly suit your application needs.

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 an IWS application are basic tags. Some examples of a basic tag include:

- **TankID** (to identify different tanks in your application)
- **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> [<anothertag>+c]`

Where:

- `<ArrayTagName>` is the tag name
- `[<ArrayIndex>]` is the unique index (fixed value or another tag)
- `+` is any arithmetic operation
- `c` is a numerical constant

Notes:

- You must specify a maximum index for each array tag by typing a value (*n*) in the **Array Size** column of an *Application Tags* datasheet or in the **Array Size** field on a *New Tag* dialog. (See “Creating Application Database Tags” on page 5–13). 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 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.

Array Tag Examples

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]**).

CLASS TAGS

Class tags are compound tags that permit a high-degree of encapsulation within the *Tags* database. Where basic tags receive a single value, *classes* are designed to receive multiple values.

You can create a class-type tag by grouping basic or array tags, which then become the class *members*. The maximum number of members for any class depends on the product specification.

You specify class-type tags in one of two formats:

- For a simple class tag the syntax is **<TagName> . <ClassMemberName>**. (Where the period is used as a separator.)

For example, if you wanted to monitor several different conditions (such as *temperature*, *level* and *pressure*) in a tank, you might create a class tag as follows:

Tank . Temperature

Tank . Level

Tank . Pressure

- For creating a complex class tag (using an array tag) the syntax is **<ArrayTagName> [<ArrayIndex>] . <ClassMemberName>**. (Where again, the period is used as a separator.)

If you wanted to monitor the *temperature*, *level*, and *pressure* conditions in multiple tanks, you might create a class tag as follows:

```
Tank [tk] .Temperature
Tank [tk] .Level
Tank [tk] .Pressure
```

Where **tk** is an array index, representing the tank number.

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** tag.

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 a 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*): Boolean or digital variables (0 or 1). Typically used for turning objects off and on or for closing and opening objects.
- **Integer** (four bytes): Integer number (positive, negative, or zero). Equivalent to the C-type, signed long integer (ranging from -2147483648 to 2147483647). Typically used for counting whole numbers or setting whole number values.
- **Real** (floating point, eight bytes): Real number that is stored internally as a double word. Equivalent to a C-type double. Typically used for measurements or for decimal or fractional values.
- **String** (alphanumeric data, up to 1024 characters): Character string that holds letters, numbers, or special characters. Supports both ASCII and UNICODE characters.

For example: **Recipe product X123, 01/01/90, *** On *****

CHANGING HOW BOOLEAN TAGS RECEIVE NUMERIC VALUES

By default, if any numeric value other than 0 (i.e. ≠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 **<Application Name>.app** file to change the following setting:

```
[Options]
BooleanTrueAboveZero=<value>
```

If **BooleanTrueAboveZero** is set to the default 0, then the application will behave as described above. If **BooleanTrueAboveZero** is set to 1, then the application 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.

Sharing Tags with a Web Thin Client

IWS allows you to decide whether a tag value can be shared with (displayed on) a Web Thin Client station.

- **Local:** Prevents a tag value on the Server station from being shared with Web Thin Client stations.
- **Server:** Enables a tag value to display on an HTML page, which makes the value available to a Web Thin Client.

If you want to view and manage your applications remotely, you must be sure to specify the **Server** option for all tags related to the management process. For example, if you want the ability to respond to alarms remotely, you must enable the **Server** option for all tags related to alarm management.

Choosing Tag Properties

Tag properties are attributes (or *parameters*) that are inherent to a database tag (such as minimum/maximum values, size, description, and so forth).

When you define tags for an IWS application, you can specify tag properties that are used during runtime as *tag fields*. These runtime tag fields are described in the following table.

Tag Field Name	Description of Value Associated with Each Field	Tag Type Associated with Field				R/RW
		Boolean	Integer	Real	String	
Name	The name of the tag, as configured in the Application Tags database.	✓	✓	✓	✓	R
MemberName	The name of the class member, in a properly configured Class. NOTE: The syntax must be: <Class> . <Member> -> MemberName Example: Tank . Lvl -> MembeName = Lvl	✓	✓	✓	✓	R
Size	Array Size. If the tag is not an array tag, it returns the value 0.	✓	✓	✓	✓	R

Tag Field Name	Description of Value Associated with Each Field	Tag Type Associated with Field				R/RW
		Boolean	Integer	Real	String	
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
Description	Description of tag (in Tags Database)	✓	✓	✓	✓	RW
Quality	Tag quality (192=GOOD; 0=BAD). Field updates every time tag receives expression results or a communication task value (Driver or OPC). Invalid expressions (such as division by zero) or reading communication errors associated with tag will set the tag quality to BAD automatically.	✓	✓	✓	✓	R
TimeStamp	Time & date when tag changes value.	✓	✓	✓	✓	R
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, psi.	✓	✓	✓	✓	RW
Max	Maximum value that can be written to the tag during runtime.	✗	✓	✓	✗	RW
Min	Minimum value that can be written to the tag during runtime	✗	✓	✓	✗	RW
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 Farenheit degrees. If user input changes DisplayValue during runtime, then the conversion is reversed before the change is actually written to the Tag: Value = (DisplayValue – UnitAdd) * UnitDiv	✗	✓	✓	✗	RW

Tag Field Name	Description of Value Associated with Each Field	Tag Type Associated with Field				R/RW
		Boolean	Integer	Real	String	
DisplayUnit	A brief description (up to 9 characters) of the Engineering Unit for DisplayValue. NOTE: This field can only be set by using the SetDisplayUnit () and SetTagDisplayUnit () functions.	x	✓	✓	x	R
UnitDiv	Number by which the Tag value is divided to get DisplayValue. To perform no division, UnitDiv should be 1. NOTE: This field can only be set by using the SetDisplayUnit () and SetTagDisplayUnit () functions.	x	✓	✓	x	R
UnitAdd	Number added to the Tag value to get DisplayValue. To perform no addition, UnitAdd should be 0. NOTE: This field can only be set by using the SetDisplayUnit () and SetTagDisplayUnit () functions.	x	✓	✓	x	R
DisplayMax	The maximum value that can be input to DisplayValue during runtime: $\text{DisplayMax} = (\text{Max} / \text{UnitDiv}) + \text{UnitAdd}$ If DisplayMax is changed during runtime, then Max is also changed as follows: $\text{Max} = (\text{DisplayMax} - \text{UnitAdd}) * \text{UnitDiv}$	x	✓	✓	x	RW
DisplayMin	The minimum value that can be input to DisplayValue during runtime: $\text{DisplayMin} = (\text{Min} / \text{UnitDiv}) + \text{UnitAdd}$ If DisplayMin is changed during runtime, then Min is also changed as follows: $\text{Min} = (\text{DisplayMin} - \text{UnitAdd}) * \text{UnitDiv}$	x	✓	✓	x	RW
B0 ... B31	Value (0 or 1) of any of the 32 bits (b0, b1, b2, ... b31) of an integer tag. (B0: LSB B31: MSB).	x	✓	x	x	RW

Tag Field Name	Description of Value Associated with Each Field	Tag Type Associated with Field				R/RW
		Boolean	Integer	Real	String	
AlrStatus	<p>Integer value containing the status of all currently active alarms associated with the tag. Each bit of this integer value indicates a specific status, as follows:</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: DevM Alarm active 	✓	✓	✓	✗	R
<p style="text-align: center;">For example:</p> <p style="text-align: center;">If Tag->AlrStatus returns value 2, "Hi" alarm is active.</p> <p style="text-align: center;">If Tag->AlrStatus returns value 3, "HiHi" and "Hi" alarms are active simultaneously.</p> <p style="text-align: center;">If Tag->AlrStatus returns value 0, there are no active alarms associated with this tag.</p> <p style="text-align: center;">For Boolean tags, only 1 (bit 1), 4 (bit 2) or 16 (bit 4) values are returned.</p>						

Field Name	Description of Value Associated with Each Field	Tag Type Associated with Field				R=Read Only RW=Read+Write
		Boolean	Integer	Real	String	
Ack	<p>This field can have two values:</p> <ul style="list-style-type: none"> • 0: No alarms associated with this tag require acknowledgment. • 1: At least one alarm associated with this tag requires acknowledgment. <p>NOTE: This works as a global acknowledge for the tag and goes to 0 only when all alarms for the tag have been acknowledged.</p>	✓	✓	✓	✗	RW
AlrDisable	<p>This field can have two values:</p> <ul style="list-style-type: none"> • 0: Enables alarm associated with tag. When an alarm condition occurs, the alarm becomes active. • 1: Disables alarm associated with tag. If alarm condition occurs, alarm will not become active. 	✓	✓	✓	✗	RW
HiHi	<ul style="list-style-type: none"> • If 0, HiHi alarm is inactive. • If 1, HiHi alarm is active. 	✗	✓	✓	✗	R
Hi	<ul style="list-style-type: none"> • If 0, Hi alarm is inactive. • If 1, Hi alarm is active. 	✓	✓	✓	✗	R
Lo	<ul style="list-style-type: none"> • If 0, Lo alarm is inactive. • If 1, the Lo alarm is active. 	✓	✓	✓	✗	R
LoLo	<ul style="list-style-type: none"> • If 0, LoLo alarm is inactive. • If 1, the LoLo alarm is active. 	✗	✓	✓	✗	R
Rate	<ul style="list-style-type: none"> • If 0, Rate alarm is inactive. • If 1, the Rate alarm is active. 	✓	✓	✓	✗	R
DevP	<ul style="list-style-type: none"> • If 0, DevP alarm is inactive. • If 1, the DevP alarm is active. 	✗	✓	✓	✗	R
DevM	<ul style="list-style-type: none"> • If 0, DevM alarm is inactive. • If 1, DevM alarm is active. 	✗	✓	✓	✗	R
HiHiLimit	Limit value for HiHi alarm.	✗	✓	✓	✗	RW
HiLimit	Limit value for Hi alarm.	✗	✓	✓	✗	RW
LoLimit	Limit value for Lo alarm.	✗	✓	✓	✗	RW

Field Name	Description of Value Associated with Each Field	Tag Type Associated with Field				R=Read Only RW=Read+Write
		Boolean	Integer	Real	String	
LoLoLimit	Limit value for LoLo alarm.	x	✓	✓	x	RW
RateLimit	Limit value for Rate alarm.	x	✓	✓	x	RW
DevSetpoint	Set point value for DevP and DevM alarms.	x	✓	✓	x	RW
DevPLimit	Limit value for DevP alarm.	x	✓	✓	x	RW
DevMLimit	Limit value for DevM alarm.	x	✓	✓	x	RW

Tag Fields Table

To access a tag field at runtime, type the tag name and field (using the syntax: **TagName->Field**) into the *Database Spy*, a command window, or an execution field.

⚠ Cautions:

- You cannot use tag fields (such as **Bit** fields) to configure *Alarm* or *Trend* worksheets.
- Although you can apply tag properties to system tags, those properties will not persist if you download your application to a CE device.

📘 Note:

If an application tries to write a value to the *Tags* database that falls outside the range specified in the **Min** and **Max** fields, the database will not accept the new value and will write a warning message to the *IWS Output* window.

If you specify a zero (0) for the **Min** and **Max** fields, the application can write any tag type value to the *Tags* database.

Creating Application Database Tags

This section explains the process for creating basic, array, and pointer tags for your project applications.

Notes:

When specifying a tag name, you must adhere to the following guidelines:

- 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*). Some other example tag names include: *Temperature*, *pressure1*, *count*.

To save development time, IWS allows you to create and edit tags from a variety of locations within the development environment.

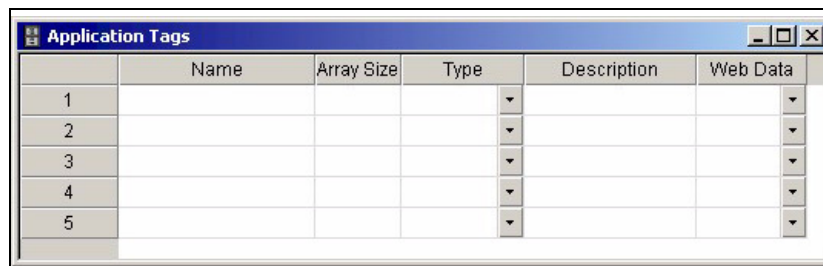
- You can add tags using the *Application Tags* datasheet
- You can add tags “on-the-fly” from a pop-up menu or from any **Tag** or **Expression** text box (located on *Object Properties* dialogs, worksheets, and so forth)

Instructions for adding tags with each method are provided in the next two sections.

Adding Tags to the Application Datasheet

Use the following steps to create tags from the *Application Tags* datasheet:

- ✓ Select the **Database** tab and open the *Application Tags* folder.
- ✓ Double-click the **Datasheet View** icon  to open the *Application Tags* datasheet:



	Name	Array Size	Type	Description	Web Data
1					
2					
3					
4					
5					

Application Tags Datasheet

- ✓ Locate an empty line in the datasheet and configure the following fields.
*(Hint: You can use the keyboard **Tab** key to move to the next column.)*

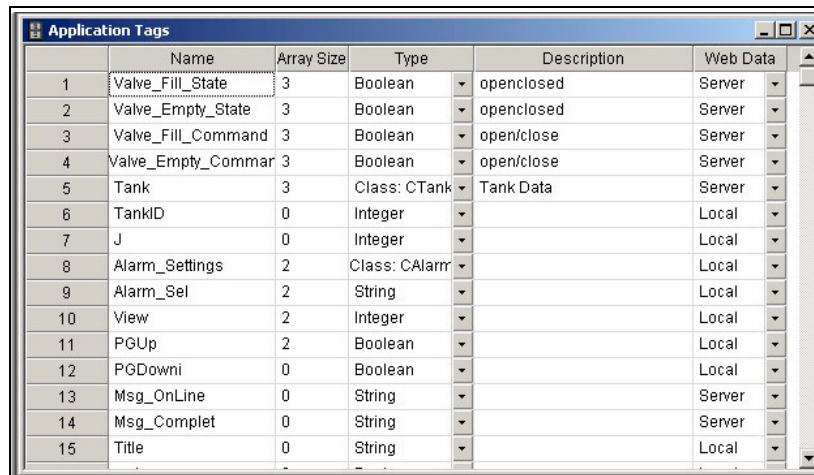
- **Name** field: Type a name using the syntax described for the tag type you are creating (see “Creating Application Database Tags”) that adheres to the guidelines described on page 5–13.
- **Array Size** field:
For an *array* tag, type a value to specify the maximum index of the array.
For any other 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” on page 5–6.)
- **Description** field (*optional*): Type a description for documentation purposes only.
- **Web Data** combo-box: Click the arrow to specify whether the tag value will be shared with (displayed on) Web Thin Client stations.

Local: Prevents a Server tag value from being shared with Web Thin Client stations. For example, you want to view a tag value on your workstation and not interfere with the value being displayed on other Web Thin Client workstations.

Server: Shares a Server tag value with Web Thin Client stations. For example, you want other Web Thin Client workstations to share/view control values (such as **On** or **Off**).

- ☑ Click in a new line to create another tag or, if you have no other tags to create, save your tag(s) to the *Tags* database (**File** → **Save**) and close the *Application Tags* datasheet.

The following example shows a variety of tags configured in an *Application Tags* datasheet.



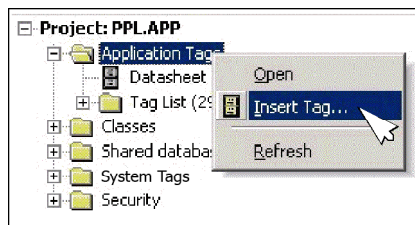
	Name	Array Size	Type	Description	Web Data
1	Valve_Fill_State	3	Boolean	openclosed	Server
2	Valve_Empty_State	3	Boolean	openclosed	Server
3	Valve_Fill_Command	3	Boolean	open/close	Server
4	Valve_Empty_Command	3	Boolean	open/close	Server
5	Tank	3	Class: CTank	Tank Data	Server
6	TankID	0	Integer		Local
7	J	0	Integer		Local
8	Alarm_Settings	2	Class: CAlarm		Local
9	Alarm_Sel	2	String		Local
10	View	2	Integer		Local
11	PGUp	2	Boolean		Local
12	PGDown	0	Boolean		Local
13	Msg_OnLine	0	String		Server
14	Msg_Complet	0	String		Server
15	Title	0	String		Local

Example Application Tags Datasheet

Adding Tags “On-the-Fly”

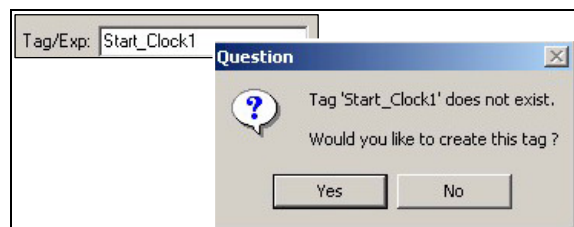
As an alternative to opening the *Application Tags* datasheet every time you want to create a new tag, IWS enables you to create individual tags “on-the-fly” by:

- Right-clicking on the *Applications Tags* folder, the **Datasheet View** icon, or *Tag List* subfolder and selecting **Insert Tag** from the pop-up menu.



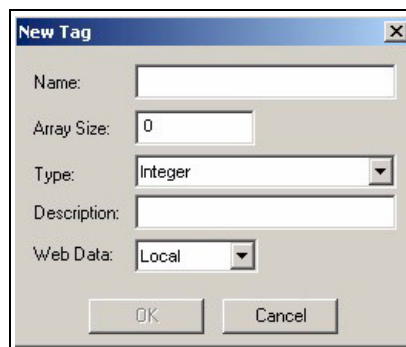
Inserting a Tag

- Typing a new tag name into any **Tag/Exp** text box (available from *Object Properties* dialogs, worksheets, and so forth). When a *Question* dialog asks if you want to create a new tag, click **Yes**.



Creating a New Tag

Using either of the preceding methods causes a *New Tag* dialog to display:



New Tag Dialog

The fields and combo-boxes on this dialog correspond in name and function to the columns on the *Application Tags* datasheet. Consequently, you can configure this dialog using the instructions provided on page 5–13. When you are finished, click **OK** to save the tag to the *Tags* database and close the *New Tag* dialog.

Resetting the Tags Database


Select **Tools > Reset Tags Database** to “reload” the tags database on the local station. This command affects all tags stored in the *Application Tags* folder. This option is useful for resetting the application tags and restoring the values they had when the application was loaded for the first time. When you stop the application but leave the development environment open, the tags are not reset by default when the application is started again. Therefore, you can execute this command to reset them before the application 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 (**Project > Stop application**) 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 application tags automatically whenever you run the application (**Project > Run Application**), you can check the option **Reset Tags Database when starting application** on the **Preferences** tab of the *Project Settings* dialog.

Creating Classes

To create a new class tag:

- ✓ From the **Database** tab, right-click the *Classes* folder, and select **Insert Class** from the resulting pop-up menu.
- ✓ When the *Insert Class* dialog displays, type a name into the **Name** field using the syntax on page 5–5 and the guidelines described on page 5–13.



Insert Class Dialog

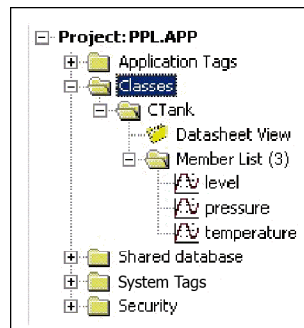
- ✓ Click **OK** to close the *Insert Class* dialog and the *Class: worksheet* displays automatically.
- ✓ Configure the columns in this worksheet as follows:
 - **Name** field: Type a class member name (see page 5–5).
 - **Type** combo-box: Click the arrow to select the class member’s data type (**Boolean**, **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

- ✓ 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

- ☑ Next, use the instructions provided in “Creating Application Database Tags” on page 5–13, 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 (openclosed)	Server
2	Valve_Empty_State	3	Boolean	Empty valve state (openclosed)	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

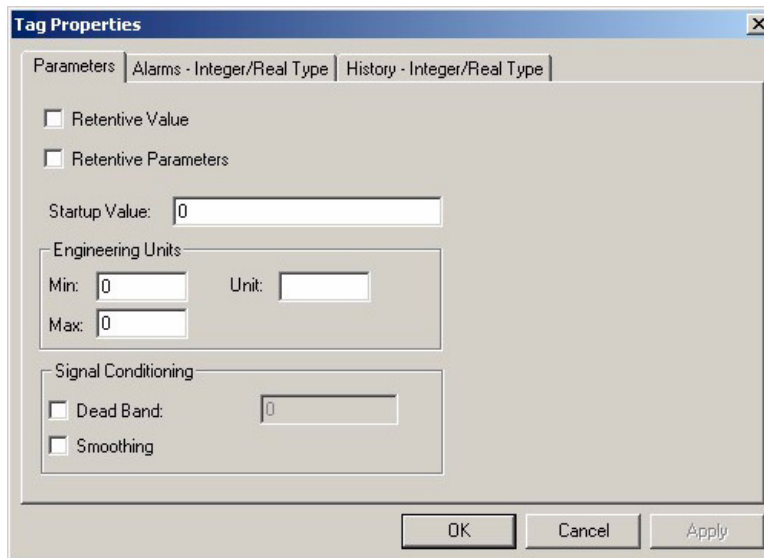
Creating the Tank Class Tag

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

Setting Tag Properties

This section explains how to configure tag properties.

- ✓ From the **Database** tab, open the *Application Tags* datasheet (or display the tag list in the *Workspace*) and highlight the tag for which you want to set properties.
 - ✓ Click the **Tags Properties** button  on the *Tags Properties* toolbar.
- A *Tag Properties* dialog displays similar to the following:



Tag Properties Dialog

A *Tag Properties* dialog can consist of one, two, or three tabs depending on the type of tag you selected in the datasheet:

- **Parameters:** Use this tab to configure retention, startup, and unit properties.
- **Alarms - Type:** Use this tab to enable, configure, and view alarm properties.
- **History - Type:** Use this tab to enable, configure, and view historical properties.

 **Note:**

Not all properties are available for all tag types. Consequently, the features on each tab will vary, depending on the tag type you selected. For example, **Minimum** and **Maximum** values are not necessary for *String* tags.

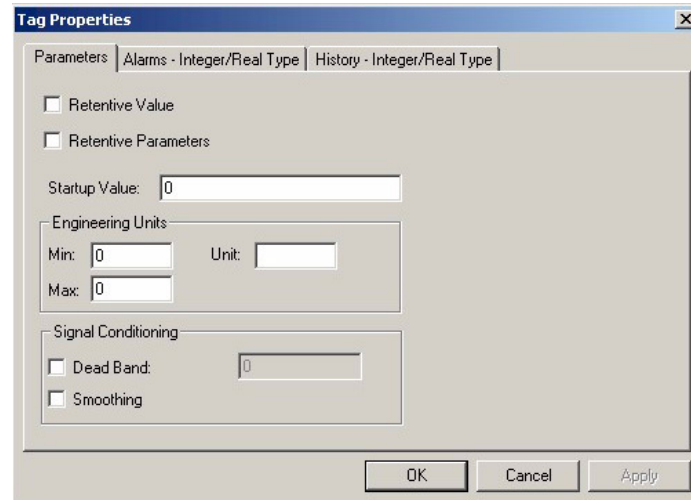
Review the table on page 5–7 to see which properties are available for the different tag types.

Instructions for configuring the parameters on each tab follow.

Configuring the Parameters Tab Properties

This section explains how to configure all of the different parameter properties.

Remember, the availability of each parameter depends on the tag type you selected.



Tag Properties: Parameters Tab

- **Retentive Value** check-box: Enable () this box to save the tag value continuously in case the system shuts down unexpectedly. When you restart the system, the tag will have retained the last-saved value.

⚠ Caution:

Specifying the **Retentive Value** parameter for tags with changing values can cause frequent hard disk access, which slows performance.

- **Retentive Parameters** check-box: Enable () this box to save all changes made to the tag fields during runtime.
- **Startup Value** field: Type a tag value for the system load. The tag uses this value if you disable the **Retentive Value** option.
- **Min** field (*accessible during runtime*): Type a minimum value for the tag in engineering units.
- **Max** field (*accessible during runtime*): Type a maximum value for the tag in engineering units.

⚠ Caution:

IWS cannot accept tag values that fall outside the specified **Min/Max** ranges. However, IWS will generate a message in the *Output* window to let you know that the system tried to write a value that was outside the defined ranges.

- **Unit** field (*accessible during runtime*): Type a string (up to nine (9) characters) to specify the tag units (such as *Ohms*, *deg.*, or *MHz*) for display purposes.
- **Dead Band** check-box: Enable () this box and type a value into the text box provided to apply a *dead band* value to the tag.

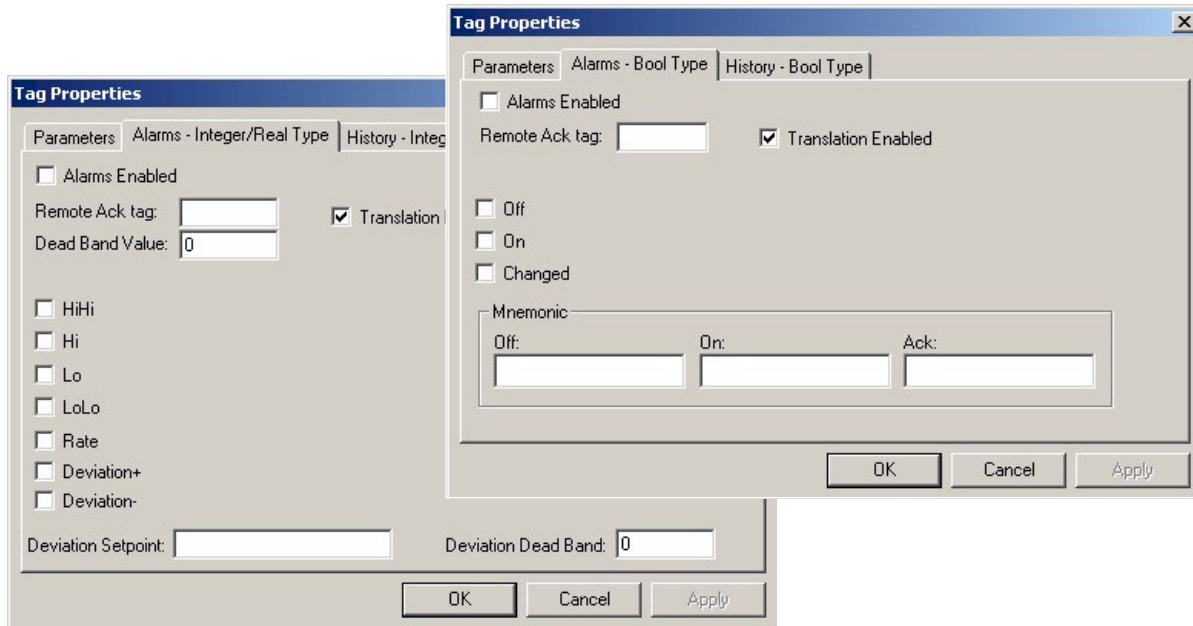
A dead band value represents a variation allowed above or below the central value of the tag (*not recognized for alarms*). When the tag value exceeds the dead band allowance, IWS generates an alarm.

- **Smoothing** check-box: Enable () this box to reduce the rate of change for the tag value.

For example, if you specify **Smoothing** for a tag containing the value 50, and the value changes to 60 in the next search, IWS stores the average of 50 + 60 in the database and the new value will equal 55.

Configuring the Alarms - Type Tab Properties

This section explains how to view and configure all of the alarm properties for a selected tag.



Tag Properties Dialog: Alarms - Type Tab

- **Alarms Enabled** check-box: Enable () this box to turn on the alarm function for this tag.
- **Remote Ack tag** field: Type a tag to allow a recipient to acknowledge an alarm remotely, which occurs when the tag values change.
- **Dead Band Value** field: Type a value to specify a dead band variation. IWS uses this value to filter alarms.

For example, if you configure the TEMP1 tag with an Hi Limit = 90 and a Dead Band Value = 5, then IWS will generate an alarm when TEMP1 is ≥ 90 . The alarm normalizes (that is, returns to normal) when $TEMP1 \leq 85$.

- **Translation Enabled** check-box: Enabling () this box allows IWS to translate alarm messages into another language(s). (For more information about translating applications, read *Chapter 16: Using the Translation Editor*.)

 **Note:**

IWS automatically saves *Alarm* messages with the **Translation Enabled** attribute enabled by default. These messages are saved to a file called **Alarm.txt**, which is located in your application's **\DATABASE** directory.

- **Alarm type** check-boxes (available during runtime): Enable () one or more of the following alarm types for this tag
 - **HiHi**: Enables *Very High* alarms.

- **Hi:** Enables *High* alarms
 - **Lo:** Enables *Low* alarms
 - **LoLo:** Enables *Very Low* alarms
 - **Rate:** Creates an alarm based on rate of change
 - **Deviation:** An alarm based on deviation from a given set point is present.
- Example:
- o If SetPoint = 50, Deviation + = 5, Deviation - = 5, and Deviation Dead Band = 0.5;
 - o IWS generates an alarm when the temp ≥ 55 or temp ≤ 45 ; and
 - o A return to the normal occurs when temp ≤ 54.5 or temp ≥ 45.5 .

When you enable any of the preceding alarm types the following fields display on the **Alarms** tab:

The screenshot shows the 'Tag Properties' dialog box with the 'Alarms - Integer/Real Type' tab selected. The 'Alarms Enabled' checkbox is checked. Below it are fields for 'Remote Ack tag' and 'Dead Band Value' (set to 0). The 'Translation Enabled' checkbox is also checked. A table lists alarm types with columns for 'Limit', 'Message', 'Group', 'Priority', and 'Selection'. All alarm types are checked. The 'Rate' type has a dropdown menu set to '1/min'. At the bottom, there are fields for 'Deviation Setpoint' and 'Deviation Dead Band' (set to 0). Buttons for 'OK', 'Cancel', and 'Apply' are at the bottom right.

	Limit	Message	Group	Priority	Selection
<input checked="" type="checkbox"/> HiHi	0		0	0	
<input checked="" type="checkbox"/> Hi	0		0	0	
<input checked="" type="checkbox"/> Lo	0		0	0	
<input checked="" type="checkbox"/> LoLo	0		0	0	
<input checked="" type="checkbox"/> Rate	0	1/min	0	0	
<input checked="" type="checkbox"/> Deviation+	0		0	0	
<input checked="" type="checkbox"/> Deviation-	0		0	0	

Alarm Tab Fields

Configure these fields (which correspond to *Alarm* worksheet columns) as follows:

- **Limit:** Type a value to specify limits for that alarm type. Note that the **Rate** parameter also provides a drop-down list that allows you to specify a rate for this limit **1/s**(econd), **1/min**(ute), and **1/hour**.
- **Message:** Type a message string to display when IWS generates an alarm.
- **Group:** Type a value indicating which group number (worksheet number) this tag belongs. IWS uses this value to filter alarms by one or more user group(s). You also can use a comma or a dash to specify more than one group (for example, 1,3,5-6).
- **Priority:** Type an integer number (from 0 to 255) to indicate priority in a group. Tags with a higher priority must have a higher priority value.
- **Selection:** Type a string which will be used to filter alarm messages. This string can have a maximum of seven characters (all others characters will be ignored).
- **Deviation SetPoint** field (accessible during runtime): Type a value indicating a reference point for the deviation.

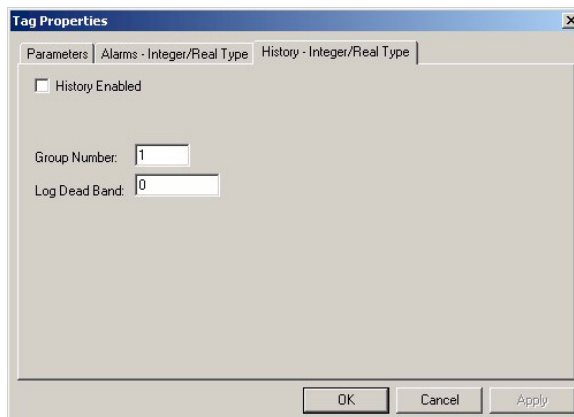
- **Deviation Dead Band** field (accessible during runtime): Type a value indicating a reference dead band value for the deviation.
- **Off** check-box: Enable () this box to generate an alarm when the tag value is zero.
- **On** check-box: Enable () this box to generate an alarm when the tag value is one.
- **Changed** check-box: Enable () this box to generate an alarm whenever the tag value changes.
- **Mnemonic** 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/Event Control Object* displays these mnemonics in the **Value** column for Boolean tags associated with the alarm message.

 **Note:**

If you do not configure a mnemonic, the *Alarm/Event Control Object* displays the tag value (0 or 1) in the *Value* column.

Configuring the History Tab Properties

This section explains how to view and configure history properties for a selected tag. These properties correspond to columns on a *Trend* worksheet (which will be discussed on page 8–13). IWS disables these tab properties if you have a *Trend* worksheet open. Before using this dialog, you must have already created *Trend* groups.



Tag Properties Dialog: History Tab

 **Notes:**

- History does not support string-type tags.
- IWS automatically saves history (.hst) files in the *Application* folder, unless you explicitly change this default location.

From the *Tag Properties* dialog, enable history logging as follows:

- **History Enabled** check-box: Enable () this box to store tag value samples.
- **Group Number** field: Type the group number associated with this tag.
- **Log Dead Band** field: Type a value to specify a **Dead Band** variation for logging purposes. When the tag value is equal to or greater than this value, IWS will take a sample for the history file.

Setting this value does not affect the **Dead Band** value you may have specified on the **Alarm** tab.


Note:

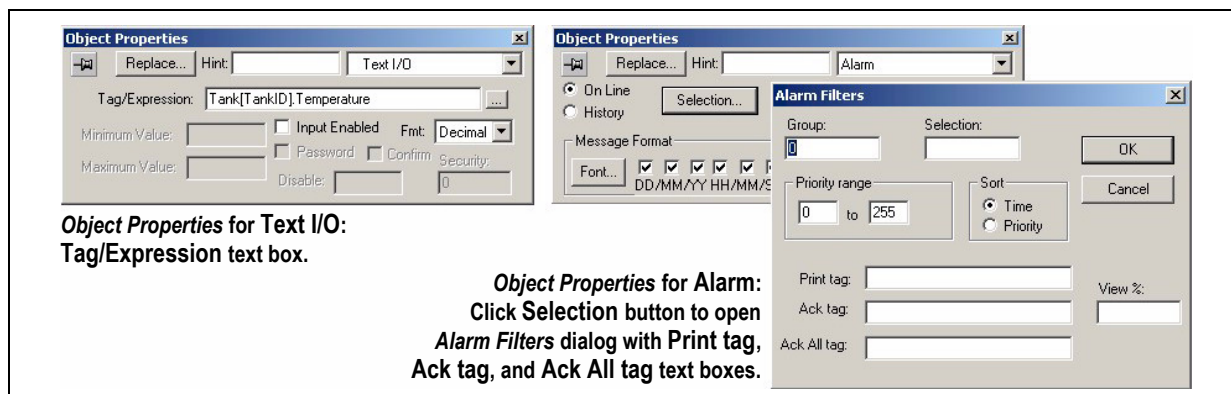
Although you can apply tag properties to system tags, if you download system tags to a CE device, the tag properties will not persist through the download. You will lose the properties.

Using Tags in Your Application

Once you have added a tag to the application database, you can use that tag in your application by associating it to objects on the display screen.

The basic process for associating tag to screen objects consists of the following steps:

- ✓ In the application screen, select the object to which you want to apply the tag.
 - ✓ Click one of the following property buttons  and an *Object Properties* dialog displays.
 - ✓ 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 will be provided throughout this manual where appropriate.

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 Application Tags Datasheet

Use the following steps to edit one or more tags in the *Application Tags* datasheet:

- ✓ Select the **Database** tab and double-click on the **Datasheet View** button  **Datasheet View**.
- ✓ When the *Application Tags* datasheet opens, locate your tag.
- ✓ Double-click in the column containing the information to be changed, and type the new information into the datasheet.
- ✓ When you are finished editing, select **File** → **Save** to save your changes to the *Tags* database.

⇒ **Tip:**

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

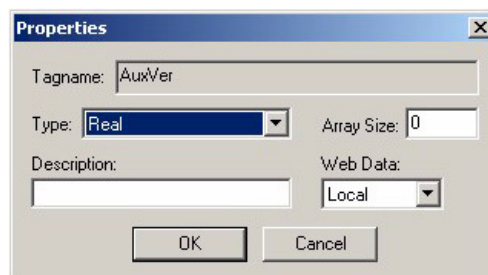
FROM THE TAG LIST FOLDER

Use the following steps to edit one or more tags from the *Tag List* folder:

- ✓ Select the **Database** tab and double-click on the the *Tag List* folder to view a list of all your tags.
- ✓ 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 select **Properties** from the pop-up menu.



Properties Dialog

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


- ✓ Make your changes in the *Properties* dialog as follows:
 - To change the current **Tag Type** or **Web Data** properties, click the arrow button and select the new information from the list.
 - To change the **Array Size** or **Description**, highlight the existing text and type the new information into the text box.
- ✓ Click **OK** to save your changes to the *Tags* database and close the *Properties* dialog.

 **Tip:**

You can sort the data in the Application Tags sheet and/or insert/remove additional columns to/from the sheet by right-clicking on it and choosing the applicable option from the pop-up menu.

Deleting Tags

 **Caution:**

Before deleting a tag, we strongly recommend using the **Object Finder**  button on the *Tag Properties* toolbar (see “Using the Object Finder” on page 3–47) to verify that you are not using the tag in another part of the application (screens, math sheets, so forth). If you delete a tag from the application database that is being used in another part of the application, you will cause a compiling error and your application will function poorly.

To delete a tag from the *Tags* database, use the following steps:

- ✓ Open the project *Application Tags* datasheet as described on page 5–25.
- ✓ Locate and click on the tag name you want to delete.
- ✓ Cut the tag from the datasheet as follows:
 - Select **Edit** → **Cut** from the main menu bar.
 - Right-click on the tag name and select **Cut** from the pop-up menu.
 - Press the **Ctrl+x** from the keyboard.
- ✓ When the pop-up displays with the “**Are You Sure?**” prompt, click **Yes** to continue or **No** to cancel the deletion.
- ✓ Save the datasheet (**File** → **Save**).

Chapter 6: Creating and Configuring a Project

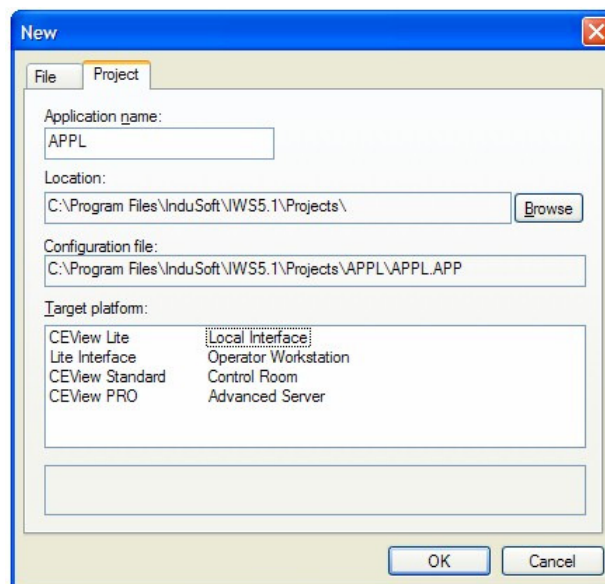
This chapter explains how to create and configure projects in InduSoft Web Studio. The information in this chapter is organized as follows:

- Creating a New Project Application
- Specifying Additional Project Settings
- Starting Runtime Modules on the Target System

Creating a New Project Application

Use the following steps to create a new project application:

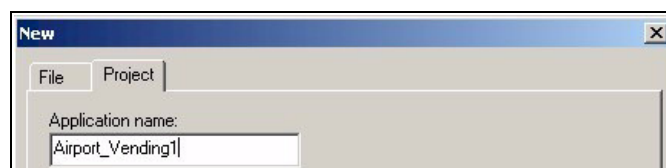
- ☑ From the main menu bar, select **File** → **New**.
- ☑ When the *New* dialog displays, select the **Project** tab.



New Dialog: Project Tab

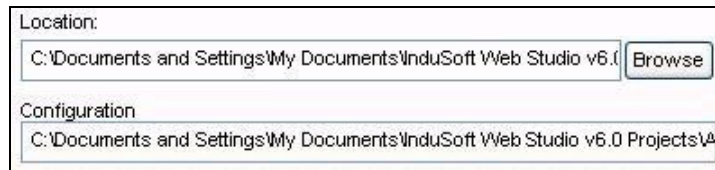
You must provide a name for the project application and indicate where to store the project and all of its related files (such as the configuration file).

- ☑ Type a name into the **Application name** text box—keeping the following guidelines in mind:
 - You must follow the usual Windows naming conventions.
 - Do not use spaces in the name if you want to access your project from the Web. (URLs do not recognize spaces.)



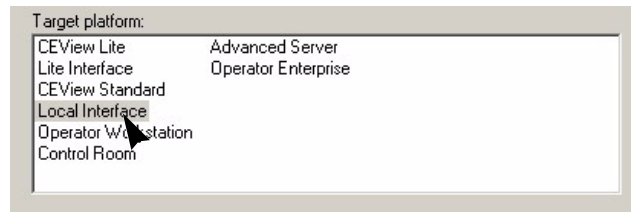
Naming the Project

- Notes:**
- IWS stores all projects in the **C:\Documents and Settings\\My Documents\InduSoft Web Studio v6.1 Projects** directory by default, so this path will display in the **Location** text field (a *view-only* field) automatically.
 - The **Configuration file** text box also is a *view-only* field. IWS automatically stores your project configuration file in the same folder as your project.



Storing Project and Configuration Files

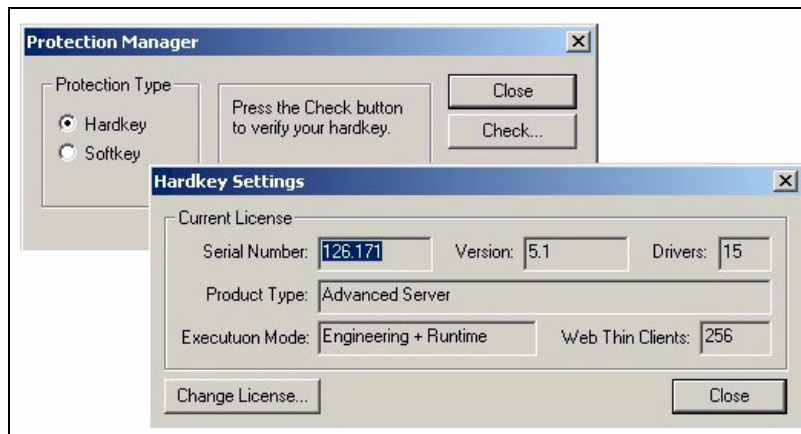
- ✓ To save your project to a location other than the default directory, click the **Browse** button to locate and select a folder.
- ✓ Select a license level from the **Target platform** list to indicate the platform on which you will be running this project.



Target Platform List

To verify your current license level, exit all open IWS modules and then select **Start → Programs → InduSoft Web Studio → Register**.

- ✓ When the *Protection Manager* dialog displays, click the **Check** button to open the *Hardkey Settings* or *Softkey Settings* dialog (depending on your license protection type), which contains your license information.
- ✓ **Close** both dialogs when you are done.



Verifying Your License Level

IMPORTANT!

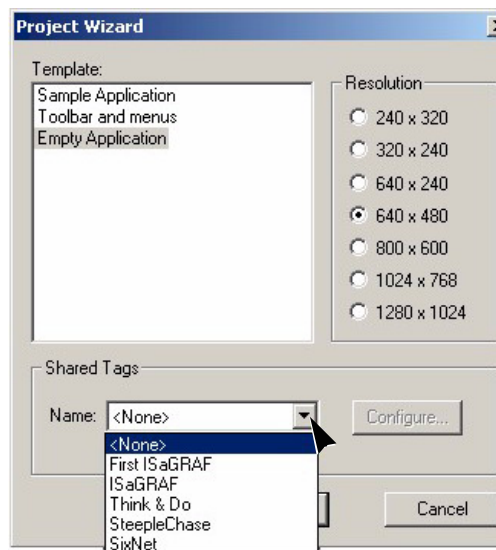
If you try to open or develop an application that was originally developed using a higher license level than exists on your current computer, IWS will prevent you from opening the application and put you into *Demo* mode.

To correct this situation, select **Project** → **Settings** from the main menu bar. When the *Project Settings* dialog displays, select the **Options** tab, and click the **Target Station** arrow button. IWS displays only those target platforms you are authorized to use. Select one of these target platforms to accommodate your current license level.

Although you can change your license level later (using the *Project Settings* dialog), we recommend that you verify and select the correct license level now, so you will not waste time developing applications requiring a license level you are not authorized to use on the runtime device.

Also, we recommend developing for the lowest license level if you will be running your application on multiple computers with different licenses.

- ✓ When you are finished configuring the **Project** tab, click **OK** to close the *New* dialog. The *Project Wizard* dialog displays automatically, as follows:



Project Wizard Dialog

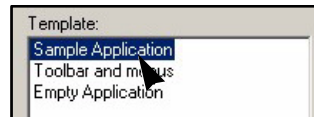
- ✓ Use this dialog to specify a template, set the default screen resolution, and to indicate that you want to share tags with another software product database, such as Steeplechase or ISaGRAF.

Instructions for using these features are provided in the following sections.

Using a Template

IWS gives you the option of using a template to create your project application or starting from scratch with no pre-configured screens or worksheets. To indicate your preference, select one of the following options from the *Templates* pane in the *Project Wizard* dialog:

- **Sample Application** (*recommended if you are using IWS for the first-time*): Select this option to use the IWS sample application as a template for your application. The sample application contains pre-configured screens and tags, a header, footer, and some background logic. Using this option provides a good base for your project and can save development time.

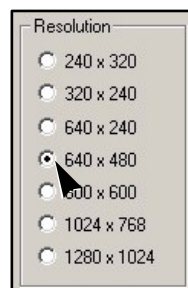


Selecting a Template

- **Toolbar and menus:** Select this option to use just the pre-configured (basic) toolbars, menus, and tags from the sample application. This option also saves development time, but does not include extraneous screens and tags that you might not use.
- **Empty Application:** Select this option to create a project application with no pre-configured features or defaults. Experienced users may prefer this option to avoid having to spend time removing extraneous screens, tags, and so forth.

Specifying a Default Screen Resolution

Use the buttons in the *Resolution* pane to specify a default resolution (display size) for your project's application screens.



Selecting a Default Resolution

Note:

You can adjust the screen resolution “on-the-fly” by selecting **Tools** → **Convert Resolution**. The **Convert Resolution** feature uses the default screen resolution as a base and converts the size of all of your project application screens accordingly.

Sharing PC-Based Control Software Program Database Tags

You can configure your project to share database tags created in a PC-based control software program, such as *SteepleChase* or *SixNet*. When you define a sharing

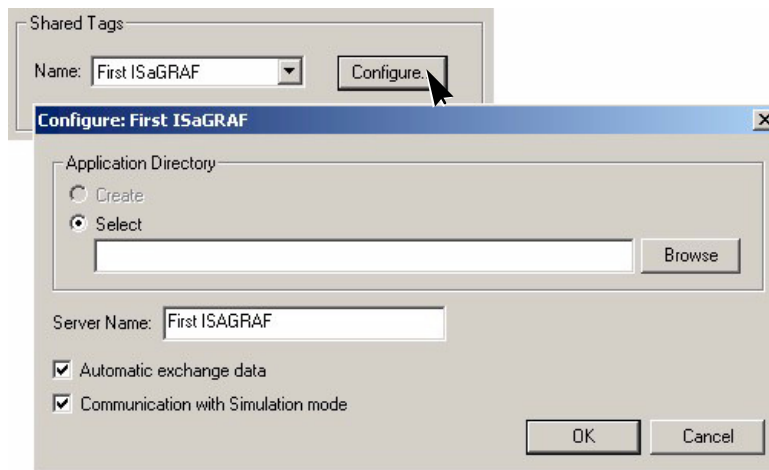
relationship and select a database, IWS will retrieve the tags and add them to your IWS application database. IWS also configures your application's Driver worksheets and tags for use in the other program.

Note:

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.

To define a shared relationship between IWS and a PC-based software control program, use the following steps:

- ✓ From the *Shared Tags* panel on the *Project Wizard* dialog, select one of the following software programs from the **Name** drop-down list.
 - **First ISaGRAF**
 - **ISaGRAF**
 - **Think & Do**
 - **SteepleChase**
 - **SixNet**
- ✓ When the **Configure** button becomes active, click the button to open the *Configure:<Program Name>* dialog.



Sharing Tags

- ✓ In the *Application Directory* pane, click (enable) one of the following buttons:
 - **Create:** Create a directory for a new application.
 - **Select:** Provide the location of an existing application directory.
Type the directory path and folder name into the **Select** field, or click the **Browse** button to select a location.

Note:
The remaining parameters on the *Configure:<Program Name>* dialog will vary depending on the program you selected in the first step.

Every PC-based control program has its own, customized interface. Before you can share information between a PC-based control application and IWS, you must provide the information about the control program interface.

- ☑ Using the manufacturer's documentation for your PC-based control program, configure the remaining parameters on the *Configure:<Program Name>* dialog:

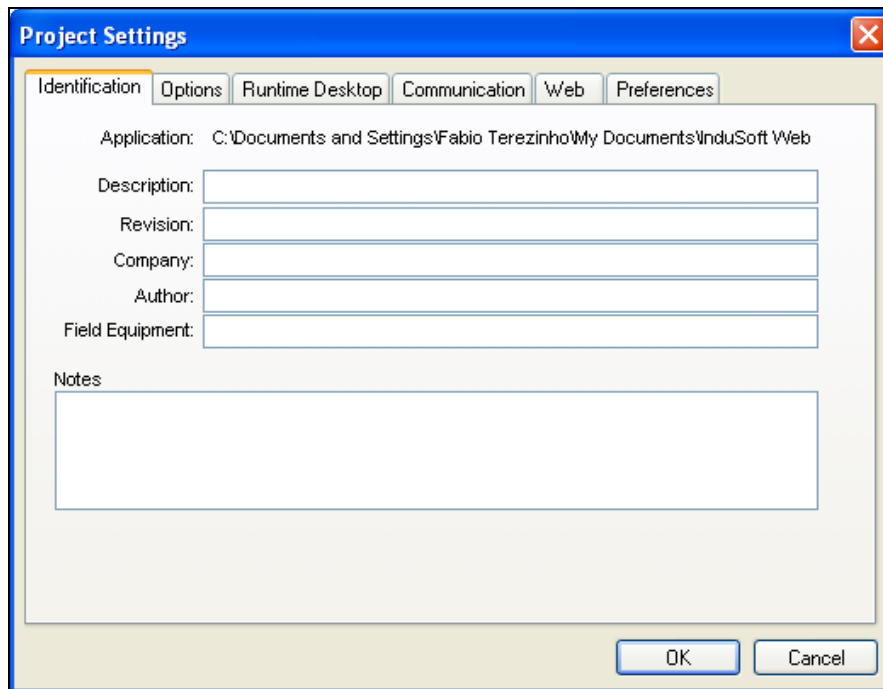
Parameter	Description
Server Name text field	IWS automatically completes this field. To change the default, type the appropriate information into the text box.
Automatic exchange data check-box	Enabling (<input checked="" type="checkbox"/>) this box allows the two programs to exchange data automatically.
Communication with Simulation Mode check-box	Enabling (<input checked="" type="checkbox"/>) this box allows the two programs to communicate in Simulation Mode.
IP Address text field (Think and Do only)	Type the IP address of the shared PC-based control application.
Project Options Pane:	
Name text field (Steeplechase and SixNet only)	Type the name of the shared PC-based control application.
Status text field (Steeplechase only)	Type the current status of the shared PC-based control application. (For example,)
Version text field (Steeplechase only)	Type the current version of the shared PC-based control application.
Last Modification text field (SixNet only)	Type the date when the shared PC-based control application was last updated.
Update button (Steeplechase and SixNet only)	Click this button to update your IWS application with data from the shared PC-based control application.

Configure: <Program Name> Parameters Table

Specifying Additional Project Settings

After creating a new project, you can use the following steps to configure some additional parameters for that project:

- ☑ From the menu bar, select **Project** → **Settings** to open the *Project Settings* dialog, which controls settings that affect the overall application.



Project Settings Dialog – Identification Tab

The *Project Settings* dialog contains five tabs:

- **Identification**
- **Options**
- **Runtime Desktop**
- **Communication**
- **Web**
- **Preferences**

Descriptions of each tab, and instructions for configuring parameters on these tabs, are in the next section.

- ☑ When you finish configuring the parameters on the *Project Settings* dialog, click **OK** to close the dialog.

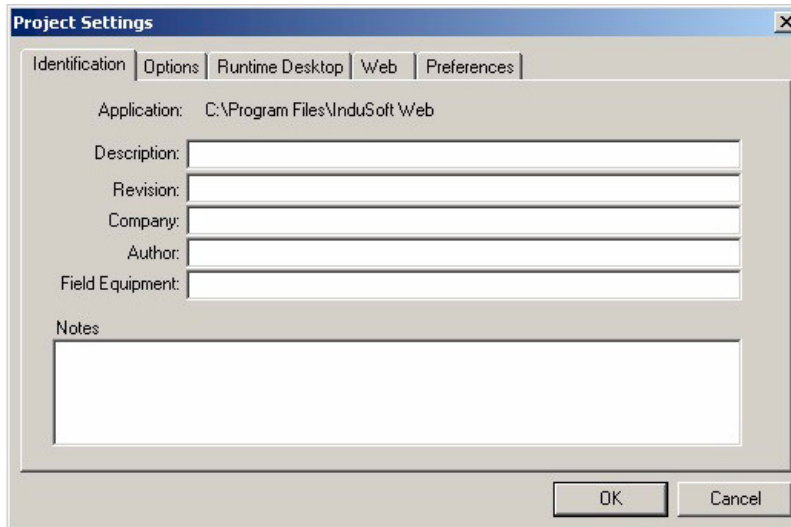
Note:

Although you can change the *Project Settings* parameters at any time during application development, we recommend configuring these parameters at the beginning of your project.

For example, the **Startup screen** field defines which screen will open when you start the application, so if you try to emulate or run the application without a legitimate value in this field, IWS will generate an error message.

Providing Project Identification Information

When you open the *Project Settings* dialog, the **Identification** tab displays by default:

The image shows a screenshot of the 'Project Settings' dialog box. The title bar reads 'Project Settings' with a close button (X) on the right. Below the title bar are five tabs: 'Identification', 'Options', 'Runtime Desktop', 'Web', and 'Preferences'. The 'Identification' tab is selected. The main area contains several text input fields: 'Application:' with the value 'C:\Program Files\InduSoft Web', 'Description:', 'Revision:', 'Company:', 'Author:', and 'Field Equipment:'. Below these fields is a larger text area labeled 'Notes'. At the bottom right of the dialog are 'OK' and 'Cancel' buttons.

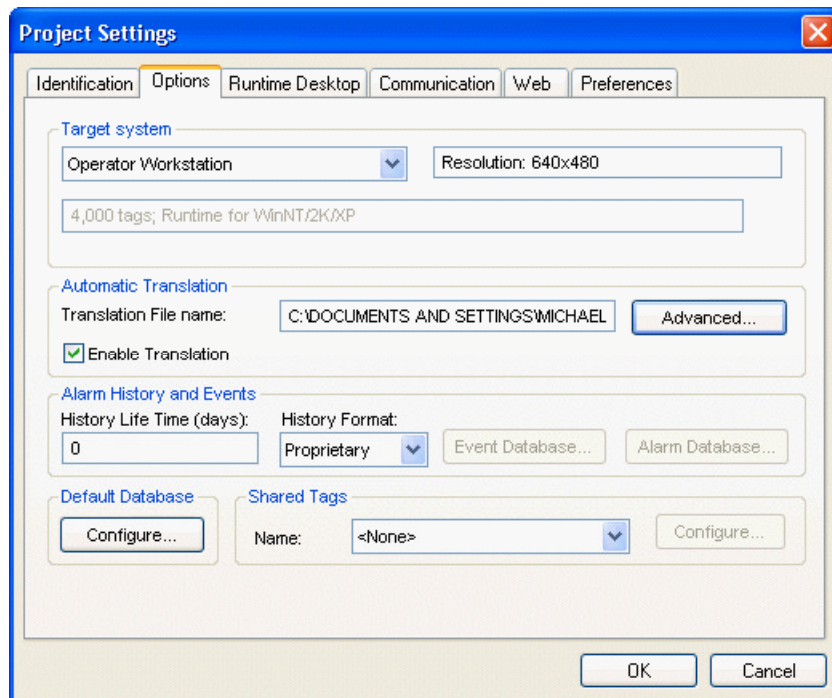
Project Settings Dialog: Identification Tab

- ✓ Use the text fields on this tab to provide the information about your project. Configuring these fields is *optional*, but the information can be very useful if you are sharing your project application with coworkers, customers, and so forth.
 - **Description:** Provide a brief description of your project (such as its purpose).
 - **Revision:** Type the current revision number/letter to keep track of revisions made to your application.
 - **Company:** Type a company name to associate your project with a particular company (for example, your company name or a client's company name).
 - **Author:** Type the name of the project author.
 - **Field Equipment:** Provide the name or type of equipment on which your project application is running.
 - **Notes:** Enter any notes you think will be useful.
- ✓ When you are finished, click another tab to continue or close the dialog.

Setting the Options Tab Parameters

Select the **Options** tab to configure the following:

- Enable language translation
- Specify a target system
- Specify how long to save an *Alarm History* file
- Indicate whether to share tags with another software application
- Specify how to buffer Driver and OPC write commands
- Specify a TCP port
- Configure settings for the default database



Project Settings Dialog – Options Tab

A description of the **Options** Tab parameters follows:

- **Target system:** Use the combo box to specify the target system for the current project. The target system sets the application 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 station, the application may not run properly.

- **Resolution:** Displays the application resolution.

- **Automatic Translation:** Click (enable) the Enable Translation check box to enable the translation feature. Type the directory path and a translation file name into the Translation File name field. IWS uses this translation file as the default language when you start the application.
- **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, IWS automatically deletes existing alarm/event history files that are older than the period specified. If you type zero in this field, IWS 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/Event history, as follows:

Type	Description
Proprietary	Saves the history data in the \Alarm sub-folder of the application (by default) in text files using the proprietary format of Studio.
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 \Alarm sub-folder of the application (by default) in binary files using the proprietary format of Studio.

- **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:** 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:

Name	Description
<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 application, 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 application, 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 application, 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 application, 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 application, and configure the communication interface with the SNET driver automatically.
Open Control	Import tags from an OpenControl project into the Shared Tags folder of the current application, 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 application, and configure the communication interface with the STRAT driver

automatically.

Note:

Each 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 application.

ENABLING LANGUAGE TRANSLATION

One of the utilities provided with IWS is a *Translation Editor*, which enables you to translate applications from one language into another automatically—and translate these applications into as many languages as necessary.

If you want to translate your project into another language, you must enable translation and specify a default translation file from the *Automatic Translation* panel on the *Options* tab:

- Enable () the **Enable Translation** check-box.
- Use the **Translation Filename** text box to specify a *default translation file name* and location.

The project's *default translation file* (**<project name>.csv**) controls the language in which your development environment and project applications will display.

For example, if you specify a Spanish default translation file, your development environment and project applications will display in Spanish by default.

- If the default file name and location are acceptable, no action is required.
- If you want to specify a different translation file or location, type the file name and location into the text box.

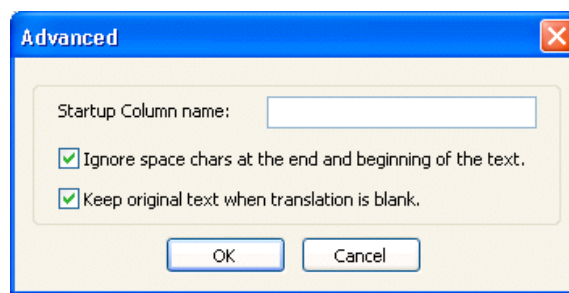
Note:

You can save translation files to any directory, but we strongly recommend saving these files in your project's *Web* folder so they can be used by a Web Thin Client.

For example, if your default development language is *English* and you want to make the file available to Web Thin Clients, you might change the default worksheet name to **English.csv** and save the file in the following folder:

```
C:\Program Files\InduSoft Web Studio v6.1\Projects\  
<Project Name>\Web\English.csv
```

- Click the **Advanced** button for additional translation settings:



Automatic Translation: Advanced Settings

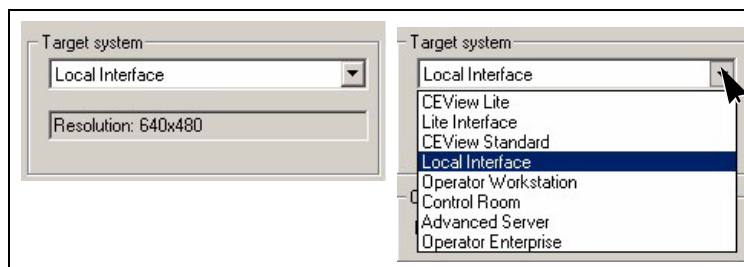
- **Startup Column Name:** You can create a CSV file with translation for more than one language. The text in the original language used when the application was created are in the first column of the CSV file. Each different language is added in additional columns. The first row of the CSV file indicates the name of each column. You can specify in this field the name of the column that must be used to translate the application by default, when launching the application. If this field is left in blank, the application is launched with its original language. The user can set a different language during the runtime by executing the `SetTranslationFile()` function.
- **Ignore space chars at the end and beginning of the text:** When this option is checked, the space character at the end or at the beginning of each text is ignored for translation purposes. This option is useful to avoid duplicated entries in the translation table due to space chars configured by mistake when creating the objects or just for alignment.
- **Keep original text when translation is blank:** When this option is checked, the original text is kept when there is not any text configured as a translation for it in the column currently set for translation of the application. If this option is unchecked, the original text is omitted from the application during the runtime, when there is not any translated text for it in the currently set for translation of the application.

After enabling translation for your project application, continue to *Chapter 16: Using the Translation Editor* for a description of the *Translation Editor* utility and to finish configuring your project for translation.

SPECIFYING THE TARGET SYSTEM'S LICENSE LEVEL

The target system on which you will run your application sets the application restrictions (such as how many tags will be supported). Consequently, the target system's license level and your development station's license level must match *or your application may not run properly*.

Click the **Target system** combo-box arrow button, and select your target system's license level from the list.



Target System Panel

The default screen resolution displays in the field just below the combo-box. This resolution is the default value you specified when you created the project.

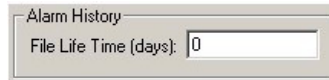
Note:

To change this resolution “on-the-fly,” close all open screens and select **Tools** → **Convert Resolution** from the menu bar. When the *Convert Resolution* dialog displays, enable the size you want to use, and then click the **Convert** button.

SAVING ALARM HISTORY FILES

The **Alarm History** panel allows you to control how long to keep alarm history files stored on your system.

Type a value into the **File Life Time (days)** field to specify how many days to keep the alarm files. For example, if you specify 7 days, IWS automatically deletes all alarm files that are more than a week old.



Alarm History Panel

SHARING TAGS WITH A PC-BASED CONTROL APPLICATION

The process for configuring this panel is identical to the process described for creating a new project. Review “Sharing PC-Based Control Software Program Database Tags” on page 6–4.

COMMUNICATING WITH DRIVERS AND OPC (BUFFER)

Because IWS contains multiple modules that can affect values in the *Tags* database, and given that only one module can run at a time, it is possible that a tag’s value can change several times before the driver or OPC Client receives the changed value to send to the PLC or OPC Server. Consequently, IWS gives you the option of buffering the tag values.

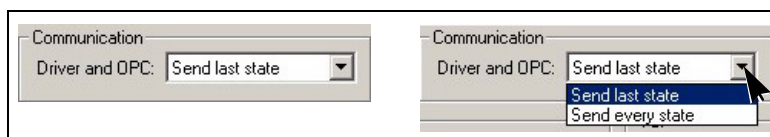
If you choose to use the buffer, IWS sends every change to a tag value to the PLC or OPC Server—in the order in which the change happened.

If you decide not to use the buffer, then IWS sends only the current value to the PLC or OPC Server.

Use the *Communication* panel parameters to specify how changes are written by drivers and OPC devices to target devices, such as PLCs and OPC Servers.

Click the **Driver and OPC** combo-box button, and select one of the following options from the list.

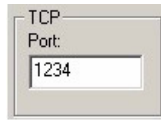
- **Send last state:** Writes just the current tag value to the device or OPC Server. (Selecting this option conserves memory.)
- **Send every state:** Writes all tag value changes to the device or OPC Server. (Selecting this option uses more memory, but ensures a complete and accurate view of the actions that occur.)



Communication Panel

SPECIFYING A TCP PORT

Use the **Port** text field on the *TCP* panel to specify a port for the IWS *TCP/IP Client* and *Server* modules to use. Type the port number into the text field:



TCP Panel

CONFIGURING DATABASE SETTINGS TO SAVE ALARM AND/OR EVENT HISTORY

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 Studio or to external SQL Relational Databases. Use the Options tab to configure the database to save Alarm and Event history. (See *Configuring Database Settings for Saving Trend History* for instructions for saving history for the trend tasks.)

Use the **History Format** combo-box to select Database, then click either the **Alarm Database** or the **Event Database** button. The Database Configuration dialog window will open.

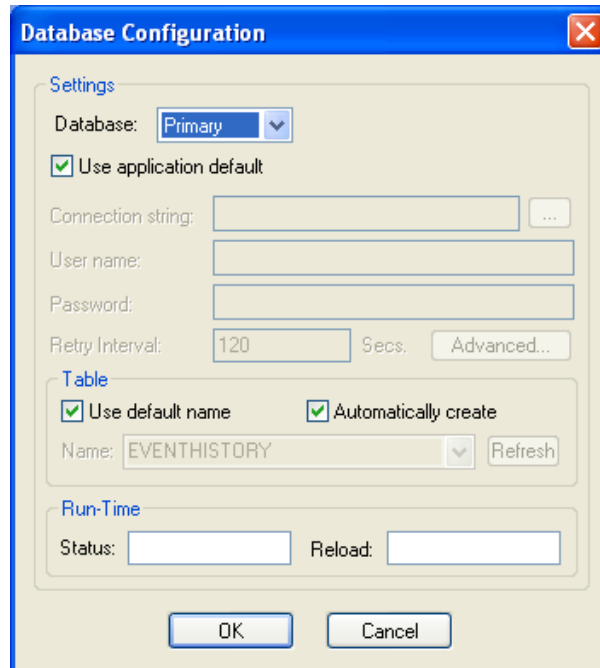


Note:

- Both **Alarms** and **Events** must be saved in the IWS proprietary format, or both **Alarms** and **Events** must be saved in external Relational Databases; however, they can be saved on different databases.
- Each **Trend** worksheet can be configured to save data either in the IWS proprietary format or in an external SQL Relational Database.

Database Configuration Dialog Window

The Database Configuration dialog window allows you to configure the necessary settings to link IWS to an external SQL Relational Database.



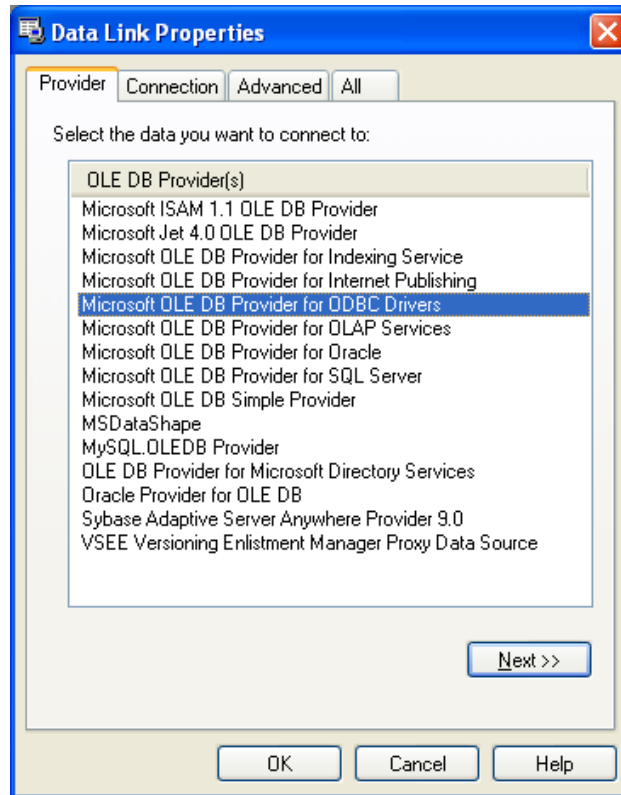
Database Configuration Window

- **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.

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 application default** check-box: 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.



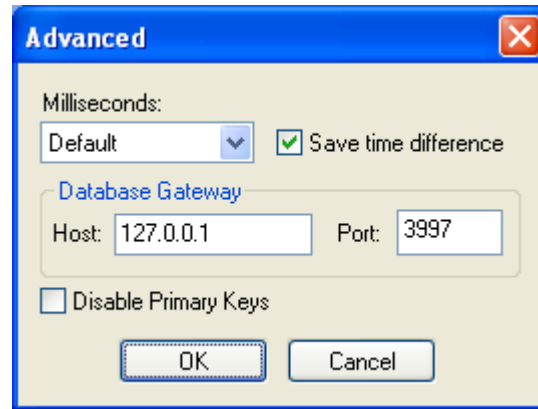
Data Link Properties window



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 applications, the default value of these settings do not need to be modified and should be kept.



Database Configuration Advanced window

- **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 by InduSoft, the **Default** option attempts to save the milliseconds in a separate field.
 - **Disable:** Does not save the milliseconds at all when saving the date in the database.
 - **Enable:** Saves the milliseconds in the same field where the date is saved.
 - **Separate Column:** Saves the milliseconds in a separated column. In this case, the date is saved in one field (without the milliseconds precision) and the number of milliseconds is saved in a different column. This option is indicated where you want to save timestamps with the precision of milliseconds but the database that you are using does not support milliseconds for the **Date** fields.

⇒ **Tip:**

The default option for each database is configured in the StudioADO.ini file, stored in the \BIN sub-folder of IWS. See “Studio Database Gateway,” in *Chapter 17: IWS Database Interface*, for information about how to configure the StudioADO.ini file.

- **Save time difference** check-box: When this option is checked (default), IWS saves the Time Zone configured in the computer where the application 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 Studio 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 Studio database gateway with the same TCP Port. See “Studio Database Gateway,” in *Chapter 17: IWS Database Interface*, for information about how to configure the advanced settings for the Studio ADO Gateway.

- **Disable Primary Key:** 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), you should check this field.
- **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. InduSoft 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** check-box: 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** check-box: 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.
- **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: Use this area to set run-time values. The following fields are available:
- **Status** (output) check-box: The tag in this field will receive one of the following values:

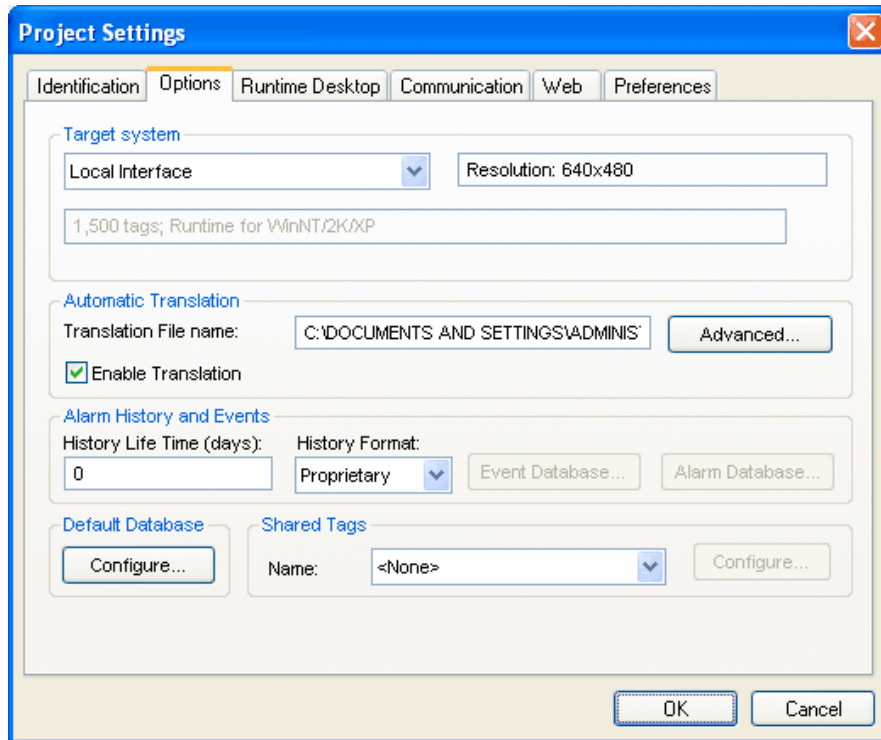
Value	Description
0	Disconnected from the database. The database is not available or your configuration is incorrect.
1	The database is connected successfully.
2	The database is being synchronized.

- **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.

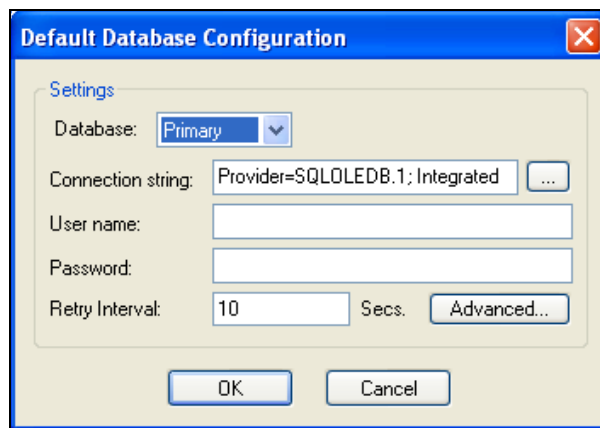
Configuring a Default Database for All Task History

Although IWS allows you to configure a different database for each task, typically the same database type (e.g. SQL Server, MS Access, Oracle, and so forth) is used by all tasks of the same project. Therefore, in order to save time when configuring the application, IWS allows you to configure the *Default Database*. When configuring each task, you can choose to use the settings configured for the Default Database. If you choose this method, it will not be necessary to re-configure the same settings for each task, since they share the same database.

The settings for the Default Database can be configured by pressing the **Configure** button from the **Default Database** box on the **Options** tab of the **Project Settings** dialog.



The Default Database Configuration dialog window opens.



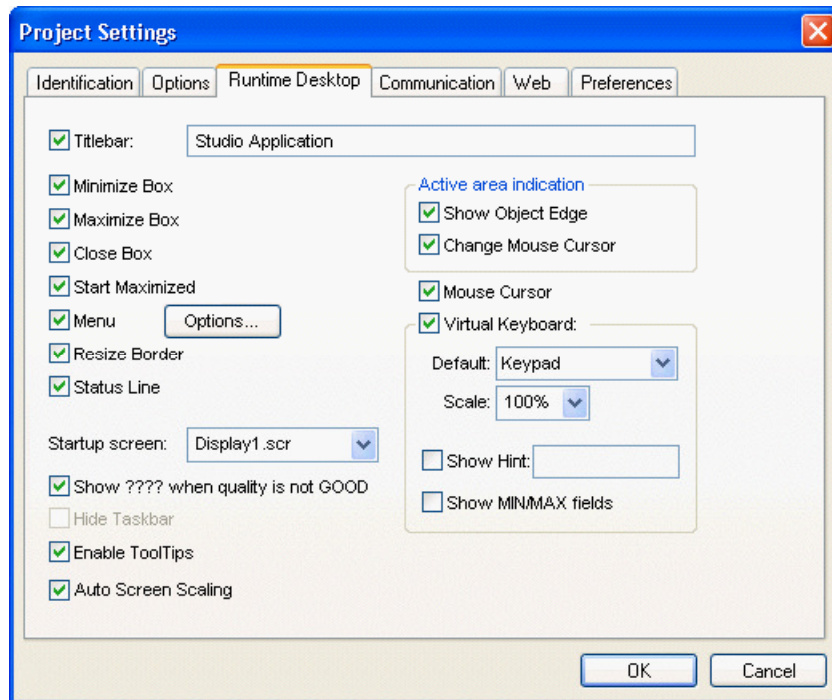
Default Database Configuration Window

Please refer to the previous section, *Database Configuration Dialog Window*, for help completing the fields in this window.

Setting the Runtime Desktop Parameters

When you run an application on the runtime (or target) workstation, IWS displays the application in a *Viewer* window. You can configure the appearance of this *Viewer* window as follows:

- Select the **Runtime Desktop** tab on the *Project Settings* dialog:



Project Settings Dialog – Runtime Desktop Tab

- Enable () the one or more of the following check-boxes to display those features on the *Viewer* window (or *disable* the boxes to hide the feature):

 **Note:**
IWS saves all of these display properties in the application's **.APP** file.

- **Titlebar:** Displays a title bar along the top of the window. Type a name into the text field to change the default title bar text.
- **Minimize Box, Maximize Box, and Close Box:** Displays three buttons so you can minimize, maximize, or close the *Viewer* window.
- **Start Maximized:** Maximizes the window automatically when you run the application.

- **Menu:** Displays a menu bar on the *Viewer* window.
To specify which menus and menu options will be available from the menu bar, click the **Options** button. When the *Runtime menu options* dialog displays, click the following check-boxes to display or hide the menu/menu options. When you are finished, click **OK** to close the dialog.



Runtime Menu Options Dialog

- **Resize Border:** Allows you to resize the *Viewer* window during runtime.
- **Status Line:** Displays a status bar along the bottom of the *Viewer* window.
- **Startup screen:** Displays a specified application screen or screen group when you start the application at runtime. Click on the combo box and select a screen (.SCR) or screen group (.SG) from the list.

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

- **Show ???? when quality is not GOOD:** Displays question marks (???) instead of a tag's value when the tag quality is not good.
- **Hide Taskbar:** Hides the *Windows* taskbar by default.
- **Enable ToolTips:** Displays *Windows ToolTips* when you run the application.
To configure tool tips for an object in your application, double-click the object to open the *Object Properties* dialog and type your tip into the **Hint** field.
- **Auto Screen Scaling:** Scales the application screen automatically when you resize the *Viewer* window. This feature is available for local applications running on Windows 2K/XP/Vista (or Web Thin Clients). *This parameter is not available for local applications running on Windows CE.*
- **Show Object Edge:** Changes the object edge when you move the cursor over any object where the **Command** dynamic was applied.
- **Change Mouse Cursor:** Changes the mouse cursor when you move the cursor over any object where the **Command** dynamic was applied.
- **Mouse Cursor:** Displays a mouse cursor in the *Viewer* window.

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

- **Virtual Keyboard:** When this option is checked, the Virtual Keyboard (see page 3–30) is enabled for the application. (This option does *not* apply when the application is running on a Web Thin Client; for more information, see “Specify Web Thin Client Parameters” on page 6–25.) 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 I/O dynamic property 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 the application, 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.

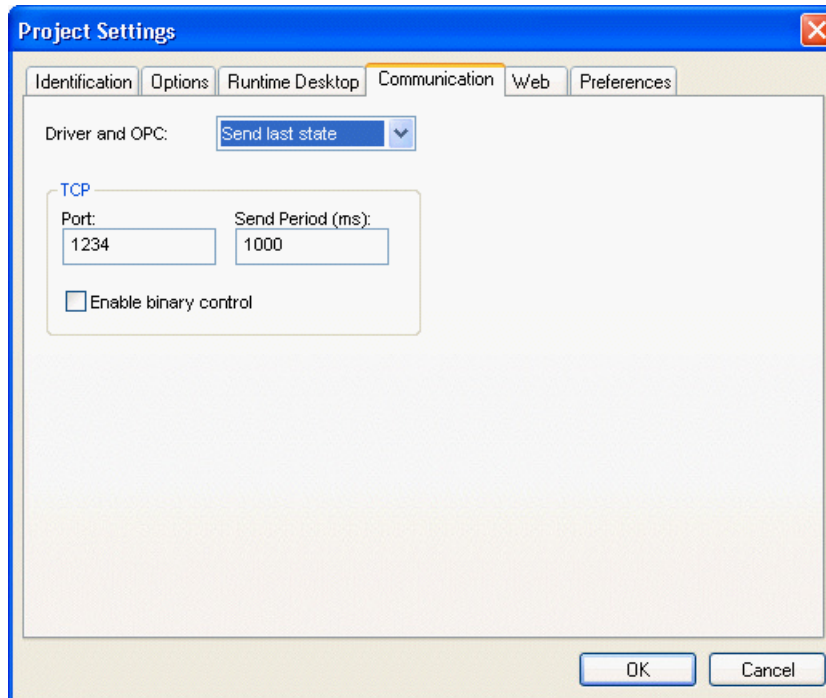
 **Note:**

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 the application will attempt to scale the input accordingly.

- When you are finished, click another tab to continue or close the dialog.

Setting the Communication Parameters

Use this tab to specify communication parameters relating to the application in general.



Project Settings > Communication Tab

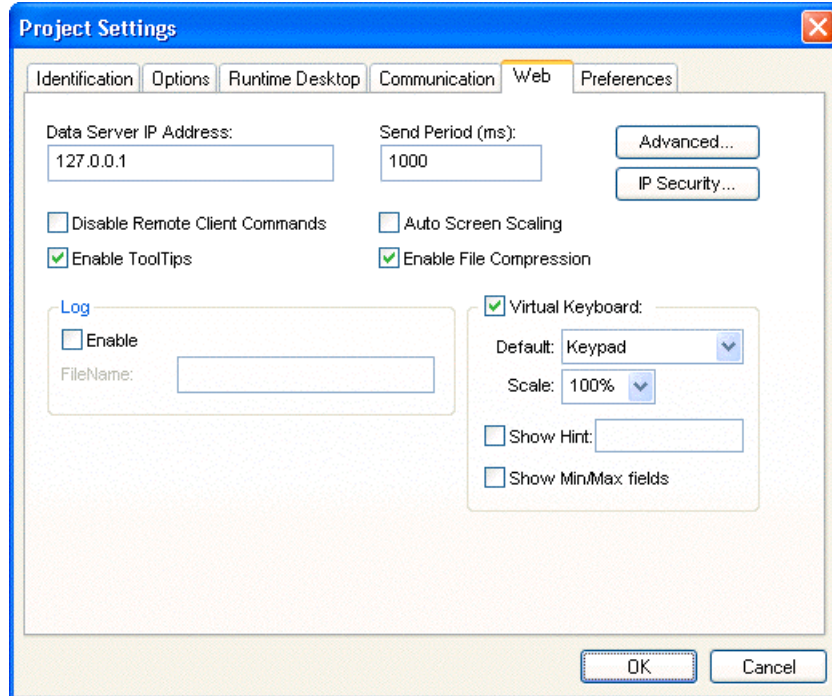
- **Driver and OPC:** Select the method used by all communication drivers and OPC Client worksheets configured in the current application when writing values to the remote PLC/device:
 - * **Send every state:** When the communication task is configured to write values upon a change of tag value, all 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 applications.
- **TCP:** Configure the settings for the TCP/IP Client and TCP/IP Server modules:
 - * **Port:** 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 application, be sure to change the same value in the remote application that is communicating with the local one.
 - * **Send Period (ms):** 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:** 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 applications or disabled in both.

Specify Web Thin Client Parameters

Use the following procedure to specify the Web Thin Client parameters:

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



Project Settings Dialog: Web Tab

- ☑ 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 Web 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 IP 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 (ms)** field: Type a value to specify the send period (in milliseconds) used to exchange data between the server and the Web Thin Client stations. It means that the Web 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 **Project Settings > Communication**. 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 Web 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** check box: Click to enable this box, to prevent a remote client from issuing commands from your Web Thin Client to your server. When this option is enabled, the Web Thin Client can read data from the server, but cannot send data (tag values, set-points) to the data server. In this case, the Web Thin Client station becomes a *Read Only* station.
- **Enable ToolTips** check box: Click to enable this box, to display the *ToolTips* configured on the objects of the screens when viewing them on the Web Thin Client (Web browser).
- **Auto Screen Scaling** check box: 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** check box: Click to enable this box to compress the files stored on the **\Web** folder of the application. This option is useful for reducing download time, particularly if you have a slow connection between your server and the Web Thin Client.
- **Log (Enable check box and FileName text field)**: Click to enable the check box, and type a file name into the text field to generate a log file on the Web Thin Client station. You can use this log file for debugging purposes.
- **Virtual Keyboard**: When this option is checked, the Virtual Keyboard (see page 3–30) is enabled for the application running on a Web Thin Client. 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 I/O dynamic property applied and the Input Enabled option checked.

You can establish a default configuration for the virtual keyboard:

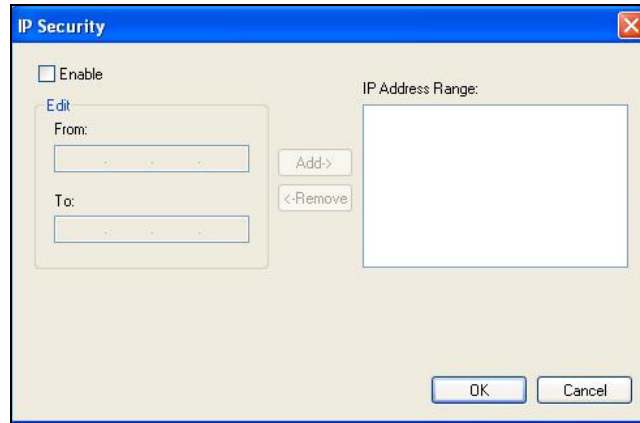
- **Default**: Select the default keyboard type to be used in the application, when no keyboard type is specified by the calling object or function.
- **Scale**: Using this option, you can shrink or enlarge the keyboard to fit the size of the target display. A value of 100% represents the default size 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 the application will attempt to scale the input accordingly.

- **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 the application as Web Thin Clients. This option is useful when you can control the IP Address of the Web Thin Client computers allowed to connect to the Server.

Click the **Enable** check box, 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 Web Thin Client attempts to connect to the server, it checks for an IP address for the Web Thin Client station that is within any range configured in the IP Security dialog box. If one is not found, the server refuses the connection request from the Web Thin Client station.


Note:
By default, IP security applies only to Web Thin Clients connecting to the Data Server. You can also implement IP security for database synchronization between applications running on different stations (see “Configuring TCP/IP” on page 10–25). To do this, insert the following parameter into the application file (*.app):

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


- **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.
 - **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 Web Thin


Clients. If the primary data server fails, the Web 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 the application). This URL is used to download the files from the secondary Web server when the primary Web server is not available.

 **Note:**
When using Web browsers running under Windows CE v3.0 operating system or Windows CE Pocket PC, the backup URL field must be configured with the URL of the primary Web server, even if you do not have a redundant architecture.

- **ISSymbol download path** field: When the Web Thin Client connects to the server, it attempts to load the ISSymbol control. If ISSymbol is not registered in the local computer (Web 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 Web 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 Web 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 Web Thin Client.

- **Web Tunneling Gateway** check box: 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 Web Thin Client computers to the data server.
- **HTTP 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 Web 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 Web Thin Client station(s). This field must be configured when you are using redundant Web servers.

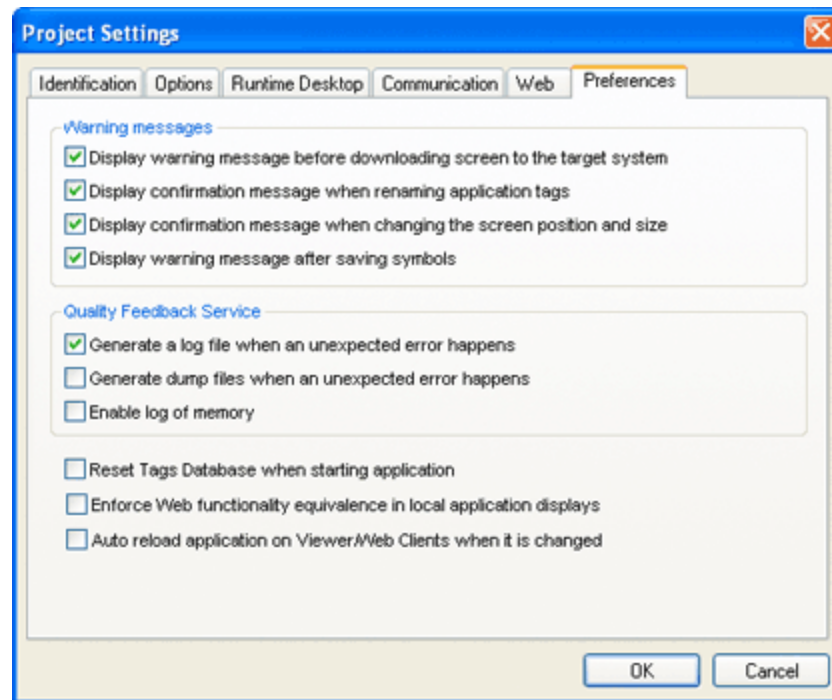
⇒ **Tip:**

Most of the Web settings can be modified dynamically by the **SetWebConfig()** function. It is especially useful when you want to create an application just once, and make sure that the Web settings are automatically configured when you run the application in different stations with different IP Addresses.

Enabling Warning Messages

IWS allows you to control whether warning messages will display before the application screens are downloaded to the target system.

- ☑ Select the **Preferences** tab.



Project Settings Dialog – Preferences Tab

- ☑ *Enable* (☑) or *disable* (☐) the options that you want to include in your application project...

WARNING MESSAGES

- **Display warning message before downloading screen to the target system** option: When this option is checked and IWS is connected to a remote station (as configured in the *Execution Environment* 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, then the screen is downloaded automatically, regardless of any confirmation.

- **Display confirmation message when renaming application tags** option: When this option is checked and you modify the name of any tag in the Application Tags worksheet, you are prompted to replace the old tag name with the new tag name in the whole application. 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 the application (screens and worksheets).
- **Display confirmation message when changing the screen position and size** option: When this option is checked, 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 checked, a warning message is shown after saving symbols.

QUALITY FEEDBACK SERVICE

This section allows you to configure IWS or CEView 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` directory of the running application.

- **Generate a log file when an unexpected error happens** option: When this option is checked, 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 checked, 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 checked, 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.

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 recommend that you enable all options in this section and then send the Log File and all Dump Files to your software vendor.

OTHER OPTIONS


- Reset Tags Database when starting application** option: When this option is checked, the application tags are reset automatically whenever you start the application (Project > Run Application). See “Resetting the Tags Database” on page 5–15 for additional details about this feature.
- Enforce Web functionality equivalence in local application displays** option: When this option is checked, the development software will automatically warn you when you try to select fuctions or features that are incompatible with the remote runtime modules (e.g. Indusoft Secure Viewer and Web Thin Client).

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

- Auto reload application on Viewer/Web Clients when it is changed** option: When this option is checked, remote stations (i.e. Web Thin Clients and Secure Viewers) will check the server to see if they have the most recent version of the application. If they do not, then they will automatically load the new version.
- When you are finished, click another tab to continue, or close the dialog.

Starting Runtime Modules on the Target System

After specifying a target system for your project application, Indusoft Web Studio allows you to specify which of your project’s runtime modules start automatically on the target system and which modules must be started manually.

 **Note:**
IWS configures certain modules to start automatically by default, but enables you to change these defaults.

With the Project Status dialog, you can:

- Configure the tasks that must be automatically executed when the application is launched (Execution Tasks tab)
- Start/stop each run-time task manually (Execution Tasks tab)
- Review information about the development system and your applications (Information tab)

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

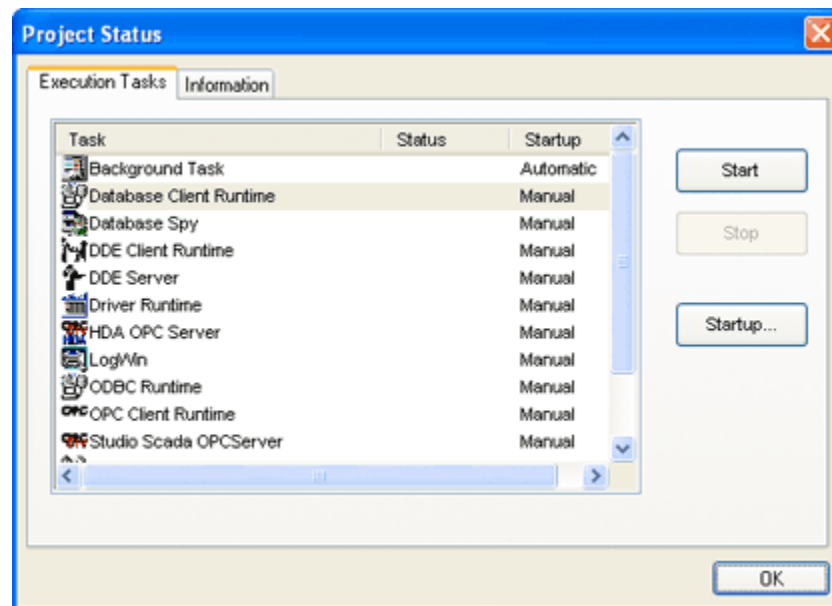
To specify how modules will start on the target system, use the following steps:

- ✓ From the main menu bar, select **Project** → **Status** to open the *Project Status* dialog.
- ✓ Select the **Execution Tasks** tab, which lists all of the runtime modules associated with your project application and their current start-up status.

Note:

The **Execution Tasks** tab is available only when the *Target Station* option from the *Execution Environment* dialog (**Project** → **Execution Environment**) is set to **Local**.

This list consists of three columns, including the *Task Name*, runtime *Status*, and current *Startup* setting (**Automatic** or **Manual**) of each runtime task. For example, see the following figure:



Project Status Dialog: Execution Tasks Tab

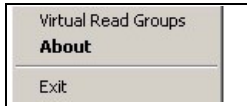
The list of tasks available in this dialog depends on the Target system configured on the Options tab. The following table lists the tasks that are not available for applications designed for the Windows CE operating system:

Task	Available for WinCE runtime OS
Background Task	Yes
Database Client Runtime	Yes
Database Spy	No
DDE Client Runtime	No
DDE Server	No
HAD OPC Server	No
Driver Runtime	Yes
LogWin	No

Task	Available for WinCE runtime OS
ODBC Runtime	No
OPC Client Runtime	Yes
Studio Scada OPC Server	Yes
TCP/IP Client Runtime	Yes
TCP/IP Server	Yes
Viewer	Yes

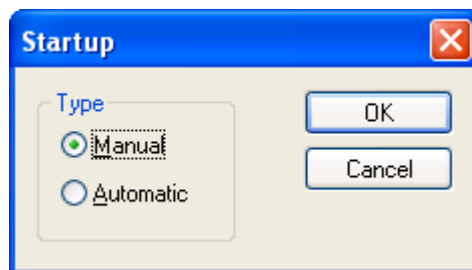
Notes:

- Because there are so many drivers and new drivers are being added all the time, the *Driver runtime* module starts all of the drivers configured in the current project application.
- Started* does not display in the *Status* column when you start the *Driver* module; instead an icon for each driver displays in the lower right-hand corner of the *Windows* taskbar.
- To stop an individual driver, right-click the icon and select **Exit** from the pop-up menu.




Verify that Drivers are Running

- ✓ You can configure tasks for automatic execution when the application is launched by clicking the name on the Execution Tasks tab, pressing the Startup button and then choosing Automatic.



Startup Dialog

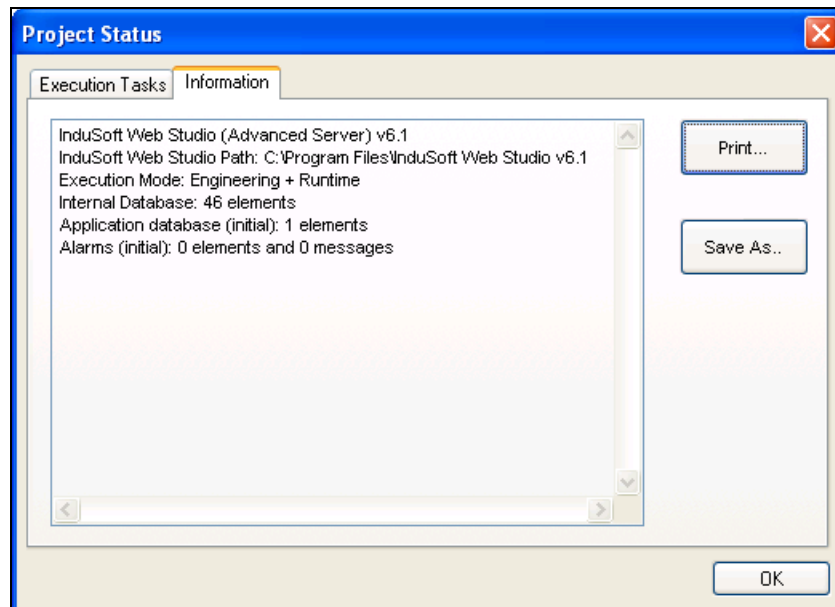
- The tasks configured with Startup = Automatic are automatically executed when the application is launched; the tasks configured with Startup = Manual are not.
- You can also Start/Stop each run-time task by clicking the name and then pressing the Start or Stop button.

Note:
If the Execution Tasks tab is not visible, click Project -> Execution Environment and make sure the Target Station is set to Local. The Execution Tasks tab is available only when the Target Station is set to local.

Tip:
You can also start/stop each task during the 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 the runtime.

Caution:
The commands triggered by this dialog will be executed in the remote station (not in the local computer) when you are connected to a remote station.

- ✓ The **Information** tab from the *Project Status* dialog displays the list of tasks available for the current application, as illustrated in the following picture:



Project Status Dialog: Information Tab

The most important information displayed in this dialog is the Application database (initial), which provides the number of application tags configured in the current application. This information is useful to evaluate which license is more suitable for the application.

- ✓ When you are finished, click **OK** to close the *Project Status* dialog.

After you create and configure a project, you can design your application screens and create your task worksheets.

- Instructions for creating screens are provided in *Chapter 7: Configuring Screens and Graphics*.
- Instructions for creating worksheets are provided in *Chapter 8: Configuring Task Worksheets*.

Chapter 7: Configuring Screens and Graphics

This chapter provides information on configuring screens and graphics. Before creating an application screen, you should consider the structure of the screen. Windows 2K/XP/Vista applications permit you to open more than one screen at a time, but for Windows CE applications you create a default screen with a header and footer (to use as a template), and then you insert objects on the screen. You then save the template screen under different names to create the different screens. Typically, an InduSoft Web Studio application screen consists of three basic areas (or screen types):

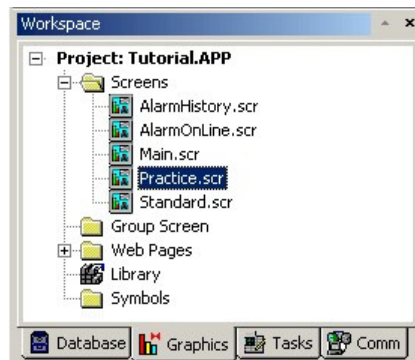
- **Header:** Objects located at the top of a screen to provide standard information (date, time, and so forth).
- **Footer:** Objects located at the bottom of a screen (typically an alarm object showing the last alarm).
- **Regular:** Area between the header and the footer to provide information about processes, alarm screens, trends, and so forth.

Using this structure to develop screens provides the following advantages:

- You can tie screens together according to their utility in the application.
- You can configure links and dynamics, common to all screens, just once.
- You can give the application a default format.
- You can build modular screens and use them in other projects.

Working in an Object-Oriented Environment

Select the **Graphics** tab to access all of the screens, Web pages, Library objects, and symbols in the application.



Graphics Tab

This tab contains the following folders:

- **Screens:** Contains all of the display screens created for the current application.
- **Group Screen:** Contains the entire screen groups (individual screens combined into manageable groups) created for the current application.

Note:
Screen groups are not available for Windows CE.

- **Web Pages:** Contains all the Web pages (screens saved in HTML format) created for the application.
- **Library:** Contains the library of common symbols and graphics provided with InduSoft Web Studio. Double-click the Library button to open the IWS Symbol Library utility, consisting of a list pane (containing all of the symbol groups) and a display screen.
- **Symbols:** Contains all of the user-defined symbols, which can be groups of images and/or text. You can create custom symbols for the application and save them into this folder.

Working with Screen Attributes

The *Screens* folder contains screens with finished graphic compilations and working drafts. To view a screen, expand the *Screens* folder and screens display to the right of the *Workspace* window. Double-click on a screen to open it.

To create a new screen, open the *Screen Attributes* dialog using one of the following methods:

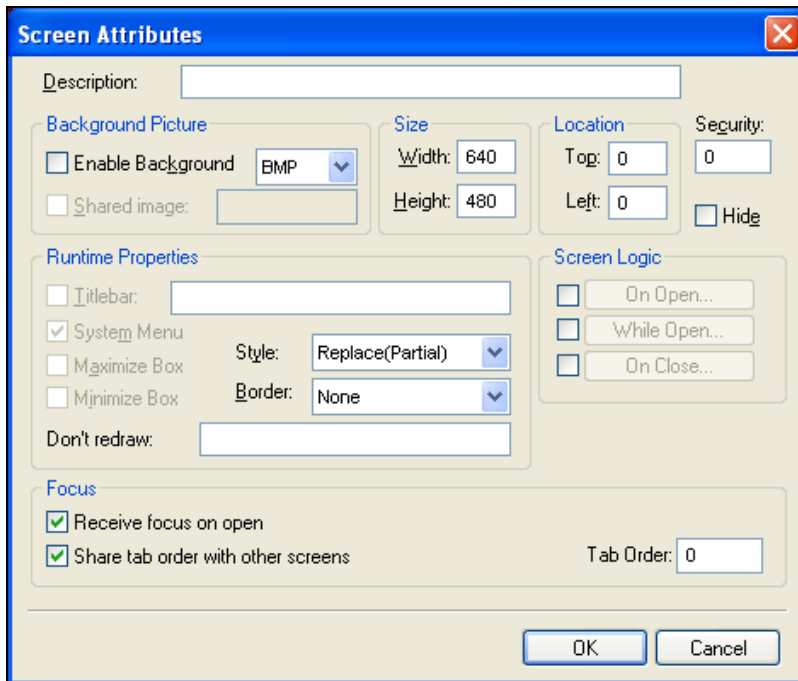
- ✓ Click the **New** button (📄) on the *Standard* toolbar. When the *New Document* dialog opens, click **Display**, and then click **OK**.
- ✓ Right-click on the *Screens* folder, select **Insert**, and click the prompt pop-up:



Creating a New Screen

- ✓ Select **File** → **New** from the menu bar. When the *New Document* dialog opens, click **Display**, and then click **OK**.

- ✓ Select **Insert** → **Screen** from the menu bar. When the *New Document* dialog opens, click **Display**, and then click **OK**.



Screen Attributes Dialog

To display the *Screen Attributes* dialog for an existing screen, select **View** > **Screen Attributes** from the menu bar.

Use the parameters on the *Screen Attributes* dialog as follows:

- **Description** text box: Type a description of the screen attribute for documentation purposes. The text you enter in this field displays in the status bar (by *default*) located at the bottom, left of the screen when you are in **Run Application** mode.
- **Background Picture** area: Specify the following parameters for the background.
 - **Enable background** check-box: Click (*check*) to enable or (*uncheck*) to disable (*default*) the use of background bitmaps.
 - **Enable Background** combo-box: Click to select one of the following Windows 2K/XP/Vista background options:
 - BMP
 - TIF
 - DXF
 - EPS
 - WMF
 - IMG
 - JPG
 - WPG
 - PCD
 - PNG
 - FMF
 - FPX
 - FAX
 - TGA

Note:
Windows CE supports **.BMPs** only.

- **Shared Image** check-box: Click (*check*) to save a **.BMP** file in a compressed form. Activated only when the **Enable Background** check-box has been selected.

⚠ Caution:

You cannot read the saved **.BMP** file in compressed format if you modify or install the Windows set-up values in an environment using a different number of colors. We recommend saving the screens in uncompressed format in case you want to switch among different configurations. In Windows CE, bitmaps must be 16-color.

- **Size** area: Type an integer number into the **Width** and/or **Height** boxes to specify the size (in pixels) of the selected window.
- **Location** area: Type an integer number into the **Top** and/or **Left** fields to specify the location of the window (in pixels) in relation to the current screen.

⚠ Caution:

If you use the mouse (pointer) to resize a window (**Thin** or **Resizing** style) directly, and then select **View > Screen Attributes** from the menu bar, IWS prompts you to update the current screen size and location.

- **Security** field: Specify the same window security level (*default is zero*) the user's access is defined using **Security** on the **Database** tab.
- **Hide** check-box: Click (*check*) to keep the screen loaded in memory after calling it for the first time, which facilitates faster loading when you open the screen. IWS executes **Screen Logics** normally.

Enabling this feature (*default is disabled*) causes a high use of GDI resources, consequently we recommend that during development, you monitor these resources using the **InfoResources** function.

- **Runtime Properties** area: Specify the following parameters to define the window properties when you are running the application.
 - **Titlebar**: Click this box to enable/disable a Titlebar for the new screen. Even if you do not check this check-box, you can configure text for the Titlebar; however, the Titlebar displays only if this option is checked.

⇒ Tips:

- You can modify the Titlebar text dynamically during the runtime by configuring tags between curly brackets. The value of the tag(s) is concatenated with the text in the Titlebar.
- Configuring dynamic text in the Titlebar is useful even if it is not visible on the screens. When the Print command is triggered from the Grid or Alarm/Event Control objects, the text from the Titlebar of the screen where the object is configured is printed in the header of the pages.

- **System Menu** check-box: Click (*check*) to enable the system menu.
- **Maximize Box** check-box: Click (*check*) to activate the **Maximize** button.
- **Minimize Box** check-box: Click (*check*) to activate the **Minimize** button.
- **Style**: Click to select a window style (*default is Replace*).
 - * **Overlapped** opens a window without closing any other window.
 - * **Popup** opens a window that remains in front of other windows, but leaves other windows enabled.
 - * **Dialog** opens a window that remains in front of other windows, but disables the other windows until you close the opened window.


- * **Replace** opens a window and closes any other *Replace* and *Popup style* windows.
- **Border:** Click to select a border style:
 - * **None:** No border and does not allow a title bar or resizing.
 - * **Thin:** Thin border window that cannot be resized during runtime.
 - * **Resizing (default):** Normal border that you can resized during runtime.
- **Don't Redraw** text box: Type a tag or value to control how screen dynamics are refreshed. Specifying a value higher than zero disables all screen dynamics.
- **Screen Logics** area: Click (*check*) the boxes to execute mathematical functions in one or more of these events: **On Open, While Open, On Close**.

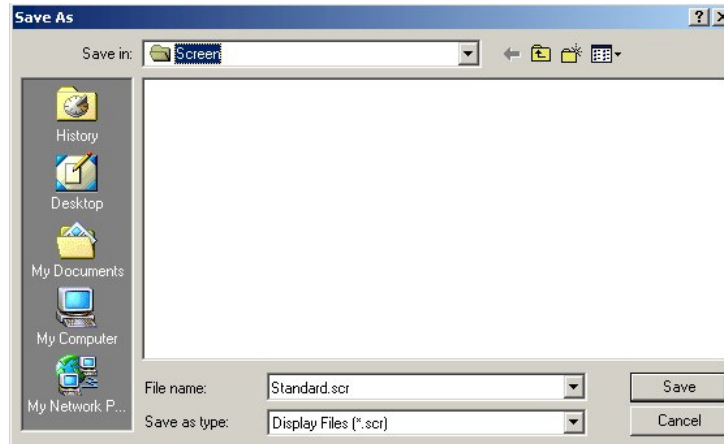
After enabling an event, click on the corresponding button to open a dialog where you can enter the following information:

 - * **Tag Name:** Type a tag name to receive a return value from the **Expression** column.
 - * **Expression:** Type a mathematical expression or function to be performed. The return value is applied to the **Tag Name** field.
 - * **Trigger (While Open dialog only):** Type a tag to work as a trigger (any value change) to execute this worksheet. If you leave this field blank, IWS executes the worksheet in the minimum time slice your system can perform.
- **Focus** area: Use the following options to set the behavior of the focus when using a group of screens (more than one screen open simultaneously):
 - * **Receive focus on open:** Check this option to bring the focus to the first object sensitive to focus, when it is opened.
 - * **Share tab order with other screens:** Check this option to switch the focus from objects from this screen to objects configured in other screens. This will occur after you press the Tab key while the last object sensitive to focus is in focus on this screen.
 - * **Tab Order:** Type a number (0 - 32767) to set the tab order. When there are more than one screen open, and you keep pressure on the Tab key (during the runtime), the focus switches from the screens with the smaller Tab Order number, to the screens with higher Tab Order numbers, and back to the screen with the lowest Tab Order number (and so on).

⇒ Tip:

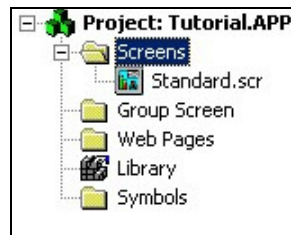
To set the order that the screens from the group will be opened when the group of screens is called, right-click on the screen group name from the Graphics tab of the workspace.

- Click the **Save** button () on the *Standard* toolbar or select **File** → **Save** (or **File** → **Save As**) from the menu bar to save the new screen. When the *Save* (or *Save As*) dialog displays, type **<filename>.scr** (or **<filename>**) into the **File name** field.




Specifying the File Name

- Expand the folders in the **Graphics** tab to see the saved screen:

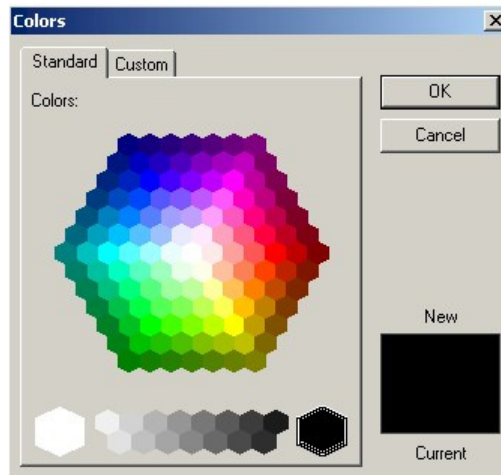


Checking for the New Screen

Specifying Background Color

To change the background color of the screen, click the **Background Color** button () located on the *Tools* toolbar or right-click on the blank screen and choose **Background Color** from the pop-up menu.

When the *Color* dialog displays, click a color, and then click **OK**.



Color Dialog

⇒ **Tip:**
Double-click on a color to select that color and automatically close the *Color* dialog.

✎ **Note:**
Depending on the palette configured for your system, the colors of graphic objects imported into the environment may have color distortion. If this happens, change the palette configured for your system.

Using Objects and Dynamics

The InduSoft Web Studio product provides full-featured objects and dynamics (the ability to modify object properties, execute commands, or inset values to tags used to build screens on the fly at runtime).


Using the Mode Toolbar

The *Mode* toolbar provides buttons for general screen editing.



Mode Toolbar


The *Mode* toolbar contains the following buttons:

- **Selection** button (- *Objects* layer: The layer on which you create the dynamic objects for your screen.
- *Background Picture* layer: The static background layer of the same screen.
- When the *Background Picture* layer is active, the *Bitmap* toolbar displays automatically.




Note:

You automatically disable the **Bitmap Editor** button when you uncheck (*disable*) the **Enable Background** check-box (.BMP-type only) on the *Screen Attributes* dialog.

- **Fill Color** button (- 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.

- **Fonts** button (

⇒ **Tip:**

To save development time, select several *text* objects and use the **Fonts** button to specify font and color settings for all of the objects at once. (You cannot use this function for grouped *text* objects however.)

- **Line Color** button (


When you click the **Line Color** button, the *Line Selection* dialog is displayed. Use this dialog to specify line styles and color for the selected objects:



Line Selection Dialog


⇒ **Tip:**

To save development time, you can select several of the preceding objects and use the **Line Color** button to specify a line color for all of the objects at once.

- **Background Color** button (

🔗 **Note:**

You automatically disable this button when you check the **Enable Background** check-box on the *Screen Attributes* dialog.

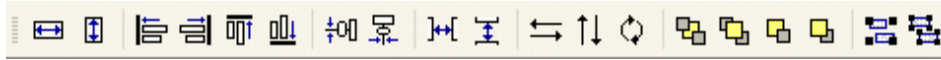
- **Grid** button (

⇒ **Tip:**

You can use the *Grid* dialog to configure the default settings for a grid. To open this dialog, right-click on the screen and select **Grid Settings** when the pop-up menu displays.



Using the Align and Distribute Toolbar

The *Align and Distribute* toolbar provides buttons that allow you to edit screen objects.



Align and Distribute Toolbar

Use the following toolbar options for resizing:

- **Resize height** button (): Click to set the height of all selected objects to the height of the last object selected (the object with the filled handles). You can use **Resize height** to resize one object by setting its height equal to its width.
- **Resize width** button (): Click to set the width of all selected objects to the width of the last object selected (the object with the filled handles). You can use **Resize width** to resize one selected object by setting its width equal to its height.

⇒ Tip:

You can use **Resize width** and **Resize height** to create circles from an ellipse or squares from rectangles. Select only one object before using these tools.

You also can use your cursor, mouse, and keyboard arrow buttons to resize objects on your screen. When you select an object (or group of objects) with the cursor, selection handles (black squares) display at each corner and at the midpoint of each side. You can use these handles as follows:

- **To enlarge an object**, click on a handle and drag your cursor (or pointer) in the direction indicated by the arrows that display. Clicking and dragging a corner resizes the entire object (height and width, while clicking on a side resizes the object in one direction only (height only or width only).
- **To enlarge an object with finer resizing control**, click on a handle and do not release the left mouse button. Click the arrow keys to resize the object (in the direction indicated by the resizing arrows) one pixel at a time. Release the mouse button when you are finished resizing the object.
- **To select and resize an open or closed polygon**, draw a selection box around the polygon and group it (as described in “Object Grouping and Ungrouping Buttons”, page 7–16). You can then click on a handle and drag it to resize the object.
- **To change the shape of an open or closed polygon**, click on a handle and a boxed square displays at the base of your cursor. Drag the handle to move its position and change the shape of the polygon.

📌 Notes:

- All objects with dynamic properties and *Group of Symbols* objects (including most symbols and library objects) have multiple *Object Properties* dialogs and properties. Use the drop-down list on the *Object Properties* dialog (**View** → **Object Properties**) to access these different dialogs and properties.
- If you resize a symbol or group of objects, IWS resizes all objects within the symbol or group accordingly.


OBJECT ALIGNMENT BUTTONS

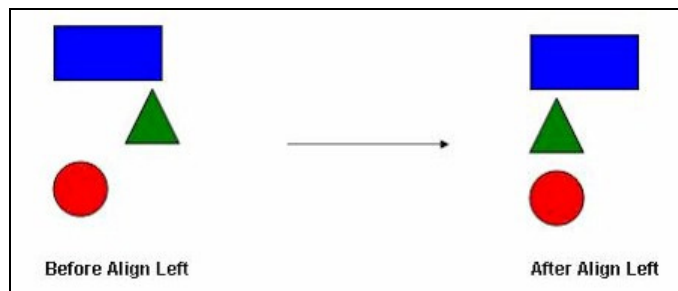
When you select a series of objects (two or more), you can align those objects based on the location of the last object selected. As you select objects, solid handles display on the last object selected, and the handles on all previously selected objects become empty (unfilled) boxes.

 **Note:**


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

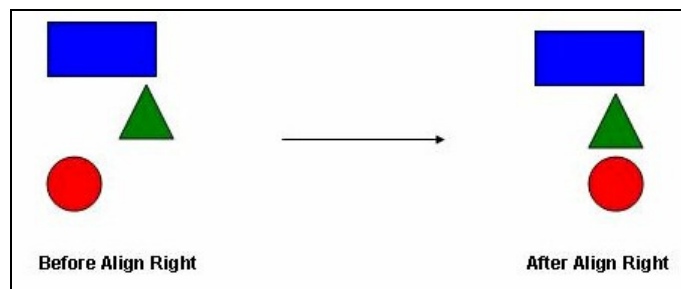
Use the following alignment buttons to align a series of objects.

- **Align left** button (): Click to align all selected objects to the left edge of the last object selected. For example:




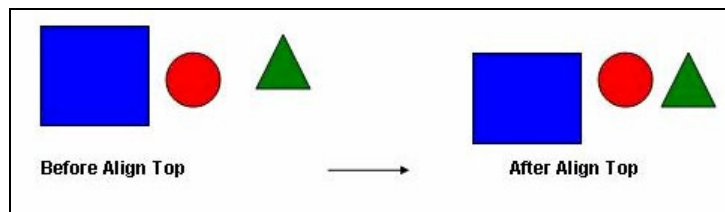
Aligning Objects Left

- **Align right** button (): Click to align all selected objects to the right edge of the last object selected. For example:

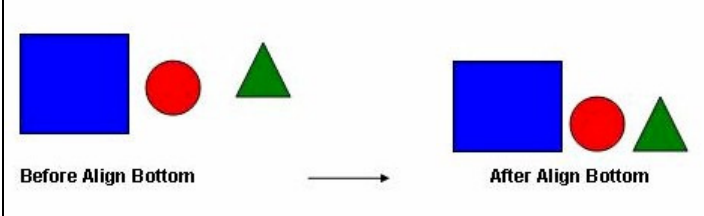



Aligning Objects Right

- **Align top** button (): Click to align all selected objects to the top edge of the last object selected. For example:



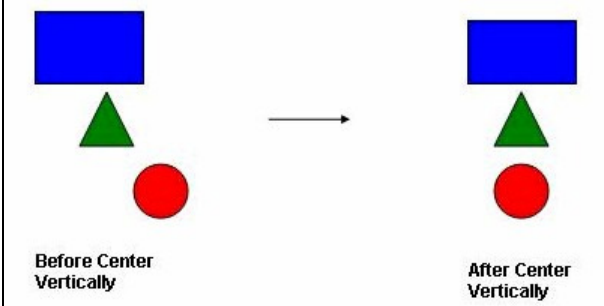

Aligning Object Tops

- **Align bottom** button (

Before Align Bottom

After Align Bottom

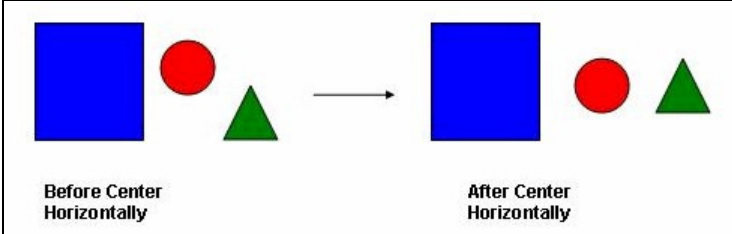

Aligning Object Bottoms

- **Center vertically** button (

Before Center Vertically

After Center Vertically


Centering Objects Vertically

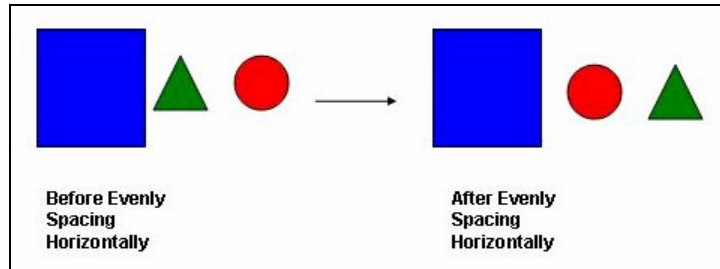
- **Center horizontally** button (

Before Center Horizontally


After Center Horizontally

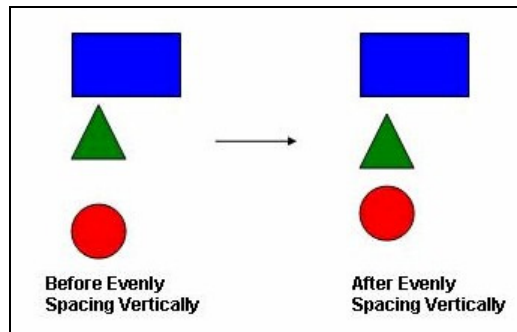
Centering Objects Horizontally

- **Evenly space horizontally** button (): Click to put an equal amount of horizontal space between a series of objects (two or more). For example:



Spacing Objects Horizontally

- **Evenly space vertically** button (): Click to put an equal amount of vertical space between a series of objects (two or more). For example:




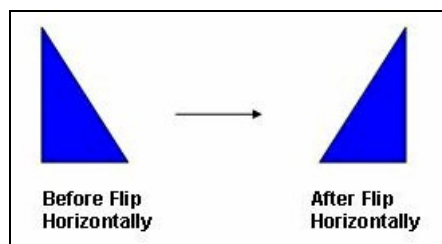
Spacing Objects Vertically

 **Note:**


The spacing functions may move the last object selected (with solid handles) by no more than a few pixels to equally space all of the objects.

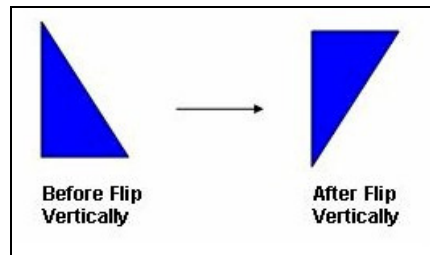
Use the following buttons to change the orientation of a single selected object or grouped object. (You cannot use these buttons with multiple objects selected.)

- **Flip horizontally** button (): Click 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:




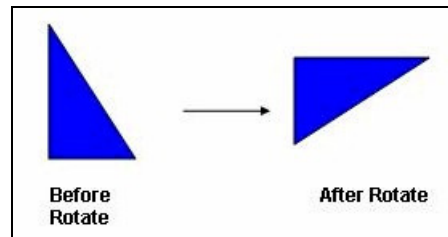
Flipping Objects Horizontally

- **Flip Vertically** button (): Click 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 example:



Flipping Objects Vertically

- **Rotate** button (): Click to rotate the selected object 90 degrees (a quarter turn) clockwise. For example:




Rotating Objects

CHANGING OBJECT LAYERS BUTTONS

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


Notes:


- 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.
- 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.

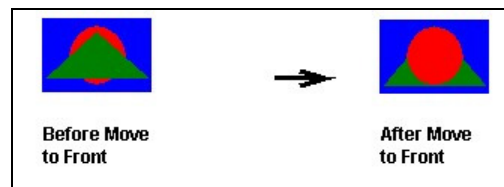
- **Move to back** button (): Click to move a selected object or group behind all other objects on the screen. IWS assigns the object the lowest ID# and moves that object behind all other objects on the screen. For example:




Moving Objects to Back (Selecting the Blue Rectangle)



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

- **Move to front** button (): Click to move a selected object or group 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. For example:



Moving Objects to Front (Selecting the Red Circle)

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


- **Move backward** button (): Click 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 its pop-up menu.) IWS assigns the selected object the next available ID# less than the object behind which it was moved.
- **Move forward** button (): Click 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 its pop-up menu.) IWS assigns the selected object the first available ID# greater than the object in front of which it was moved.


OBJECT GROUPING AND UNGROUPING BUTTONS

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


 **Note:**


All objects with dynamic properties and *Group of Symbols* objects (which include most symbols and library objects) have multiple *Object Properties* dialogs and properties. You can use the drop-down list on the *Object Properties* dialog (**View** → **Object Properties**) to access these different dialogs and properties.

- **Group** button (

 **Note:**

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

- **Ungroup** button (

 **Notes:**

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

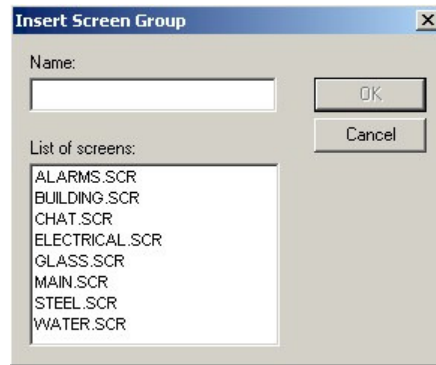
Using Groups of Screens

IWS provides a screen-grouping feature that allows you to open a linked set of screens at the same time.

 **Note:**

This feature is not supported for CView applications or for screens that will be exported in HTML format.

From the menu bar, select **Insert > Screen Group** to open the *Insert Screen Group* dialog, where you can create a new group of screens for your application.



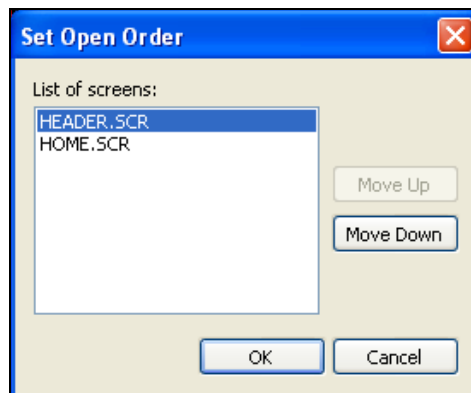
Insert Screen Group Dialog

To create a new screen group, type a group name into the **Name** text box or click on a screen provided in the **List of screens**. Click on **OK** when you are done.

Note:

- You also can create a new screen group from the **Graphics** tab in the *Workspace*. Right-click on the *Group Screen* folder then select **Insert screen group** from the pop-up menu. The *Group Screen* folder combines individual display screens from the *Screens* folder into more manageable groups.
- Select **File > Save Screen Group As HTML** to save the *Screen Group* in HTML format, making them available to the remote Web Thin Client through a Web Browser.

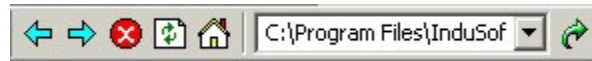
To set the order that the screens from one group should be opened when the group is called, right-click on the screen group name (from the Graphics tab of the Workspace). Click on Set Open Order in the pop-up menu.



Set Open Order Dialog

Using the Web Toolbar



The *Web* toolbar provides buttons that allow you to open and navigate HTML files.



Web Toolbar

Note:

You must install Internet Explorer v4.1 (or higher) before you can use any of the tools on the *Web* toolbar.

- **Back** button (): Type a Web page URL address into the text box to open (download) that page to your *Internet Explorer* Web browser.
- **Go** button (

SAVING SCREENS IN HTML FORMAT

IWS also enables you to save screens in the HTML format. You cannot create the HTML pages contained in the *Web Pages* folder; instead, they are generated from pre-existing screens. For information on how to configure and run a Web Thin Client application, see *Chapter 13: Configuring a Web Solution*.

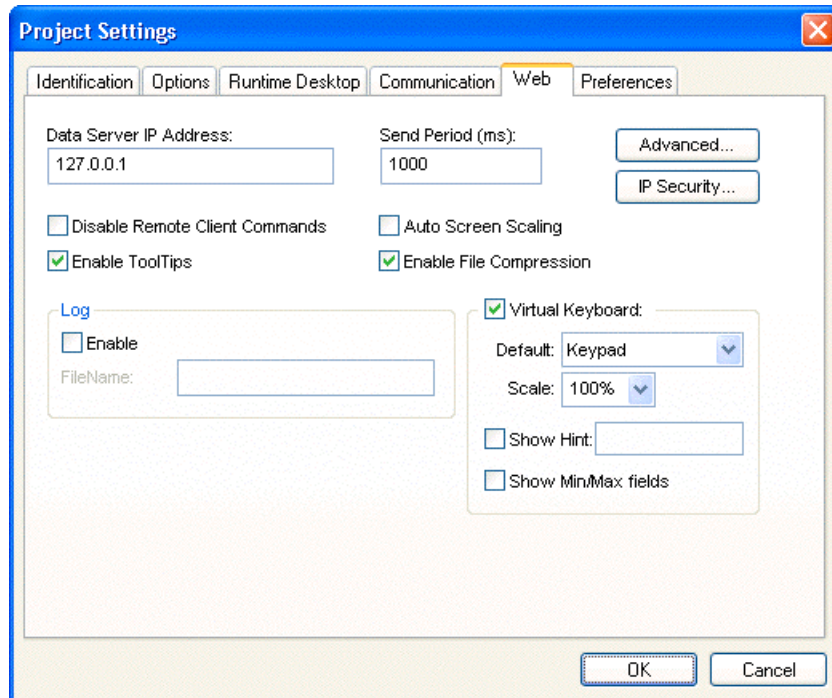
To create an HTML page, you must first create a display screen. Configure a screen as you usually would (create objects, add properties, and so on), but keep in mind that this screen will become a Web page. Save the screen as usual when you finish. Then, with the screen still open, select **File > Save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

Caution:

The Web pages that are generated when you select **File > Save As HTML** are independent of the screen file from which they were generated. Consequently, if you change that display screen, the changes will not appear on the Web page until you select **File > Save As HTML** again.

Select **File > Save Screen Group As HTML** to save the *Screen Group* in HTML format, making them available to the remote Web Thin Client through a Web Browser.

To view your Web pages, you must first configure the Web settings on the *Project Settings* dialog (**Web** tab).



Project Settings Dialog: Web Tab

- ✔ Open the dialog and type the IP address (from which to run the application) in the **Data Server IP** field. 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 Web Thin Client will assume that the Web Server (i.e. the address entered into the browser) is also the Data Server.

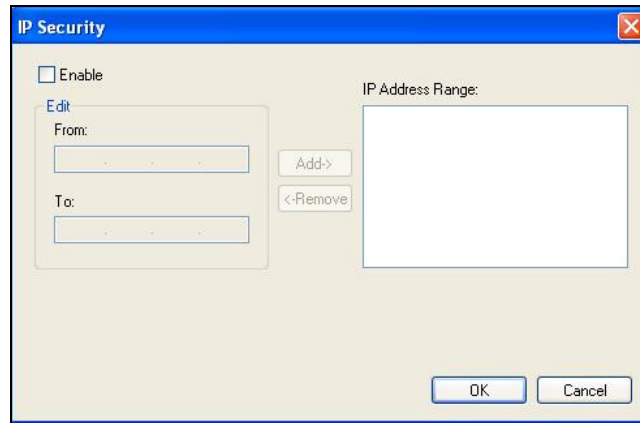
⇒ **Tip:**

You can use the IP address **127.0.0.1** (localhost) to access the TCP/IP server on the local computer (regardless of its IP 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.

- ✔ Type a value in the **Send Period** field to specify the send period (in milliseconds) used to exchange data between the Server and the Web Thin Client stations.
- ✔ Click (*enable*) the following check-boxes if applicable:
 - **Disable Remote Client Commands** check-box: Click (*enable*) this box to prevent a remote client from issuing commands from your Web Thin Client to your Server.
 - **Enable ToolTips** check-box: Click (*enable*) this box to see *Windows ToolTips* when viewing the application screens on the Web Thin Client (browser).
 - **Auto Screen Scaling** check-box: Click (*enable*) this box to automatically scale screens displayed in a Browser window.

Enable this parameter if you are running remotely on a Web Thin Client, and you want IWS to scale screens automatically when you resize the Browser window.

- **Enable File Compression** check-box: Click (*enable*) this box to compress the files stored on the **\web** folder of the application. This option is useful for reducing download time, particularly if you have a slow connection between your Server and the Web Thin Client.
- ☑ Click the **IP Security** button to open the *IP Security* dialog. Use the parameters on this dialog to specify the range of IP addresses for the computers that are allowed to access the application as Web 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 applications as Web Thin Clients.



Note:

By default, IP security applies only to Web Thin Clients connecting to the Data Server. You can also implement IP security for database synchronization between applications running on different stations (see “Configuring TCP/IP” on page 10–25). To do this, insert the following parameter into the application file (*.app):

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

- ☑ To enable logging for the Web Thin Client, move to the *Log* pane and click (*enable*) the **Enable** check-box and type a file name into the **Filename** field to generate a log file on the Web Thin Client station. You can use this log file for debugging purposes.
- ☑ To enable the Virtual Keyboard for the Web Thin Client stations, independently from the local station, click the **Virtual Keyboard** check-box. You can establish a default configuration for the virtual keyboard:
 - **Default:** Select the default keyboard type to be used in the application, 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 the application will attempt to scale the input accordingly.

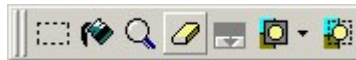
- Click **OK** to close the *Project Settings* dialog.

Notes:

- If you change any of the Web information on the *Project Settings* dialog, you must re-verify the application for the new setting to take effect. To verify the application, select **Tools** → **Verify** from the menu bar. (If you have any windows open in the development system, IWS prompts you close them before you verify the application).
- Because the Web pages display information from the application through the Web server, you must be running the runtime system, Web server, and the TCP/IP server to view your Web pages.




Using the Bitmap Toolbar

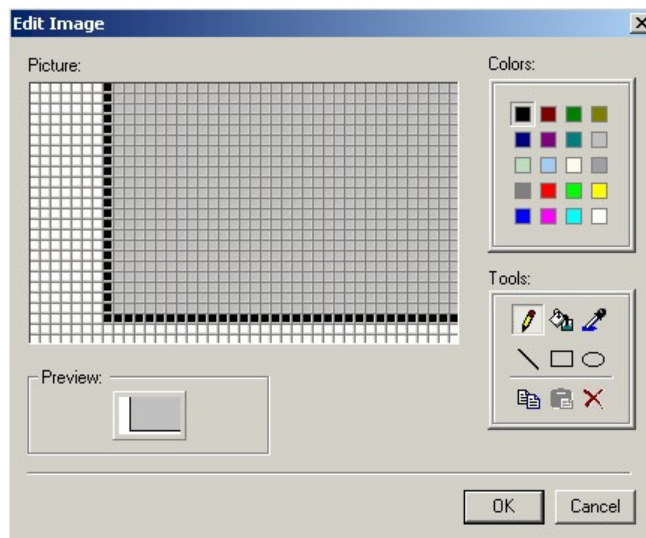
Use the *Bitmap* toolbar to access the *Bitmap Screen Editor* tools. (This toolbar is available only when the *Background Picture* layer is active. You can enable the *Background Picture* layer in the *Screen Attributes* dialog.)





Bitmap Toolbar

The *Bitmap* toolbar contains the following buttons:

- **Select Area** button (): Click to select an area within the *Bitmap Screen Editor*.
- **Flood Fill** button (): Click the **Flood Fill** button, and then click on the screen to paint the surrounding area with the color you specified with the **Fill Color** button.
- **Pixel Editing** button (): Click to open an *Edit Image* dialog, where you can draw detailed bitmaps, pixel by pixel.





Edit Image Dialog

- **Erase Area** button (): Click to remove a selected area from the screen.
- **Change colors** button (): Click to change the transparent fill color for a selected area.

 **Note:**

Before you can use this button, you should have already specified a fill color (**Fill Color** button), selected a transparent color (**Select Transparent Color** button), and defined the area to fill (**Select Area** button).

- **Select Transparent Color** button (): Click to specify a transparent color (referenced by the **Change Colors** button).

- **Toggle Transparent Color** button (

⇒ **Tip:**

You can use the **Copy (Ctrl+C)** and **Paste (Ctrl+V)** commands to exchange bitmap pictures between IWS's *Bitmap Screen Editor* and any other bitmap editor (*Paint Brush*, for example).


Using the Static Objects Toolbar

The *Static Objects* toolbar provides buttons you can use to create polygons, rectangles, lines, and other objects for your screen.



Static Objects Toolbar

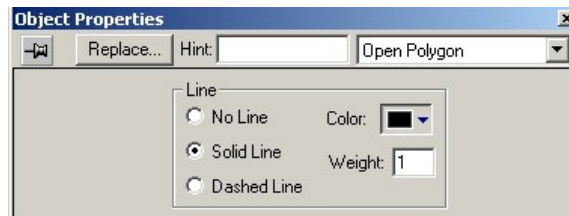
The *Static Objects* toolbar contains the following buttons:

- **Open Polygon** button (): Click 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 view the object properties, double-click on the polygon object and the *Object Properties* dialog is displayed:

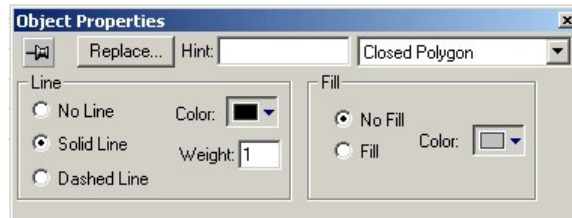


Object Properties Dialog: Open Polygon

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** button (): Click to draw a closed polygon, using a border in the specified foreground color.
- To draw a closed 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 point.
 - Repeat this process until you create the desired polygon shape.
 - Double-click or right-click to stop drawing the polygon.

To view the object properties, double-click on the polygon object. The *Object Properties* dialog is displayed:




Object Properties Dialog: Closed Polygon

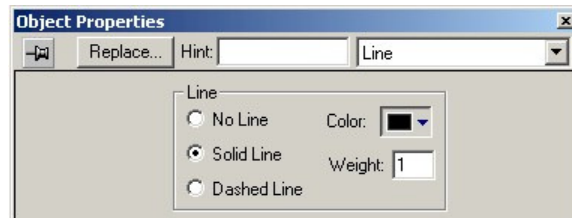
Use the *Object Properties* dialog to specify the following parameters for the polygon:

- **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.

- **Line** button (): Click to draw an orthogonal line in the drawing area, as follows:
 - Click the left mouse button to set the starting point of the line.
 - Drag the cursor to adjust the line size.
 - Click again to place the object.


To view the object properties, double-click on the object. The *Object Properties* dialog is displayed:

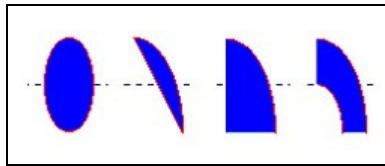


Object Properties: Line

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.

- **Ellipse** button (): Click to draw ellipses, chords, arcs, and rings. For example:



Oval, Chord, Arc, and Ring

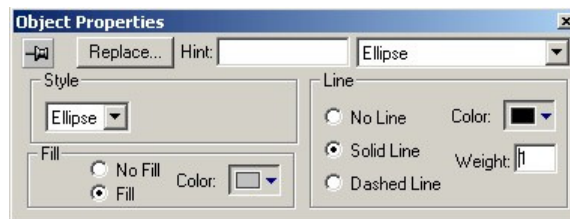
⇒ **Tip:**

The *Ring* style is particularly useful when you are creating plumbing drawings.

To create an ellipse, use the following steps:

- Click in the drawing area and drag the mouse/cursor to create an oval shape.
- Release the mouse button to stop drawing the oval.
- Use the *Object Properties* dialog to change the shape to a chord, arc, or ring.

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** rectangle. 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, and 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.

- **Rounded Rectangle** button (): Click to draw rounded rectangles (empty or filled), as follows:

- Click in the drawing area and drag the mouse/cursor to create the rectangle.
- Release the mouse button to stop drawing the object.

Double-click on the object to view the *Object Properties* dialog:




Object Properties: Rounded Rectangle

Notes:

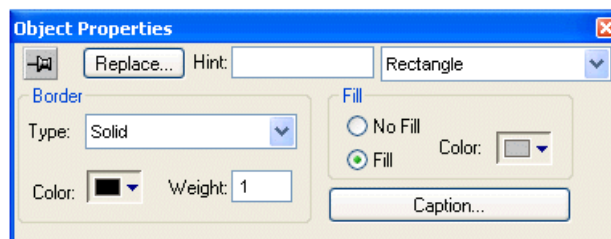
- You cannot use the rounded rectangle button to create a bar graph for Windows CE applications.
- A rounded rectangle has one extra tracker in the lower left corner, which enables you to modify the arc angle.

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, and then close the dialog.
- **Caption:** This option is not enabled for this object.

- **Rectangle** button (): Click to create rectangles, as follows:
 - Click in the drawing area and drag the mouse/cursor to draw the rectangle.
 - Release the mouse button when the rectangle is the size you want.

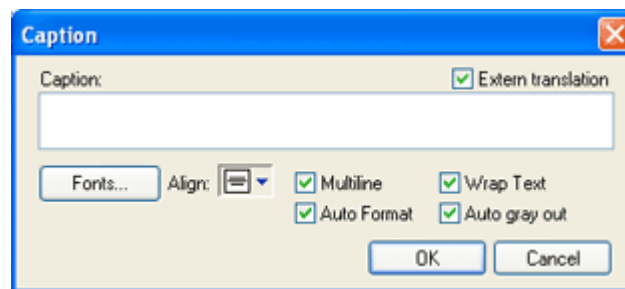
Double-click on the object to view the *Object Properties* dialog:




Object Properties: Rectangle

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

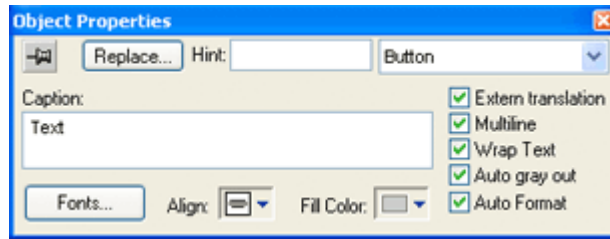
- **Type:** Specify a border line style by clicking on **None, Solid, Dashed, Etched, Raised** or **Sunken**.
- **Color:** Specify a border line color by clicking the **Color** button to open the *Color* dialog. Click the color to select it, and then close the dialog.
- **Weight:** Specify a border line 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, and 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.



Selecting the Caption Button


- * **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 }**).
 - * **Extern translation:** Click (check) to enable translation during runtime using the Translation Tool.
 - * **Fonts:** Specify a font style for the caption by clicking the **Fonts** button.
 - * **Align:** Specify the alignment for the caption of the rectangle.
 - * **Multiline:** Allows the caption of the rectangle to be shown in more than one line, when checked.
 - * **Wrap Text:** When checked, the object automatically wraps the text when necessary.
 - * **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.
 - * **Auto gray out:** Turns the caption of the rectangle to gray when the *Command* dynamic applied to the rectangle is disabled by the **Disable** field or due to the Security System.
- **Button** button (): Click to create custom-sized buttons, as follows:
 - Click in the drawing area and drag the mouse/cursor to create the button shape.
 - Release the mouse button when the button is the size you want.

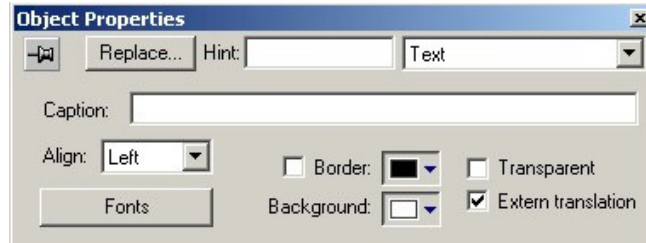
Double-click on the object to view the *Object Properties* dialog:



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* }).
 - **Fonts:** Specify a font style for the caption by clicking the **Fonts** button. When the *Fonts* dialog displays, specify the following parameters:
 - * **Font** (typeface)
 - * **Font style**
 - * **Size**
 - * **Effects**
 - * **Color**
 - * **Script style**
 - **Weight:** Specify a border line width by typing a number representing the line width (in pixels) into the text box.
 - **Align:** Specify the alignment for the caption of the button.
 - **Fill Color:** Specify the Fill Color for the button.
 - **Extern translation** (*optional*): Specify an external translation file for the button label by clicking (*checking*) the box.
 - **Multiline:** Allows 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* dynamic 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.
- **Text** button (): Click to create text objects, as follows:
 - Click in the drawing area. When a cursor displays, you can type a line of text.
 - After entering the text string, double-click on the new text object to view the *Object Properties* dialog.



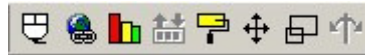
Object Properties: Text

Use the *Object Properties* dialog to specify the following orthogonal line parameters:

- **Caption:** Specify a text string by typing a caption in the text box.
- **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.
- **Transparent** check box: Enable (check) this option to not show the background color. Disable (uncheck) to show the background color.
- **Extern translation** (*optional*): Specify an external translation file for the text by clicking (*checking*) this box.


Using the Dynamic Properties Toolbar

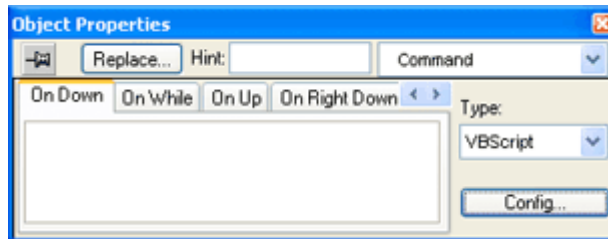
Use the *Dynamic Properties* toolbar to apply *dynamics* to objects or a group of objects. Dynamics enable you to modify object properties on the fly (during runtime) according to tag values. Some dynamics also enable you to execute commands or insert values (set points) to the tags.



Dynamic Properties Toolbar

The *Dynamic Properties* toolbar contains the following buttons:

- **Command** property (): Click to add the command dynamic to a selected object or group of objects. The command dynamic enables you to click on the object or press a pre-defined key to execute the command at runtime.
 - Double-click the object to view its object properties.



Object Properties: Command

The Command dynamic 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 dynamic:

Event	Description
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.

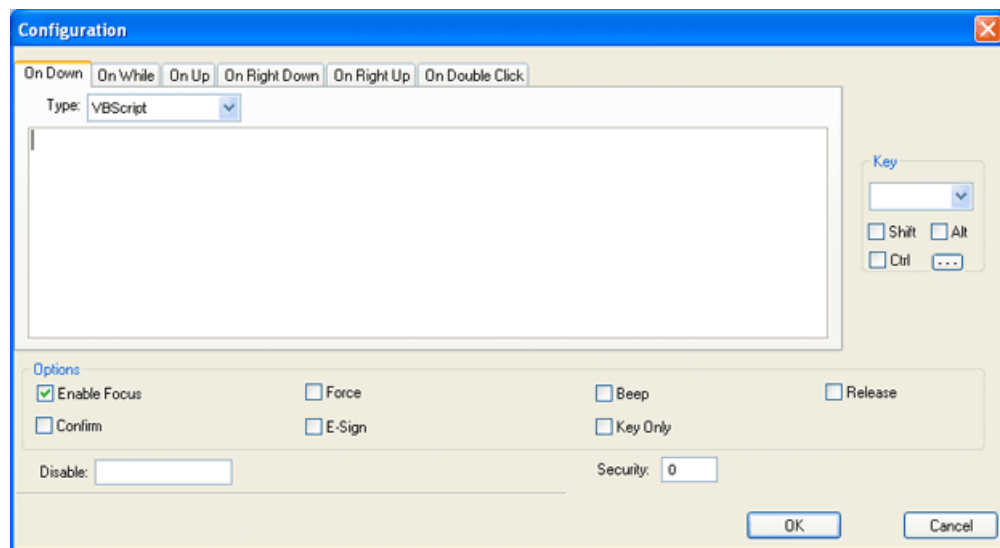
- Notes:**
- IWS treats the touch-screen actions the same way it treats the mouse pointer actions. In other words, it is transparent for IWS if an event was triggered by a touch-screen interface or by a regular mouse pointer.
 - The events On Right Down, On Right Up and On Double Click are not supported by CEView applications (running on the Windows CE operating system).
 - When creating an application for a touch-screen device, it is important to keep in mind that events On Right Down and On Right Up cannot be triggered on such devices.

- **Type:** This setting defines the type of action that must be executed by the event of the Command dynamic. Notice that each event has its own type. Therefore, the same Command dynamic 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 IWS Built-in Language chapter for further information about this the IWS Built-in Language.
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 dynamic. Consult the VBScript chapter for further information about the VBScript language.
Open Screen	Allows you to configure the Command dynamic to open a specific screen when the event is triggered during the runtime. This type is equivalent to the IWS built-in function Open(). 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, IWS will attempt to open the named screen.
Close Screen	Allows you to configure the Command dynamic to close a specific screen when the event is triggered during the runtime. This type is equivalent to the IWS built-in function Close(). 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, IWS will attempt to close the named screen.
Set Tag	Allows you to configure the Command dynamic to set a tag when the event is triggered during the runtime. You can either type the tag name in the Set Tag field or browse it. When the event is executed, IWS will write the value 1 to the tag configured in this field.

Type	Description
Reset Tag	Allows you to configure the Command dynamic to reset a tag when the event is triggered during the runtime. You can either type the tag name in the Reset Tag field or browse it. When the event is executed, IWS will write the value 0 to the tag configured in this field.
Toggle Tag	Allows you to configure the Command dynamic to toggle a tag when the event is triggered during the runtime. You can either type the tag name in the Toggle Tag field or browse it. When the event is executed, IWS will toggle the value of the tag configured in this field.

- **Config:** Launches the Configuration dialog screen, where the Command dynamic can be fully configured. This dialog allows you to configure the Command settings, as follows:



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:

- **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 applications for runtime devices that do not provide a mouse or touch-screen interface — the keyboard is the only physical interface available to interact with the application 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.

- **Options** pane:
 - o **Enable Focus** checkbox: When this option is checked, the object that the Command dynamic was applied to can receive the focus during runtime by the navigation keys.
 - o **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 dynamic overwrites the same value 0 to this tag, any other task in IWS will recognize that this tag changed value (even if it did not) after executing the dynamic. 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 dynamic is executed.

 **Note:**

For applications created with InduSoft Web Studio version 6.1 SP3 or earlier, the **Force** option is enabled by default and cannot be disabled.

- o **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 dynamic was successful or not.
- o **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.
- o **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.
- o **E-Sign** checkbox: When this option is checked, the user will be prompted to enter the Electronic Signature before executing the command.
- o **Key Only** checkbox: When this option is checked, the user can *only* use the keyboard shortcut (configured in the *Key* pane described above) 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 dynamic.

- **Hyperlink** button (): Click to add the hyperlink property to a selected object or group of objects. Applying this property 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. IWS uses this protocol when it loads the URL.
- **E-Sign** check-box: When this option is checked, the user is prompted to enter the Electronic Signature before executing the dynamic.
- **URL** field: Type the URL address you want to load (for example: InduSoft.com).

⇒ **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, IWS 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 property for the selected object(s).
- **Security** field: Type a value into this field to specify a security level for the object(s). If a user logs on, and does not have the required security level, IWS disables the hyperlink command for the object(s).

- **Bargraph** button (

Object Properties: BarGraph


Note:
You cannot use the rounded rectangle button to create a bar graph for Windows CE applications.

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

- **Tag/Expression** field: Type a tag or expression that determines the bar graph level. You also can click the button 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 size 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 size of the bars.

Tip:
IWS 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 the database.

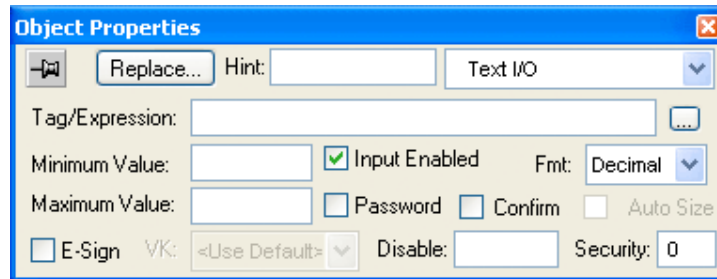
- **Foreground Color:** To specify a fill color for the bars, click the combo-box button. When the *Color* dialog displays, click on a color to select it, and then close the dialog.
- **Direction** pane: Click the **Vertical** or **Horizontal** radio button to specify the direction of the bar graph.
- **Orientation** pane: Click the **Up**, **Center**, or **Down** button to specify the orientation of the maximum and minimum values when drawing the bars.

- **Text I/O button** (

Note:

You can apply this dynamic property only to text objects containing the # character (each # represents one character) to be replaced by a value during the runtime.

Double-click on the object to open the *Object Properties* dialog. You can use this dialog to specify the following parameters:



Object Properties: Text I/O

- **Tag/Expression** text field: Type one of the following into the field:
 - * A tag on which to perform an input or output operation; or
 - * An expression on which to perform an output operation only.
 You can click the button to browse your directories for an existing tag or expression.
- **Input Enabled** check-box: Enable (*check*) this option to allow data entries. Disable (*uncheck*) the option and this dynamic only executes the data outputs.
- **Fmt** 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.


Note:

This option does not actually change the specified tag or expression in any way.

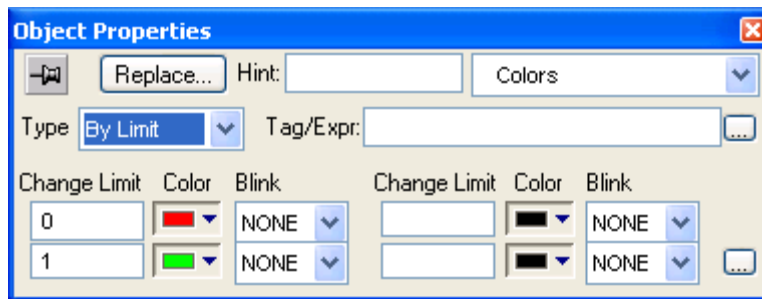
For example, **Tag/Expression** is set to a tag of Integer type, **Input Enabled** is checked, and **Fmt** is set to **Hexa**. You may input a new value in hexadecimal format, but it is saved in the application database as an integer.

- **Minimum Value** field: Enable (*check*) this option to define 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: Enable (*check*) this option to define 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.

- **Confirm** check-box: Enable (*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).
- **Password** check-box: Enable (*check*) this option to hide password text entries by replacing the text with asterisks (*).
- **E-Sign** check-box: When this option is checked, the user is prompted to enter the Electronic Signature before changing the tag value.
- **VK** field: Virtual Keyboard type used for this object. You need to enable the **Virtual Keyboard** option on the **Project > Settings > Runtime Desktop** interface 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).

Colors button (): Click to add the color change property to a selected object. The Colors dynamic allows you to modify the color of a static object during the runtime based on the value of a tag or expression.

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



Object Properties: Colors

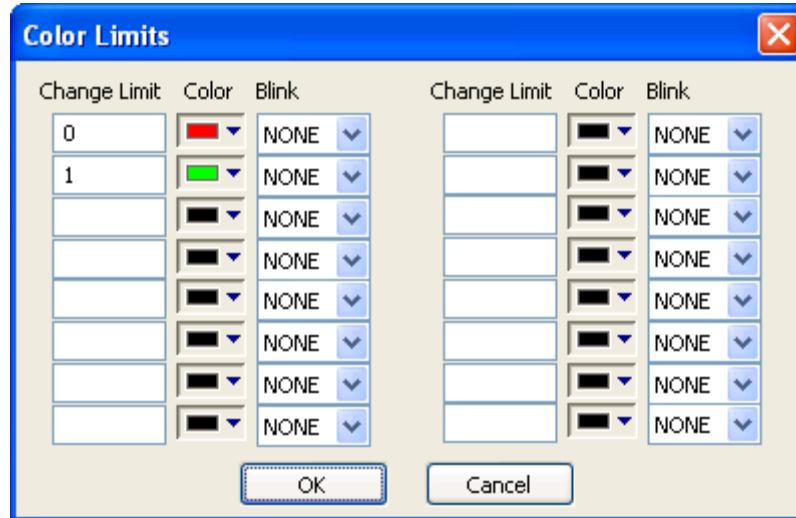
You can use this dialog to specify the following parameters:

- **Type** field: Determines the mode in which this dynamic works:
 - * **By Limit**: When selecting this type, you can specify up to four limits (Change Limit) for this dynamic 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 *IWS Development Environment -> Standard Interfaces -> 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 window: 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 dynamic.



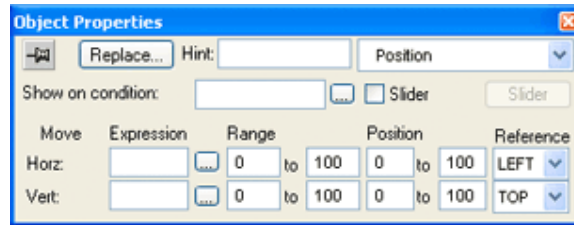
Color Limits Dialog Box

- **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.

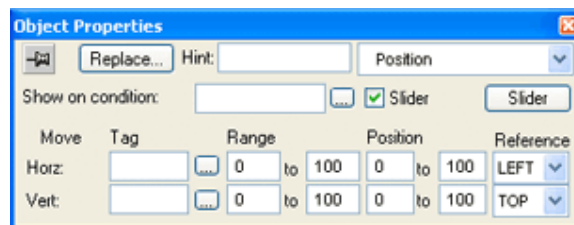
Note:
The following fields are automatically disabled (grayed out) when Type = By Color: **Change Limit**, **Color** and **Blink**.

- **Position** button (): Click to specify when and where to display an object, using the specified tag values.

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



Object Properties: Position (with Slider not enabled)



Object Properties: Position (with Slider enabled)


You can use this dialog to specify the following parameters:

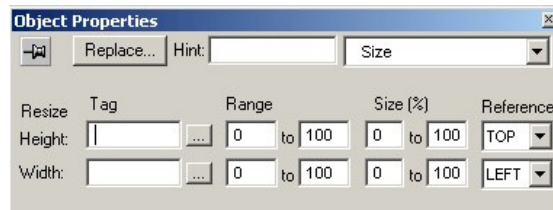
- **Show on condition** field: Configure a tag/expression in this field to control when the object is shown. When the value of the tag/expression is TRUE (i.e. greater than 0), the object is shown. When the value is FALSE (i.e. less than or equal to 0), the object is hidden. You can also leave the field blank to always show the object.
- **Slider** checkbox: Click (check) this option to operate the object like a slider, which means that you can drag the object around the screen to input values to the configured tags. (See *Move* below.)

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

- **Disable** field: Configure a tag/expression in this field to control when sliding is disabled. When the value of the tag/expression is TRUE (i.e. greater than 0), sliding is disabled. When the value is FALSE (i.e. less than or equal to 0), sliding is enabled.
- **Security** field: 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** fields: The meaning of these fields changes depending on whether the **Slider** option is enabled (see above).
 - **Expression** field: If the **Slider** option is not enabled, then configure tags/expressions in these fields to determine the horizontal and vertical positions of the object on the screen.
 - **Tag** field: If the **Slider** option is enabled, then configure *only* tags in these fields to receive input values as the user drags the object around the screen.
 - **Range** fields: Enter the minimum and maximum possible values for the configured tag/expression. Values outside of the range are ignored.

- **Position** fields: Enter values in these field to specify how much change in position (in pixels) you can move an object on the screen according to the established condition. Positive values move the object right and down; negative values move the object left and up.
The actual position of the object is proportional to the value of **Tag/Expression** within **Range**.
- **Reference** combo-box: Select one of the following options as a reference point, to be used while moving the object on the screen. Specifying this option is necessary only if you want to resize the object as you move it.
 - **Left**: Left corner of the object.
 - **Right**: Right corner of the object.
 - **Center**: Center of the object.
 - **Top**: Upper corner of the object.
 - **Bottom**: Lower corner of the object.


- **Resize** button (): Click to increase or decrease the size of a selected object or symbol. Double-click on the object/symbol to open the *Object Properties* dialog:

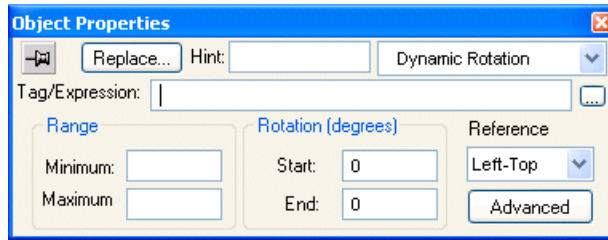


Object Properties: Resize

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

- **Tag** fields: Type values in the **Height** and **Width** fields to increase or decrease the object's horizontal and vertical size.
- **Range** fields: Type values to specify upper and lower tag limits, which IWS uses to increase and decrease the object size.
- **Size (%)** fields: Type values to specify a percentage range, which IWS uses to increase and decrease the object size.
- **Reference** combo-boxes: Select one of the following reference points to determine how the object increases size horizontally and vertically.
 - * **Left**: From the left side
 - * **Right**: From the right side
 - * **Center**: Horizontally and vertically from the center of the object
 - * **Top**: From the upper side
 - * **Bottom**: From the lower side

- **Dynamic Rotation** button (): Click to rotate a line. Double-click on the line to open the *Object Properties* dialog:



Object Properties: Dynamic Rotation

Use this dialog to specify the following parameters:

- **Tag/Expression** field: Enter a Tag name or expression to associate with the Rotation property. 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.

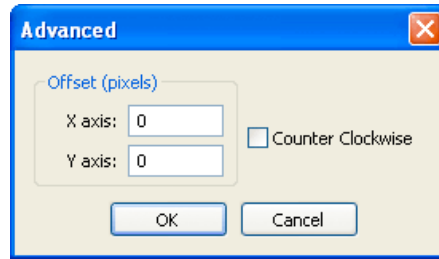
⇒ **Tip:**

For example, a Rotation property 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:

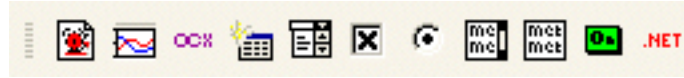


Object Properties: Rotation Property – Advanced Dialog

- **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.


Using the Active Objects Toolbar

The *Active Objects* toolbar provides buttons you can use to create dynamic objects. Active objects typically require more parameters than static objects.



Active Objects Toolbar

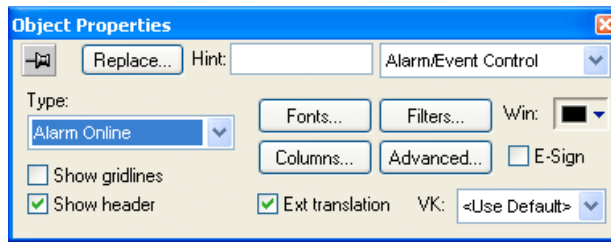
The *Active Objects* toolbar contains the following buttons:

- **Alarm/Event Control Object** button (): Click to add an Alarm/Event Control Object to your application screen.

To create and configure an Alarm/Event Control Object:

- Click the **Alarm/Event Control** button.
- Click in the display, and drag the mouse to create and adjust the object’s shape.

Double-click on the object to open the following *Object Properties* dialog:



Object Properties: Alarm/Event Control

Use the **Alarm/Event Control** *Object Properties* dialog to configure the Alarm/Event Control Object, as follows:

- Select an alarm object mode in the *Type* pane:
 - * **On Line**: Click (*enable*) this button to display current alarm messages.
 - * **History**: Click (*enable*) this button to display alarm messages from the Alarm history database.
- Click (*enable*) the **Show gridlines** check-box to display gridlines in the object:

Active Time	Tagname	Message	Value
08/27/2003 14:38:29	TTTTTTTTTT	MMMMMMMMMMMMMMMMMM...	123
08/27/2003 14:38:29	TTTTTTTTTT	MMMMMMMMMMMMMMMMMM...	123
08/27/2003 14:38:29	TTTTTTTTTT	MMMMMMMMMMMMMMMMMM...	123
08/27/2003 14:38:29	TTTTTTTTTT	MMMMMMMMMMMMMMMMMM...	123

Without Grid

Active Time	Tagname	Message	Value
08/27/2003 14:38:29	TTTTTTTTTT	MMMMMMMMMMMMMMMMMM...	123
08/27/2003 14:38:29	TTTTTTTTTT	MMMMMMMMMMMMMMMMMM...	123
08/27/2003 14:38:29	TTTTTTTTTT	MMMMMMMMMMMMMMMMMM...	123
08/27/2003 14:38:29	TTTTTTTTTT	MMMMMMMMMMMMMMMMMM...	123

With Grid

Displaying a Grid

- Click (*enable*) the **Show Header** check-box to display a header on the object:

Active Time	Tagname	Message	Value
08/27/2003 14:38:29	TTTTTTTTT	MMMMMMMMMMMMMMMMM...	123
08/27/2003 14:38:29	TTTTTTTTT	MMMMMMMMMMMMMMMMM...	123
08/27/2003 14:38:29	TTTTTTTTT	MMMMMMMMMMMMMMMMM...	123
08/27/2003 14:38:29	TTTTTTTTT	MMMMMMMMMMMMMMMMM...	123

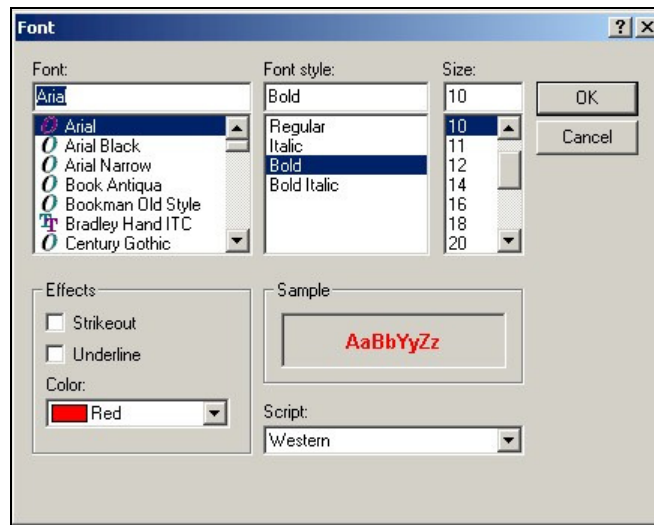
With Header

08/27/2003 14:38:29	TTTTTTTTT	MMMMMMMMMMMMMMMMM...	123
08/27/2003 14:38:29	TTTTTTTTT	MMMMMMMMMMMMMMMMM...	123
08/27/2003 14:38:29	TTTTTTTTT	MMMMMMMMMMMMMMMMM...	123
08/27/2003 14:38:29	TTTTTTTTT	MMMMMMMMMMMMMMMMM...	123

Without Header

Displaying a Header

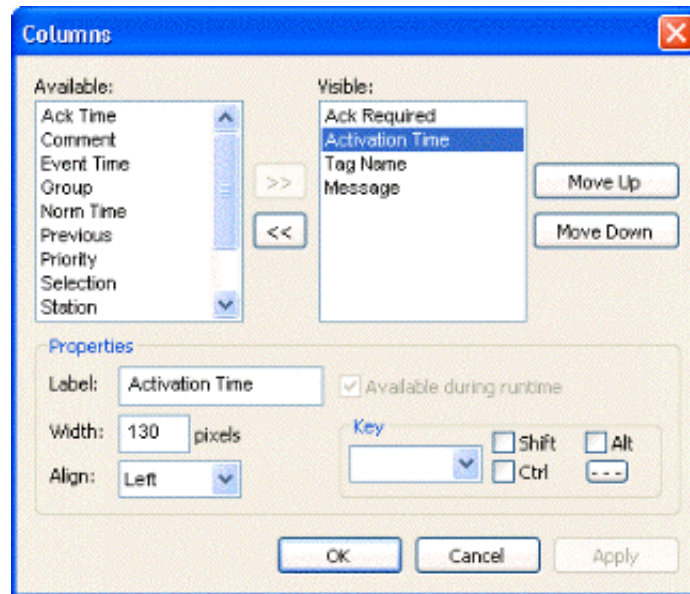
- Click the **Font** button to open the *Font* dialog where you can specify display properties for the alarm text.



Font Dialog

- Use the **Font**, **Font Style**, and **Size** lists to specify a typeface, font style, and size.
- * Click (*enable*) the check-boxes in the *Effects* pane to **Strikeout** (~~strikeout text~~) or **Underline** (underlining) the alarm text. (**Note:** The *Sample* pane shows how the text will look in the object.)
 - * By default, IWS uses Western-style fonts. If you want to change the default, click the **Script** combo-box button and select from the list. (**Note:** The contents of this combo-box list will depend on your operating system.)
 - * When you are finished, click **OK** to close the *Font* dialog.

- Click the **Columns** button to open the *Columns* dialog where you can specify display properties for columns in the Alarm/Event Control Object.



Columns Dialog

- * 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.
- * 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 buttons to move selections between the two lists.

⇒ **Tip:**

Use the *Columns* dialog to display the most recently replaced value with the new value. To do so, move both *Value* and *Previous* from the **Available** list 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** check-box to allow the user to add selected columns to the visible list during runtime.
- * When you are finished, click **OK** to close the *Columns* dialog.

📌 **Note:**

When acknowledging an alarm, the Alarm Control object sends a message to the Alarm Task with the following information: Tag, Alarm to be acknowledged (Hi, HiHi, Lo, etc), User Name and Station. This is a solution to control acknowledged alarms from a Web Thin Client.

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

Filters Dialog

- * Use the **Group** field to filter alarm messages by one or more user group(s). Type the *Group* number into the text field (for example, 1). You also can use a comma or a dash to specify more than one group (for example, 1, 3, 5-6)
- * Use the **Selection** field to filter alarm messages by the **Selection** text configured on the *Alarm* worksheet.
- * Use the **From** and **To** parameters in the *Priority* pane to filter alarm messages based on priority. Type values into the text fields to delimit the priority range.
- * Use the **Tagname**, **Message**, and/or **Username** text fields in the *Search in columns* pane to specify criteria for filtering alarm 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 alarm messages by the last *x* number of messages (**Latest**) or based on a period of time (**Period**).

Notes:

- You can configure tag names (string tags) between curly brackets { } in the **Group**, **Selection**, **Tagname**, **Message**, and **Username** fields to modify the filtering options during runtime.
- You can configure integer tag names for the fields in the *Priority* pane and/or on the latest field from the *Interval* pane to modify these values during runtime.
- You can configure string tag names for the **Period** fields in the *Interval* pane to modify those values during runtime.
- 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 alarm 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.

Notes:

- 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.

Tips:

- 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 column type from the **Column** combo-box, click the **Asc** or **Desc** radio button to sort in ascending or descending order. Click the **Allow sort in runtime** check-box if you want to enable the sort to occur during runtime.
- Click the **Advanced** button to open the *Advanced* dialog where you can specify advanced properties for the Alarm/Event Control Object.

The screenshot shows the 'Advanced' dialog box with the following sections and controls:

- Date & Time Format:** Checkboxes for Day, Month, Year, Hour, Minute, Second, and MS. A 'Sample' text field shows '10/05/2007 08:58:10'.
- Ack:** Security (0), Ack All trigger, Confirm, Ack trigger, and Enable comment (individual ack only) checkboxes.
- Run-time dialog triggers:** Columns and Filters text boxes.
- Delete Message:** Security (0) and Confirm checkbox.
- Print Trigger:** Text box and Multiline checkbox.
- PDF Trigger:** Text box.
- PDF Filename:** Text box.
- Total items:** Text box and Auto Format checkbox.
- Selected tag:** Text box.
- First Row Text:** Text box.
- Summary Changes:** Text box.
- Buttons: OK, Cancel, and Navigation Triggers...

Advanced Dialog

Use the parameters in the *Date and Time Format* pane to control which date and time information displays in the alarm message. Click (*enable*) the check-box to include that element in the display. (**Note:** MS stands for milliseconds.)

Hint: 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 levels can acknowledge an alarm message. Only those users with the specified level can respond.
- * **Ack all tag** 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 tag** 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** check-box: Click (*enable*) this box to display a confirmation dialog when the user tries to acknowledge a single alarm.
- * **Enable comment (individual ack only)** check-box: Click (*enable*) this box to allow the user to enter comments about the alarm, just after acknowledging it.

Use the parameters in the *Run-time dialog triggers* pane to control:

- * **Columns** field: Type a tag to receive a value. When the tag changes value, it opens a dialog allowing you 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 you 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** field: Type a tag in this field to print an alarm summary from your default printer when this tag changes value.
- * **PDF Trigger** field: Type a Tag in this field. When the value of the Tag changes, the data currently filtered in the Alarm/Event Control is distilled to a PDF file and saved to the path specified in the **PDF Filename** field below.
- * **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.

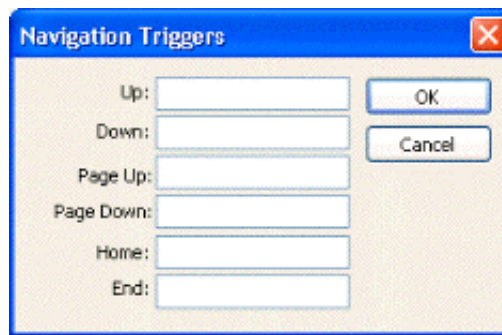
 **Note:**

PDF Trigger and **PDF Filename** are not supported in applications running on Windows CE or Web Thin Client.

- * **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.

- Use the **Win** color box to select a background color for the Alarm/Event Control Object. Click the color box to open the color palette pop-up, and then simply click a color to select it.
- Click (*enable*) the **Ext translation** check-box to enable the external translation of alarm messages using the Translation Tool. (See *Chapter 16: Using the Translation Editor* for more information.)
- **E-Sign** check-box: When this option is checked, the user is prompted to enter the Electronic Signature before executing the dynamic.
- **VK** field: Virtual Keyboard type used for this object. You need to enable the Virtual Keyboard option on the **Project > Settings > Runtime Desktop** interface before configuring the Virtual Keyboard for this interface.
- **Trend Control object** (📊): Click to display 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 *Trend Control Object - Appendix A*, which follows this section, for an example of 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 the runtime
- Built-in legend, which displays the main information associated to each pen linked to the object
- Zooming and auto-scaling tools
- Horizontal and vertical orientation

**Note:**

For compatibility with applications created in older versions of IWS (legacy), the Trend object is still supported and available from the Legacy toolbar. However, since the Trend Control object provides all functionalities of the previous Trend object in addition to other advantages, it is recommended that you use the Trend Control object with new projects.

Trend Control - Development Interface

This section describes the development interface and all the settings available to you as you configure the object on the screen.

Although the Trend Control object supports flexible configurations to meet the specific needs of your application, 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 the runtime), which can be done easily by clicking the Points button from the Object Property window.

Click the Trend Control tool to add it to your application screen. Double-click on the object to launch its Object Properties dialog window, as follows:

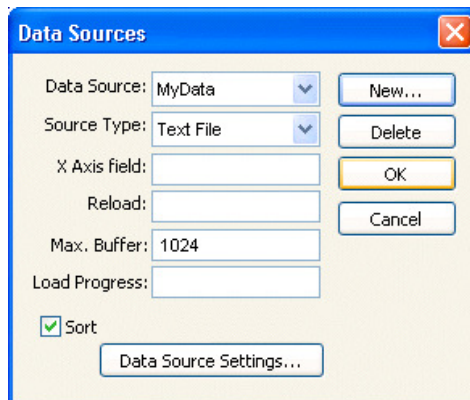


Trend Control Object Properties Dialog

- * **Border** box: Specify a border line Type (style) by clicking on None, Solid, Dashed, Etched, Raised or Sunken. You can also select the color of the border line with the color box to the right of the Type field.
- * **Fill** box: If you click Fill, you can choose a background color for the Trend Control object by selecting it from the color box to the right of this radio button. If you select No Fill, the background of the Trend Control object will remain transparent.

The rest of the buttons on this dialog launch other dialogs for configuring specific settings for the Trend Control object:

- * The **Data Sources** button on the Trend Control Object Properties dialog launches this dialog:



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.

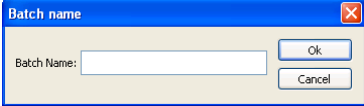
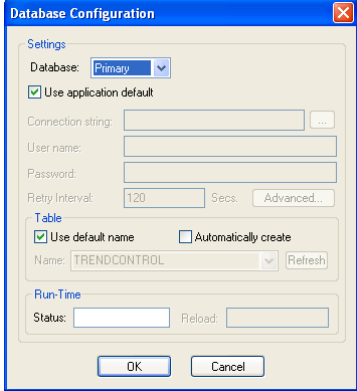
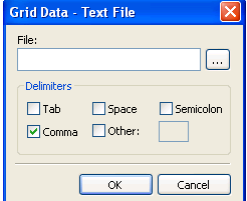
The data source tag is available by default to the Trend Control object. You can add additional Data Sources with the New button. The name you enter will be used as an alias to link the data points to this new data source.

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** field: 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.

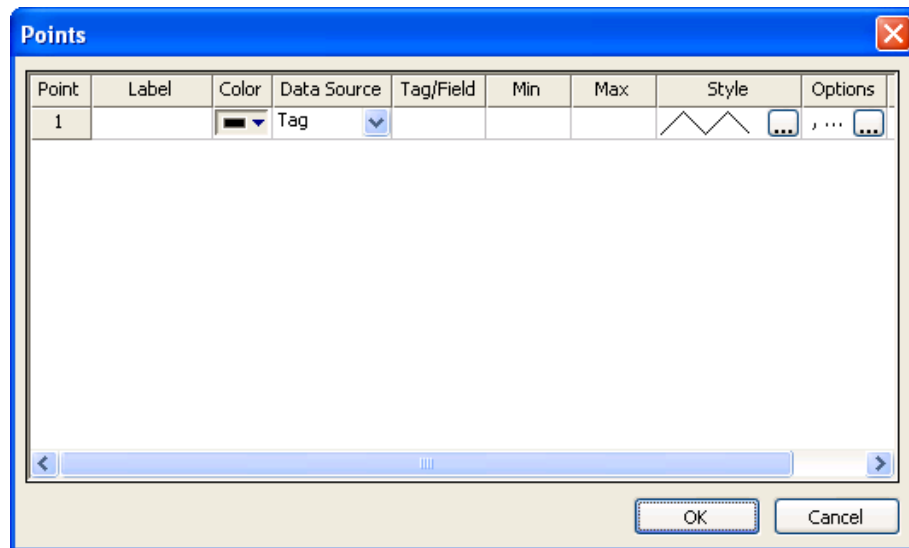
- **Load Progress:** The tag in this field will receive a real value (0-100) that represents the percentage of the Data Source load progress.
- **Sort:** This option is useful for plotting data from a Text file. When enabled (checked), it sorts the data and shows the Cursor column value until the **Max Buffer** is filled. When disabled (unchecked), the data are not sorted and the Cursor column value is not shown.
- **Data Source Settings:** Click to define the settings for the selected Source Type.

The following table summarizes the settings for each Data 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' Batch Name for their retrieval. You can configure a tag between curly brackets in this field to change this setting dynamically during the 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 <i>Database Configuration Dialog Window</i> in Chapter 17 or further information about this dialog interface. Please refer to <i>Trend Control Object - Appendix B</i>, which follows this section, 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 forth.	 <p>Enter the name of the text file that holds the data points. The default path is the current application path. You can configure a tag between curly brackets in this field to change this setting dynamically during the runtime.</p> <p>You can also choose one or more delimiters for the data</p>

			<p>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 <i>Trend Control Object - Appendix A</i>, which follows this section, for an example of configuring text files.</p>
--	--	--	--

* The **Points** button on the Trend Control Object Properties dialog launches this dialog:



Points Dialog

The value of each data Point can be represented in the Trend Control object as a pen, during the runtime. You can select which data Points will be visible during the runtime (add/remove pens to the chart), regardless of how many data Points you associate with the Trend Control object.

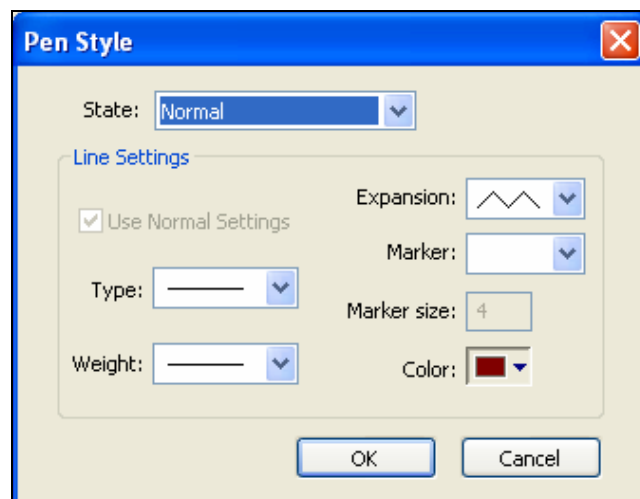
The following table summarizes the properties of each Data Point:

Property	Description
Point	Data Point ID. Each data Point has a unique ID, which is assigned automatically when the data Point is created in this interface.
Label	The label associated with the Data Point can be displayed on the Legend, during the runtime, providing a short reference to the user for each data Point.
Color	Color of the pen used to draw the values of the Data Point on the Trend Control object
Data Source	Data Source that holds the values for the data Point. The Data Source Tag is available by default. See the Data Sources button above for further information about how you can make additional Data Sources available for the object.

Property	Description
Tag/Field	<p>The meaning of this parameter depends on the Data Source Type associated with the data Point:</p> <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 data Point values. The number 0 refers to the first column, 1 refers to the second column, and so on.
Min	Minimum value displayed in the Y scale for the data Point
Max	Maximum value displayed in the Y scale for the data Point
Style	Configure the style for the pen (color, type, state, etc.). See the Pen Style dialog below for further information.
Options	Configure optional settings for each data Point. You can use the dialog to configure these settings or type their values directly in the Options field, using the comma character as the delimiter. See Options dialog above for further information about these settings.
Hide	You can configure a tag in this field. When this tag has the value 0, the pen associated with the point is displayed on the object; otherwise, it is hidden.



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

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).



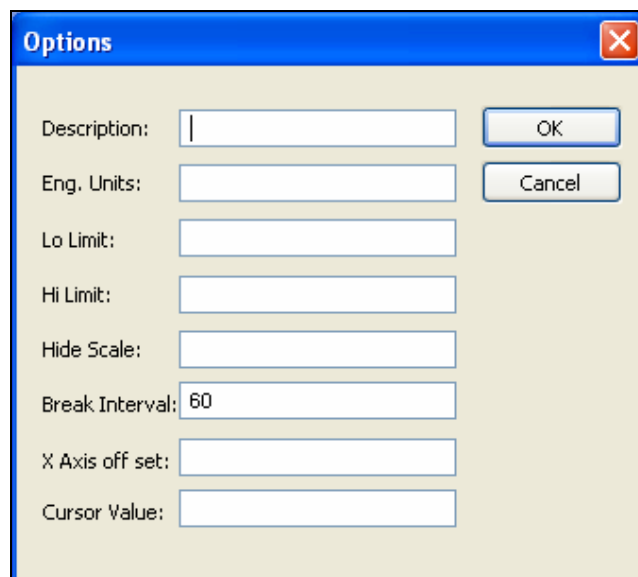
Pen Style Dialog

After you select a State (Normal or Out of Limits), you can configure its pen style:

Property	Description
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	Select the type of the line used to draw the pen.
Weight	Select the weight (thickness) of the line used to draw the pen.
Expansion	Select the algorithm used to link the points, as follows: <ul style="list-style-type: none">  : The consecutive points are interpolated directly to each other using a line. This option is suitable for analog values.  : The consecutive points are linked through vertical and horizontal lines only (steps). This option is suitable for Boolean values.
Marker	Select the type of marker (if any) which must be displayed on each specific sample retrieved from the Data Source and displayed on the object.
Color	Select the color of the Marker (if any) and line used to draw the pen on the object.
Marker Size	Select the size of the Marker (if any).

Note:
 When running the application under the WinCE operating system or on the Web Thin Client (any OS), the Pen Style dialog box – available during the runtime – allows the user to change the pen color only.

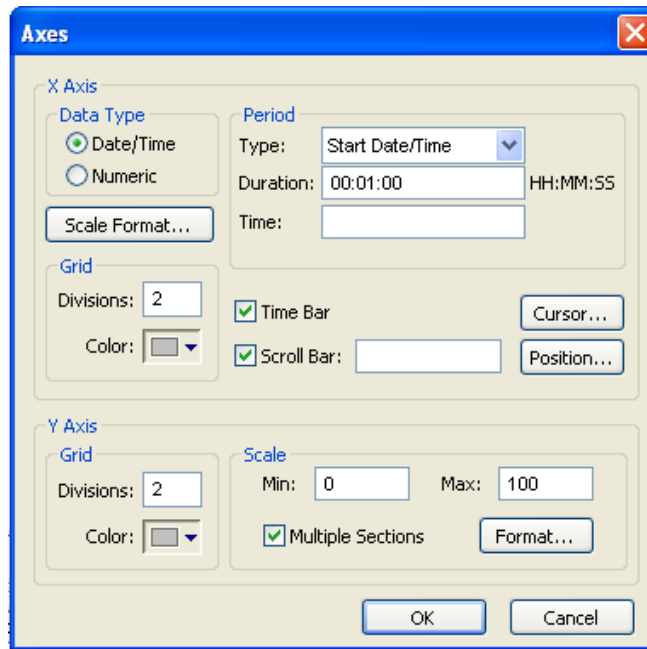
* **Options** dialog: Use this dialog to configure optional settings for each data Point, as follows:



Options Dialog

Property	Description
Description	This text can be displayed in the legend, providing a short description about the data point, during the runtime. When using tags, the default description is the one configured for the tag.
Eng. Unit	This text can be displayed in the legend, providing the Engineering Unit associated to the data point, during the runtime. When using tags, 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 the runtime. See Pen Style dialog above for further information. When using tags, 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 the runtime. See Pen Style dialog above for further information. When using tags, 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 the runtime by changing the value of this tag (0=Show ; 1=Hide).
Break Interval	<p>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.</p> <p>This field has some special values:</p> <ul style="list-style-type: none"> ▪ -1 : Do not connect the points. ▪ -2 : Connect only points in ascending order.
X Axes off-set	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	You can configure a tag in this field. During the runtime, the Trend Cursor object updates the value of this tag with the value of the intersection between the data point pen and the Vertical cursor (if any).

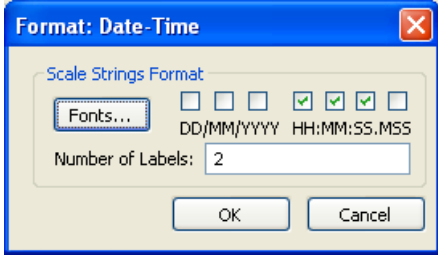
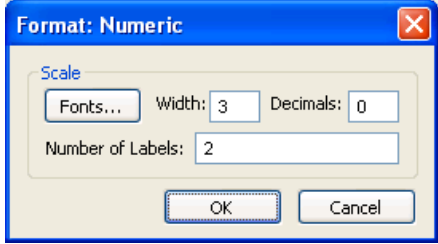
- * **Axes** button: The Axes button on the Trend Control Object Properties dialog launches this dialog:



Axes Dialog

This dialog allows you to configure the settings for the X and Y axis.

- **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 the runtime.

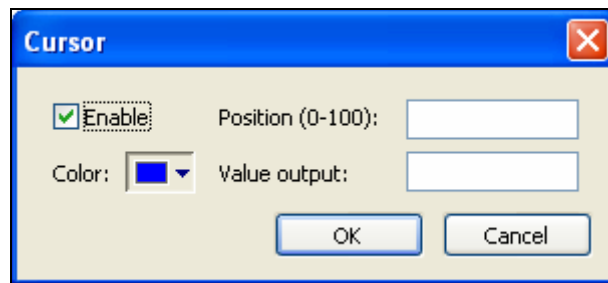
- **Period or Range:** The settings available in this interface depend on the Data Type configured for the X axis, as follows:

Data Type	Property	Description
Date/Time (Period)	Type	<ul style="list-style-type: none"> ▪ 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. ▪ Auto: When this option is selected, the Trend Control object works with Start Date/Time when it is triggered to Pause Mode, and it works with Time Before Now when it is triggered to Play Mode.
	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 the runtime by changing the value of this tag. The format of the value supported by this property is HH:MM:SS. E.g. 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. E.g.: 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: <ul style="list-style-type: none"> a. Time (string value). E.g. 48:00:00 (forty eight hours) b. Number of hours (real value). E.g. 2.5 (two hours and thirty minutes)
Numeric	Min	Minimum value displayed in the X axis
	Max	Maximum value displayed in the X axis

Note:

- The tags configured in the Period / Range fields are automatically updated when the user changes the X scale dynamically during the runtime, using the Time bar embedded in the object.
- 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.

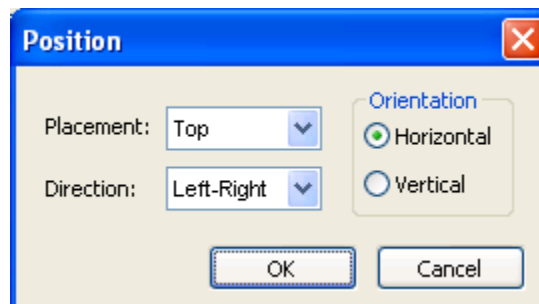
- **Grid** (X axis or Y axis): You can configure the number of divisions (vertical or horizontal lines) drawn on the object for the X and/or Y axis respectively, as well as the color of these lines.
- **Time bar**: When checked, the Time bar is displayed below the X axis during the runtime; otherwise, it is hidden. The time bar is a standard interface that can be used by the operator to change the X axis scale during the runtime.
- **Scroll bar**: When checked, the Scroll bar is displayed below the X axis during the 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 the 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.
- **Cursor**: The cursor is an optional ruler orthogonal to the X axis, which can be used during the 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 the 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:



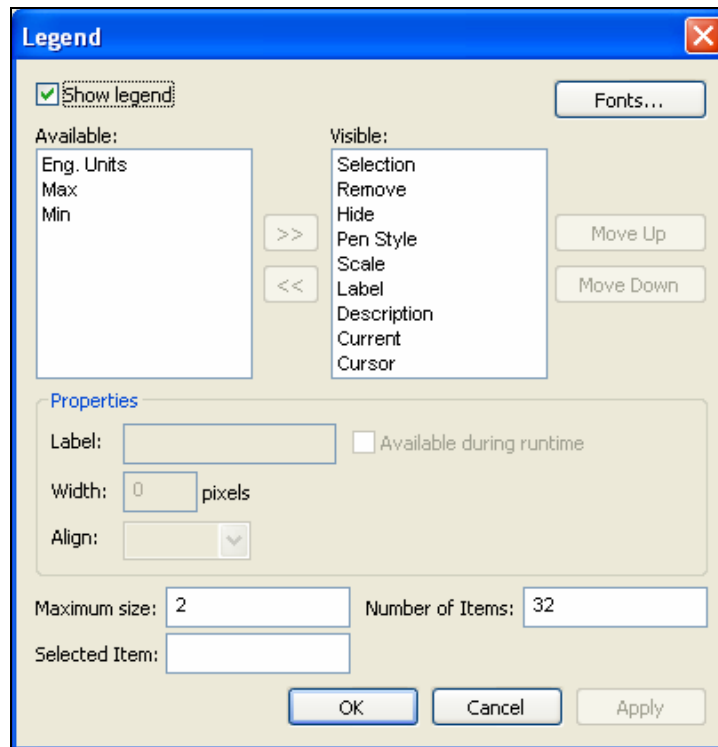
Position Dialog

Property	Description
Placement	Position where the X axis will be placed
Direction	Direction of the X axis
Orientation	Orientation of the X axis

- **Scale:** Defines the properties of the Y axis, as follows:

Property	Description
Min / Max	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).
Multiple Sections	When checked, the Y scale is divided automatically into one section for each pen; otherwise, all pens share the same Y scale.
Format	Launches a dialog for configuring the format of the labels displayed by the Y axis.

- * **Legend button:** The Legend button on the Trend Control Object Properties dialog launches this dialog:

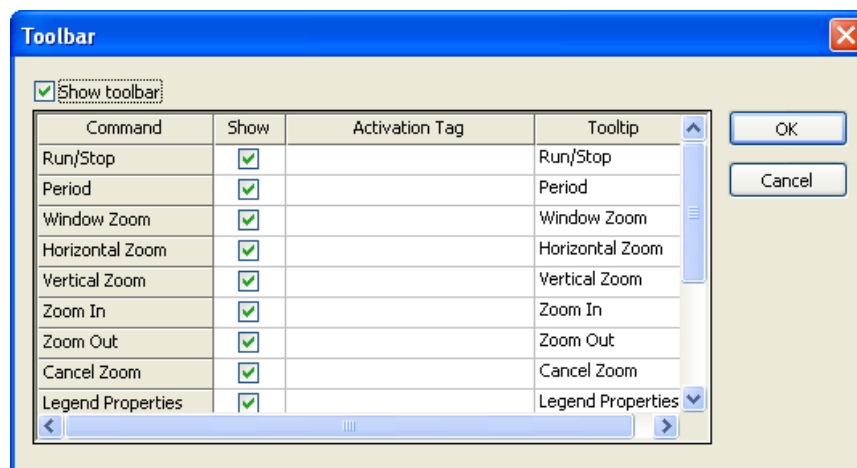


Legend Dialog

- **Show:** When checked, the embedded legend is displayed during the runtime. This interface provides useful information associated with the pens currently linked to the object.
- **Available / Visible:** The fields in the Visible box are displayed in the legend during the runtime. You can add fields 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 fields are displayed in the legend during the runtime.
- **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 the runtime
Width	Width for the field (in pixels) during the 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 the 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 sizes 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 the 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.
- * **Toolbar button:** The Toolbar button on the Trend Control Object Properties dialog launches this dialog:

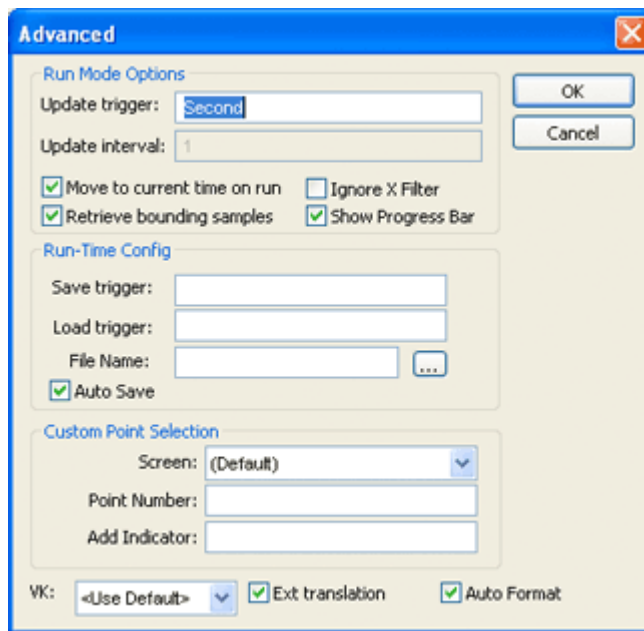


Toolbar Dialog

- **Show toolbar:** When this box is checked, the embedded toolbar is displayed during the runtime. This interface provides useful buttons to trigger actions associated with the object. You can configure the following settings for each Command (button) available for the toolbar:

Property	Description
Show	When this box is checked, the button is displayed on the toolbar embedded on the Trend during the runtime.
Activation Tag	You can configure a tag in this field (optional). When the tag changes value, it triggers the respective command. This option is useful when you want to create customized interfaces to trigger the commands, instead of (or redundant with) the embedded toolbar.
Tooltip	The text configured in this field is displayed as a tooltip during the runtime when the mouse cursor is on the icon of the toolbar.

- * **Advanced button:** The Advanced button on the Trend Control Object Properties dialog launches this dialog:



Advanced Dialog

Run Mode options: The settings on this area define the behavior of the trend when in run mode:

Property	Description
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 is of type numeric, the value on this field will be added to the minimum and maximum values of the X Axis.
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 the runtime.

Property	Description
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 the history. On the other hand, the object will not draw lines linking the first and last samples to the extremities of the 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.
Show Progress Bar	When this box is checked, a progress bar will appear during the trend load.

Run-Time Config: The settings of the Trend object modified during the runtime can be saved in temporary files. This option can be used to:

- Keep the settings consistent, so when the user closes the screen and opens it again, or re-starts the application, the settings configured during the runtime are not lost.
- Create standard settings for different scenarios and load the appropriate configuration during the run-time, based on a pre-defined condition or based on the user-selection.

The properties of this frame are described in the following table:

Property	Description
Save trigger	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 Web Thin Client.
Load trigger	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 the runtime.
File Name	<p>If this field is left blank, the temporary file is saved in the application \Web sub-directory with the syntax <ScreenName><ObjectID>TrendControl.stmp (e.g. MyScreen10TrendControl.stmp). The Web Thin Client station saves/loads the temporary file in the standard Temp directory of the operating system (e.g. \Documents and Settings\<CurrentUser>\Local Settings\Temp).</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 the runtime by changing the value of this tag. If you do not specify any path, the file is saved in the application \Web sub-directory by default.</p>
Auto Save	When this box is checked, the current settings of the Trend are automatically saved in the temporary file when the screen where the Trend is configured is closed during the runtime. If the box is not checked, the settings are saved only when the Save trigger command is executed.

⚠ Caution:

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.

Custom point selection: This interface allows you to create your custom dialog to modify or insert pens to the object, as follows:

Property	Description
Screen	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 the runtime.
Point number	Point Number (from the Points Dialog), indicating the point associated to the pen that will be inserted or modified during the runtime.
Add Indicator	Flag that indicates that the user triggered an action to insert a new pen (value 1) instead of modifying a pen that is already been visualized (value 0).

VK: Virtual Keyboard type used for this object.

Ext 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 **SetDecimalPoints ()** function.















📌 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.

Trend Control – Runtime Interface






When enabled, some embedded interfaces can help the user to interact with the Trend Control during the runtime. This section describes these interfaces:

* **Toolbar:** The commands available in the embedded Toolbar are described in the following table:

Command	Icon	Description	Activation Tag
Run		Set the Trend to the Play Mode. In this mode, the X axis is continuously updated (Online Mode). This option is disabled (grayed out) when the trend is already in Play Mode.	0 = Play Mode on 1 = Play Mode off
Stop		Set the Trend to the Stop Mode. In this mode, the X axis is not continuously updated (History Mode), so the user can visualize history data in a frozen period of time. This option is disabled (grayed out) when the trend is already in Stop Mode.	0 = Stop Mode on 1 = Stop Mode off
Period		Launches an embedded dialog, where the user can modify the main settings of the X axis scale	When the Activation Tag changes value (e.g. toggles), this command is executed.
Window Zoom		Allows the user to click on the Trend area and drag the cursor to select the area that must be visible when the cursor is released. This option is disabled (grayed out) when the Multiple Section option (for the Y scale) is active.	
Horizontal Zoom		Allows the user to click on two points on the Trend area, defining the Horizontal scale that must be available	
Vertical Zoom		Allows the user to click on two points on the Trend area, defining the Vertical scale that must be available. This option is disabled (grayed out) 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 area.	0 = Zoom In on 1 = Zoom In off
Zoom Out		Allows the user to zoom out each time they click on the Trend area.	0 = Zoom Out on 1 = Zoom Out off
Cancel Zoom		Cancels the current Window , Horizontal or Vertical Zoom and returns the Trend display to its original scale.	When the Activation Tag changes value (e.g. toggles), this command is executed.
Legend Properties		Launches an embedded dialog, where the user can modify the Legend main settings	
Pen Style		Launches an embedded dialog, where the user can modify the style of the selected pen.	
Add Pen		Launches a dialog, where the user can add a new pen to the Trend object	
Remove Pen		Removes the selected pen from the Trend object	
Multiple Sections		Switches the Y scale to Multiple Sections (a section for each pen) or Single Section (all pens share the same Y scale section).	0 = Multiple Sections on 1 = Multiple Sections off

Cursor		Turns the cursor (ruler) to visible or hidden	0 = Cursor on 1 = Cursor off
Auto Scale		Changes the Y axis scale to fit all values from the pens 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 display. (Historical data are not printed.)	When the Activation Tag changes value (e.g. toggles), this command is executed.

- * **Legend:** The commands available in the embedded Legend are described in the following table:

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

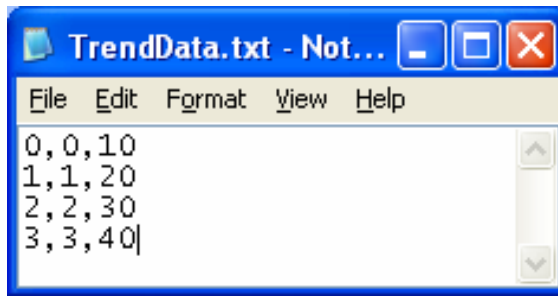
- * **Scroll bar:** Using the Scroll bar, the user can slide through the X axis values, according to the period configured for this scale.
- * **Time bar:** Using the Time bar, the user can modify the Duration, as well as the Start Date/Time and/or the End Date/Time, for the data displayed on the object. Changing these values will affect the tags associated with the X axis scale (if any).

Trend Control Object – Appendix A – 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:

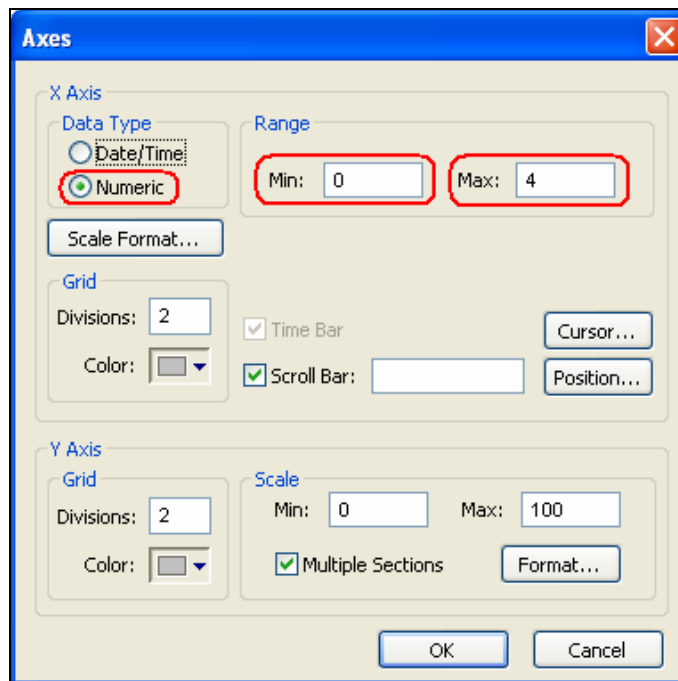
X Value	Y1 Value	Y2 Value
0	0	10
1	1	20
2	2	30
3	3	40

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:

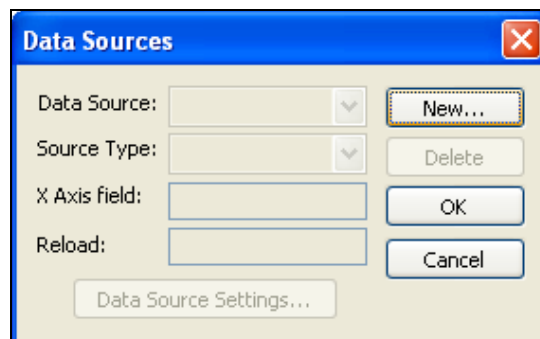


We strongly recommend that you save the file in the same folder where the application is. By doing so, you do not have to specify the entire path and your application will still work, even if it is copied to a different computer.

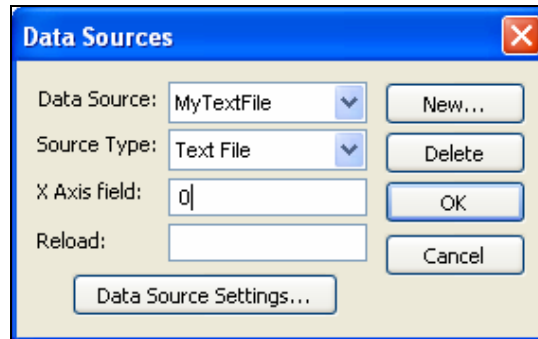
Once you have added the Trend Control to your screen, double click on the object to open then Object Properties and click on Axis. Change the Data Type of the X Axis to numeric, and set the ranges as shown in the picture below:



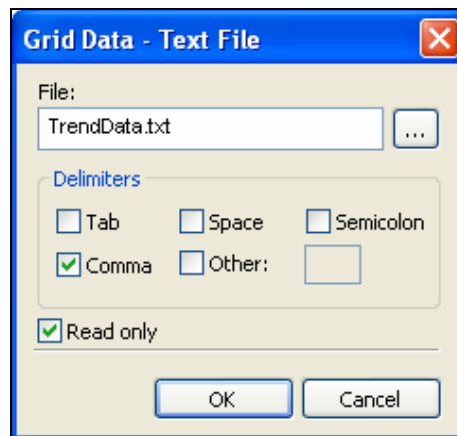
Click Ok on this Window and then, in the Object Properties window, click on the Data Sources button. The following window will display:



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:

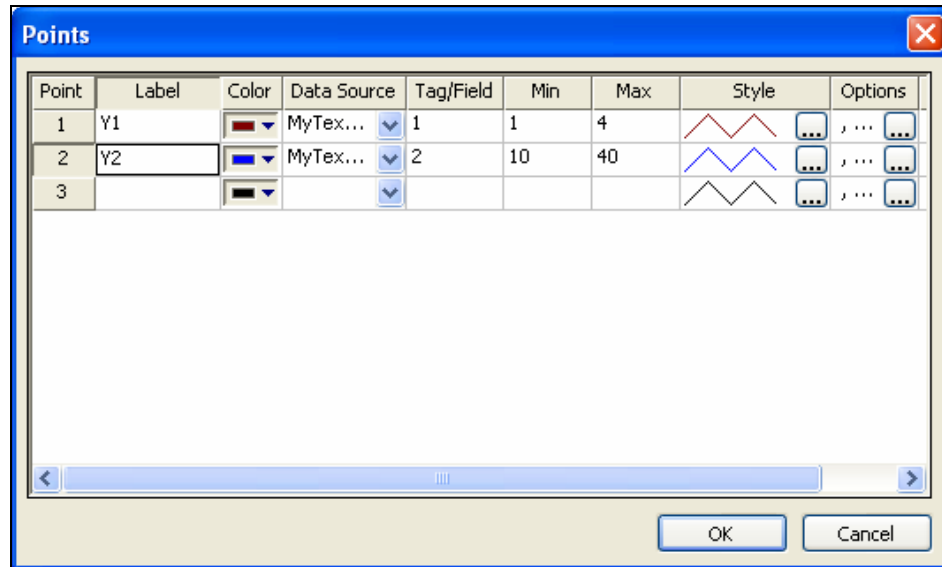


On the X Axis field we need to indicate which column in our text file represents the X Axis. In our example we are using column zero, so enter with zero for this field, then click on the button Data Source Settings, the following Window will display:

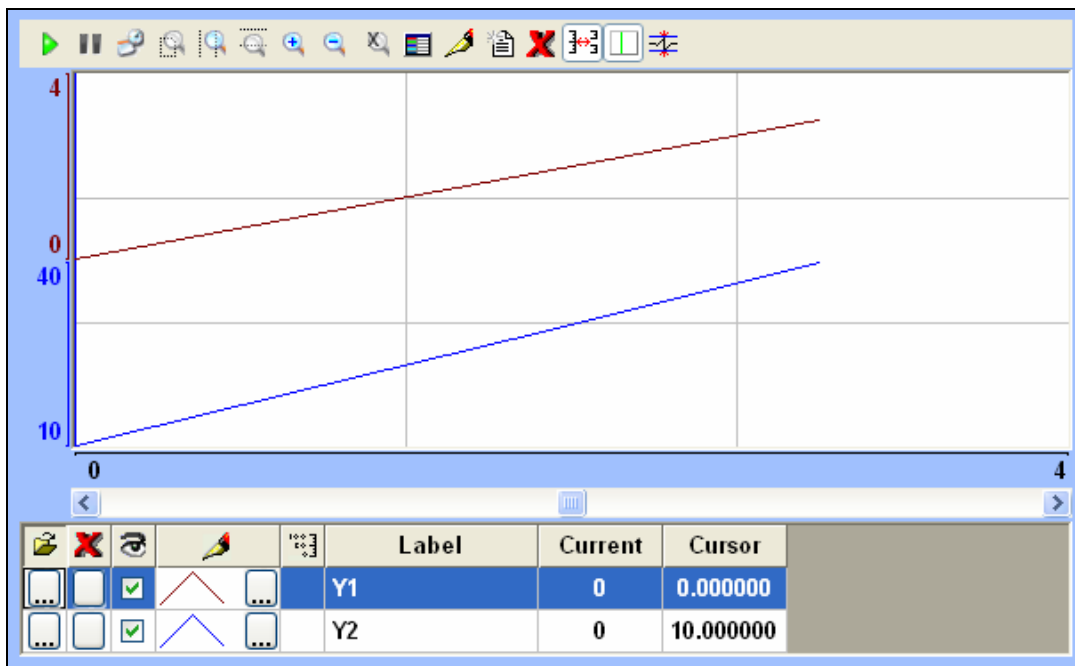


If you have copied the text file to the application folder, you only have to specify the file name, otherwise, enter with the complete path where the file is located (use the browse button as needed). Click Ok on this window and Ok again to finish the data source configuration and close the Data Source 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 application and you should see something similar to the figure below:



Trend Control Object – Appendix B – 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:

The 'Data Sources' dialog box is shown with the following fields and buttons:

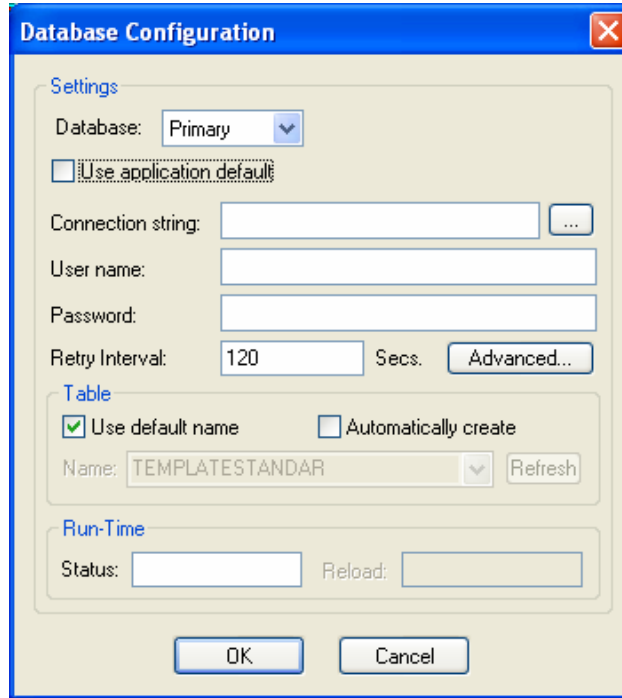
- Data Source: (empty dropdown)
- Source Type: (empty dropdown)
- X Axis field: (empty text box)
- Reload: (empty text box)
- Buttons: New..., Delete, OK, Cancel, Data Source Settings...

We need to create a data source in order to access to the database. Click on the new button, specify the Data Source Name “MyDB” and then click Create. You should see the following information now:

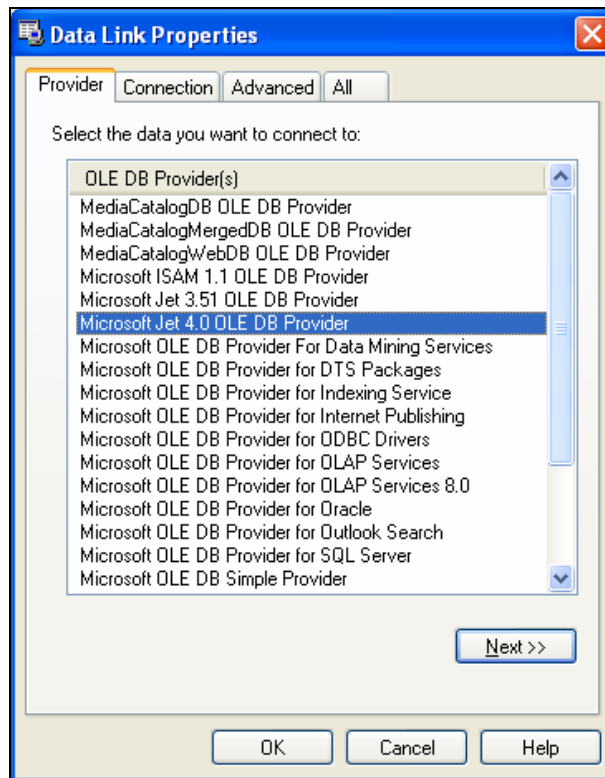
The 'Data Sources' dialog box is shown with the following fields and buttons:

- Data Source: MyDB
- Source Type: Database
- X Axis field: Time_Stamp
- Reload: (empty text box)
- Buttons: New..., Delete, OK, Cancel, Data Source Settings...

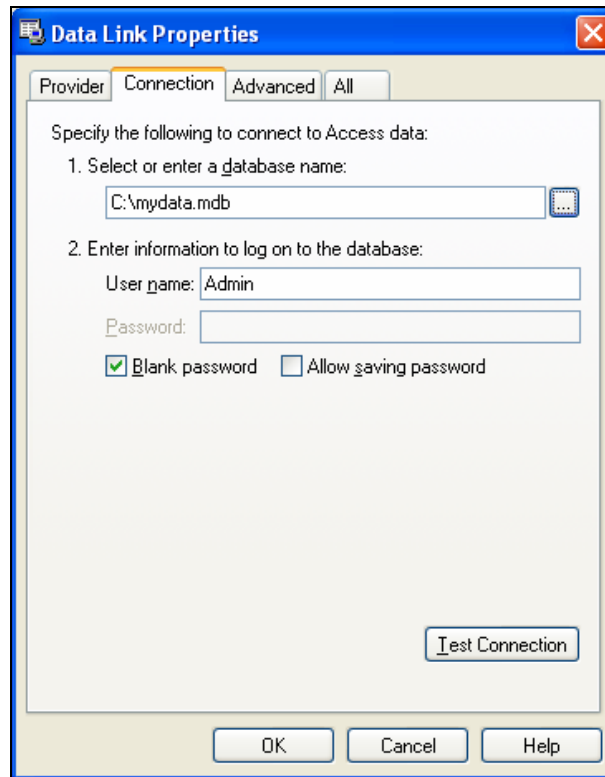
Change the Source Type to Database and specify Time_Stamp in the X Axis field. Then click on the Data Source Settings button, the following window will display:



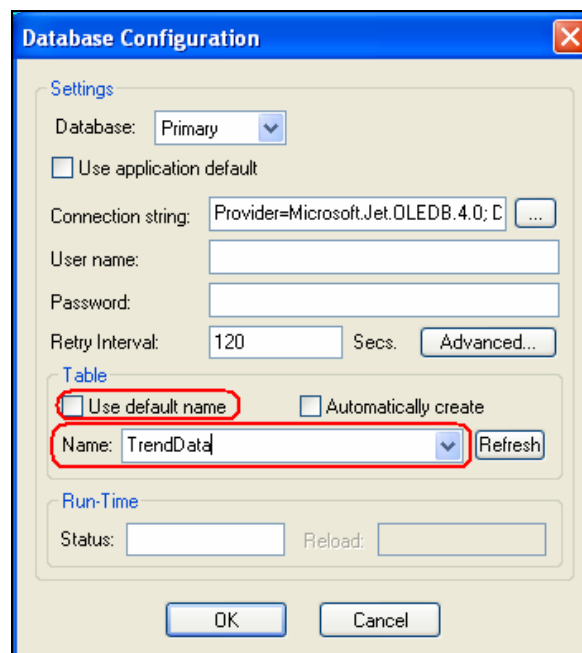
Uncheck the check box Use application default and click on the browse button in order to configure the connection string. The following window will display:



Select the Microsoft Jet 4.0 OLE DB Provider and click Next. In the following window, you should specify the database path:

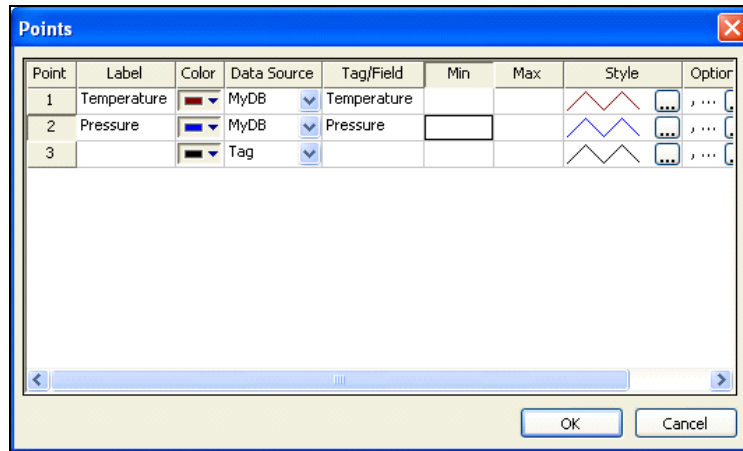


Click Ok to finish the Connection String configuration. Now uncheck the option Use default name and select the table from your database as shown below:

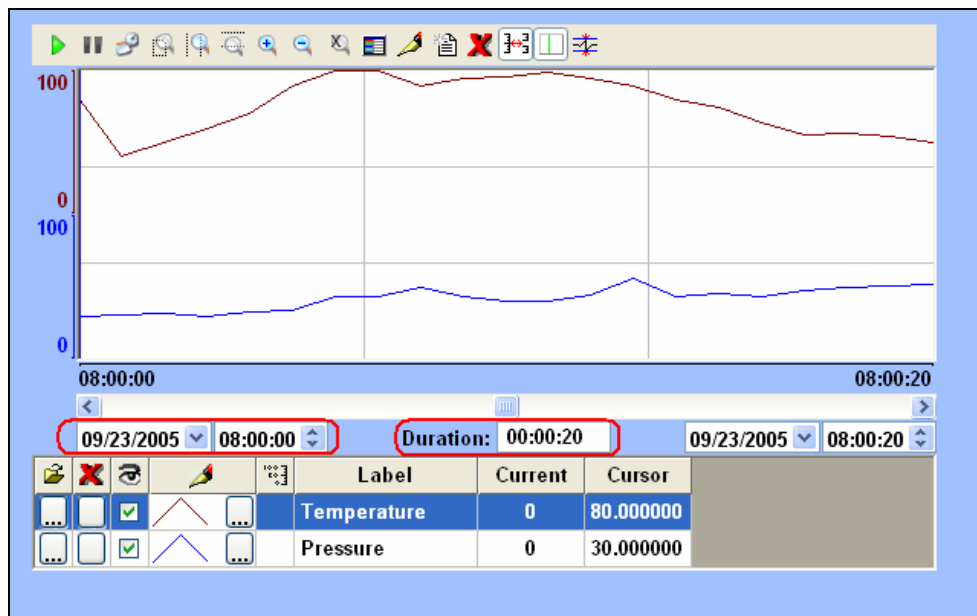



Click Ok on this window and Ok again to finish the data source configuration and close the Data Source configuration window.

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 on 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:



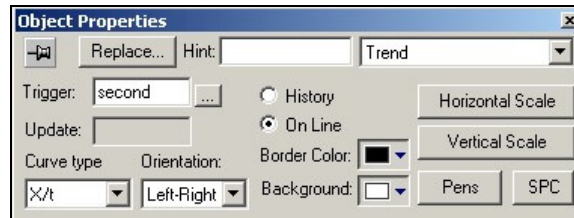
- **Trend** button (): Click to select an area on the screen in which to exhibit trend curves of tag data. You can display up to eight curves simultaneously in the object, and use configuration fields in the object properties to specify:
 - Time period to exhibit
 - Values to exhibit

- Format of the graphic

To create a trend object:

- Click the **Trend** button.
- Click in the display, and drag the mouse to create and adjust the object's shape.

Double-click on the trend object to open the *Object Properties* dialog:



Object Properties: Trend

Use the **Trend Object Properties** dialog to specify the following parameters:

- **Trigger** field: Type (or click the button to select) a variable to define how trend curves are redrawn. When the value of this tag changes, IWS redraws the displayed trend curves. You *must* complete this field if you specify (*enable*) **On Line** trends, but it is not required for **History** trends.
- **Update** field: Type a variable for IWS to use in creating the trend. Use this field in conjunction with **Crisp** trends *only*.
- **Curve Type** combo-box: Select a curve type from the following options:
 - * **X/t**: Select to plot curve values according to time
 - * **X-Y**: Select to plot curve values according to **Tag X**, another tag in the application
 - * **Crisp**: Select to plot curve values in a format appropriate to interface with VAX stations
- **Orientation** combo-box: Select one of the following to specify how the trend pens are oriented in relation to the screen:
 - * **Left-Right**: Select to plot from left to right, with latest values on the right
 - * **Right-Left**: Select to plot from right to left, with latest values on the left
- **History** radio button: Click (*enable*) to display the application's historical trend curves.

⚠ Caution:

- To use history-type graphics for this trend object, you must:
 - Create a trend group (from the *Trend* folder on the **Tasks** tab)
 - Create tags for this group and set the **Save on Tag Change** or **Save on Trigger** parameters on the *Trend Group* worksheet. These tags will have their samples stored on the hard disk.

- **On Line** radio button: Click (*enable*) to display on-line trend curves for the application.

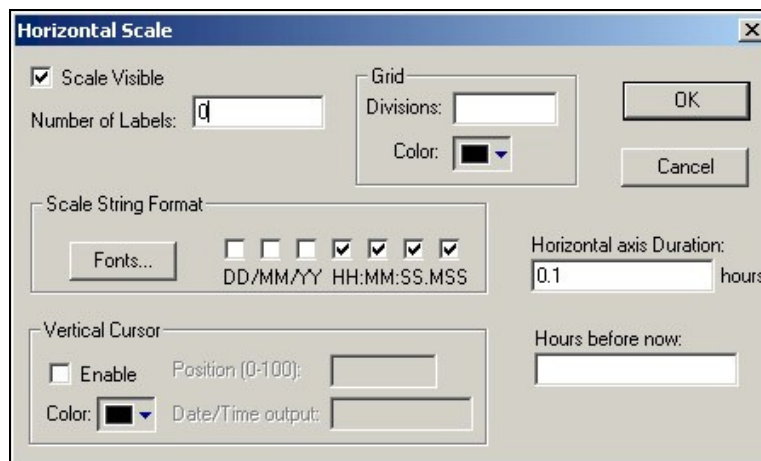
📌 Note:

If you enable this button, you must enter a value in the **Trigger** text box to

indicate when IWS will redraw the trend curves.

- **Border Color** combo-box: Click to select a color for boarder of the trend graphic area. When the *Color* dialog displays, click on a color to select it, and then close the dialog.
- **Background Color** combo-box: Click to select a background color for the trend graphic area. When *the Color* dialog displays, click on a color to select it, and close the dialog.
- **Horizontal Scale** button: Click to open the *Horizontal Scale* dialog, which allows you to define horizontal scale properties for the trend curves.

Note:
The content on this dialog changes, depending on the parameters you specify on the *Object Properties Trend* dialog.



Sample Horizontal Scale Dialog

The following section explains how to set all of the **Horizontal Scale** parameters.

- * **Scale Visible** check-box: Click (*check*) to display the horizontal scale or uncheck the box to hide the horizontal scale.
- * **Number of Labels** field: Type a value to specify how many labels you can use in the horizontal scale.
- * **Grid** area: Use the following parameters to configure a grid for the trend graphic.
 - Divisions** field: Specify how many division lines to use for the grid. If you leave this field blank, no grid lines display.
 - Color** combo-box: Click to select a color for the grid. When the *Color* dialog displays, click on a color to select it, and then close the dialog.
- * **Scale String Format** area: Use the following parameters to specify a format for the text string on horizontal scale.
 - Fonts** button: Click this button to open the *Fonts* dialog, which allows you to select a style, size, color, and font type for the horizontal axis labels.
 - DD/MM/YY** check-boxes (*Curve X/Y and Graph X/t only*): Click (*check*) these boxes to add a date (days/months/year) to the text string in the scale.

HH:MM:SS check-boxes (*Curve X/Y and Graph X/t only*): Click (*check*) these boxes to add the time (hours:minutes:seconds) to the text string in the scale.

- * **Vertical Cursor** area: Use the following parameters to specify a format for the vertical cursor.

Enable check-box: Click (*enable*) to display a vertical cursor.

Color combo-box: Click to select a color for the vertical cursor. When the *Color* dialog displays, click on a color to select it, and close the dialog.

Position (0-100) field (*becomes active when you enable the vertical cursor*): You must enter a real tag value, which IWS updates using the cursor position (0 corresponding to the left edge and 100 the right edge of the trend).

Date/Time output field (*becomes active when you enable the vertical cursor*):

Enter a tag name to receive a string. This string is constantly updated with the current time of the vertical cursor in the trend.

- * **Horizontal axis Duration** field (*Curve type X/t only*) or **Duration (hours)** (*History and Graph X/t only*): Type a tag name or numeric value for width of the trend. For example: If you specify **Horizontal axis Duration** = 0.03333 (2 minutes) the trend will display 2 minutes of data from beginning to end.

- * **Hours before now** field: Specify a tag to perform scrolling in the trend area. The value of this tag specifies the start time of the trend in relation to the current time.

For example: If you specify the following, IWS allows you to display the trend graphic for up to five hours before the current time.

Time = 17:00:00

Hours before now = 5

Duration = 1

Trend will show 11:00.00 > 12:00.00

 **Caution:**

The *Trend Graphic* dialog holds a maximum of 32,000 samples.

 **Note:**

When you use the **Hours Before Now** parameter, you are handling historical data. Consequently, you must configure the trend pens into one trend group (using the *Trend* folder on the **Tasks** tab).

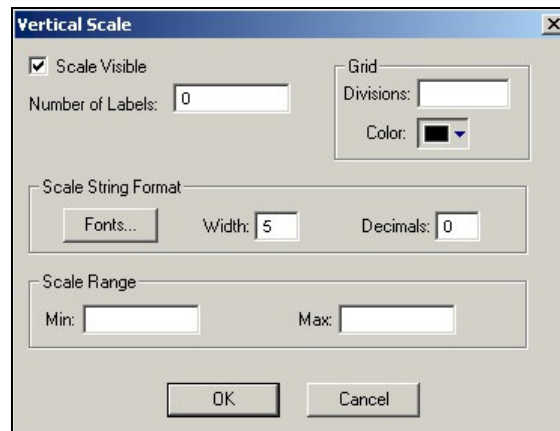
- **Type of History Files** area (*History and Graph X/t only*): Use the following parameters to specify in which format the history files were created.
 - * **Date** (*default*): Click to handle files created in date format by trend group (InduSoft format).
 - * **Batch**: Click to handle files created in batch format by trend group (batch file format).
 - * **Start Date** field (*History and Graph X/t only*): Specify a date (typically a string-type tag) in DD/MM/YYYY format to specify when to start the history curves.
 - * **Start Time** field (*History and Graph X/t only*): Specify a time (typically a string-type tag) in HH:MM:SS format to start the history curves.

- * **Tag X** field (*Graph X/Y only*): Specify a tag for the X-axis. You must specify an array and declare the index in which the axis starts (for example, MyTagX[1]).
- * **Points** field (*Graph X/Y only*): Specify the number of points (samples) in the graph dialog.
- * **Max** field (*Graph X/Y only*): Specify a maximum value for the X variable.
- * **Min** field (*Graph X/Y only*): Specify a minimum value for the X variable.

⇒ **Tip:**

You can use **Recipe** module (from the **Tasks** tab) to save and load historical information for the X-Y trend. See *Chapter 8: Configuring Task Worksheets*.

- **Vertical Scale** button: Click to open the *Vertical Scale* dialog, which allows you to define vertical scale properties for the trend curves.

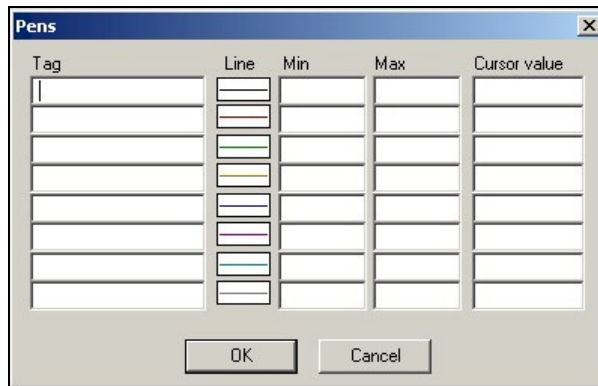


Sample Vertical Scale Dialog

- **Scale Visible** check-box: Click (*check*) to display the vertical scale or uncheck the box to hide the vertical scale.
- **Number of Labels** field: Type a value to specify how many labels you can use in the vertical scale.
- **Grid** area: Use the following parameters to configure a vertical grid for the trend graphic.
 - * **Divisions** field: Specify how many division lines to use for the vertical grid. If you leave this field blank, no grid lines display.
 - * **Color** drop-down list: Click to select a color for the vertical grid. When the *Color* dialog displays, click on a color to select it, and then close the dialog.
- **Scale String Format** area: Use the following parameters to specify a format for the text string on vertical scale.
 - * **Fonts** button: Click this button to open the *Fonts* dialog, which allows you to select a style, size, color, and font type for the vertical axis labels.
 - * **Width** field: Specify how many digits are permitted for numbers in the vertical scale string.
 - * **Decimals** field: Specify how many decimal places are permitted in the vertical scale string.

For example: If you enter **Width** = 3 and **Decimals** = 2, the horizontal scale string can use three-digit numbers with two decimal places.

- **Scale Range** area: Use the following parameters to calculate how the curve positions are defined in the *Pens* dialog. You can use a numeric value or a tag to dynamically change the vertical scale.
 - * **Minimum** field: Type a minimum value for the trend graphic scale.
 - * **Maximum** field: Type a maximum value for the trend graphic scale.
- **Pens** button: Click to open the *Pens* dialog:



Pens Dialog

Use the following parameters to configure the pens used to draw each tag curve.

- **Tag** fields: Type the tag names to be monitored in the trend (for example, MyTagY[1]). You can create a generic *Trend* dialog by specifying an indirect tag to define the tag to monitor.
- **Line** buttons: Click to select a color for the lines in each trend curve. When the *Color* dialog displays, click a color to select it, and close the dialog. You can use up to eight different colored lines on the trend graph.
- **Min** and **Max** fields: Specify a minimum and a maximum value of scale (numeric or tag) with which to draw the curve.



Note:

The minimum/maximum values of each tag are not required to have the same minimum/maximum values as the trend graphic scale.

- **Cursor Value** fields: Type tags to receive the value of the *Trend* line where it intersects with the vertical cursor.



Caution:

The *On-Line* trend allocates memory for each point inside the period of visualization. CEView will discharge the older values (even if they are inside the period of visualization) when the memory available is lower than a critical limit (1MB by default). In addition, a warning message is launched locally.

- **SPC** button: Click to open the *SPC Parameters* dialog, which you can use to return **Mean**, **Min**, **Max**, and **Standard Deviation** (-2s, +2s) values from a selected period for each pen. (You can also draw the results on the *Trend* object.)

	Tag	Draw	Line
Mean:	<input type="text"/>	<input type="checkbox"/>	<input type="button"/>
Min:	<input type="text"/>	<input type="checkbox"/>	<input type="button"/>
Max:	<input type="text"/>	<input type="checkbox"/>	<input type="button"/>
Median:	<input type="text"/>	<input type="checkbox"/>	<input type="button"/>
Sum:	<input type="text"/>	<input type="checkbox"/>	<input type="button"/>

Standard Deviation

Tag: +2s -2s

OK Cancel


SPC Parameters Dialog

Use the parameters on this dialog as follows:

- **Pen** combo-box: Click the arrow button to select a pen from the list.
- **Disable**: Type a tag into the text field. If the tag value is other than 0 (false), IWS disables the **SPC** feature.
- **Tag** fields: Type a tag name into the text fields to return values for **Mean**, **Min**, **Max**, **Median**, **Sum**, and/or **Standard Deviation**.
- **Draw** check-box: Click (*enable*) these boxes to draw the results of the **Mean**, **Min**, **Max**, **Median**, **Sum**, and/or **Standard Deviation -2s/+2s** values on the trend graph.
- **Line** button: After enabling the **Draw** check-box, click this button to set display parameters for the different pen line(s).

When the *Line Selection* dialog displays, specify the following:

- * **Line** pane: Click a radio button to specify a **Solid** or **Dashed Line** and type a value into the **Weight** text box to set the line thickness.
- * **Color** combo-box: Click the arrow button to display a color palette. Click on a **Color** button to select a pen line color and then click **OK** to close the color palette box.

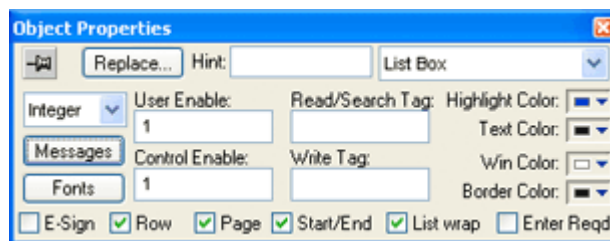
- **List Box Object** button (InduSoft Web Studio v6.1 SP6

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

- Check (*enable*) the **Enter Reqd** 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 Reqd** field to write the value of a selected (highlighted) message the write tag automatically.

To add list box objects to a screen:

- Click the **List Box** button on the *Active Objects* toolbar.
- 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.
- 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 pop-up menu.

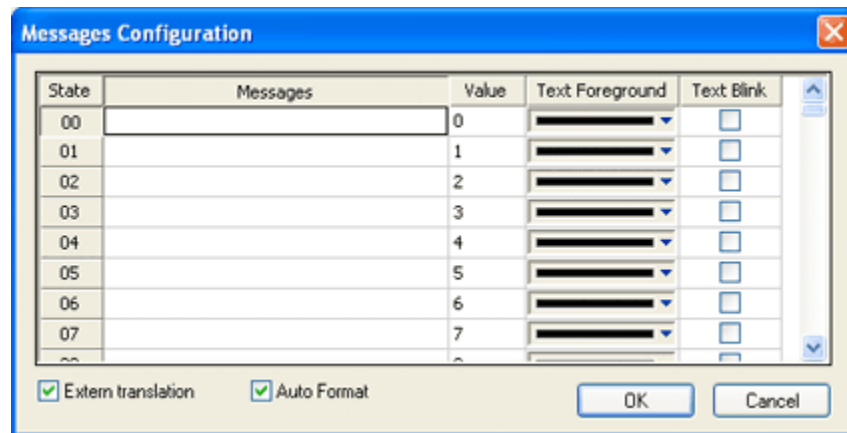
You can use this dialog to specify the following parameters:

- **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.



Messages Configuration Dialog

Use the parameters on this dialog as follows:

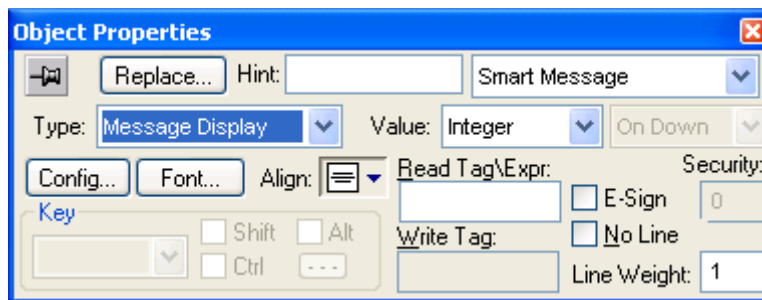
- * **State** field (*read-only*): Use this field to view the indexed individual messages. IWS numbers this field based on the **Read/Search Tag** type you selected:
 - Boolean**: Provides two valid states, labeled 0 and 1
 - Integer**: Provides 255 valid states, labeled 1 to 255
 - LSB**: Provides 32 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. When the *Color* dialog displays, click on a color to select it, and close the dialog.
 - * **Text Blink** check-box: Click (*check*) to cause a selected message to blink, once per second, when it displays.
 - * **Extern 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.
- **Fonts** button: Click to open the *Font* dialog, which allows you to change the characteristics (style, size, and so forth) of the message font.
 - **User Enable** text box: Type a tag, expression, or a (nonzero) number to select a message in the runtime application. The default is 1 (true, enabled).
 - **Control Enable** text box: Type a tag, expression, or a (nonzero) number to select a message in the runtime application — depending on the current value of the **Read/Search Tag**. The default is 1 (true, enabled).

- IWS bases this parameter on the **Value** field (*Messages Configuration* dialog) 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 (*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** check-box: Click (*check*) to prompt the user to enter the Electronic Signature before executing the dynamic.
- **Row** check-box: Click (*check*) to include set up and set down arrows in the list box object scroll bar.
- **Page** check-box: Click (*check*) to include page up and page down arrows in the list box object scroll bar.
- **Start/End** check-box: Click (*check*) to include home and end arrows in the list box object scroll bar.
- **List wrap** check-box: 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.
- **Enter Req'd** check-box: 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).
- **Smart Message Objects** button (): Click to create one or more smart message objects, which you can use to display messages and graphics based on tag values when you execute the application. 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 switch (toggle between) messages by clicking on the object during the 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:

- Click the **Smart Message** button and position the mouse on the screen.
- 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.
- 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 the runtime and the features supported by it:
 - * **Message Display** (*default*)
 - * **Multistate Indicator**
 - * **Multistate Pushbutton**
- **Value** drop-down list: Click to select the tag values used to index the message 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.

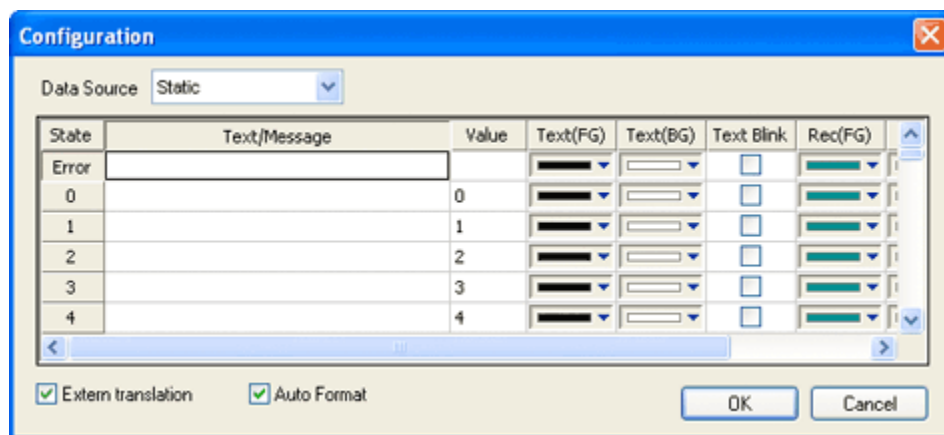


Note:

If Multistate Pushbutton is the Smart Message type, only 16 different messages can be associated with the object, even for Integer or LSB value types.

- **Read Tag/Expr** text box: Enter the name of an integer or a Boolean tag. The value of this tag will determine which message will be displayed by the object during the runtime.

- **Write Tag** text box (*optional and available for Multistate Pushbutton only*): Enter the name of an integer or a Boolean 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** (*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 applications 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 the application during the 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 dynamic.
- **Security** text box (*available for Multistate Pushbutton only*): Security System Access Level required for the object/dynamic.
- **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...** button: Launches the *Configuration* dialog, where you can configure the messages for the object, as follows:



Sample 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 for further details about the format of the text file supported by the Smart Message object when the Data Source = Text File.

- * **Extern translation:** When this option is checked, the text displayed by the object will be controlled by the Translation Tool, during the runtime.
- * **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 the 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 the 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 the runtime.
Text (BG)	Background color for the messages displayed during the runtime.
Text Blink	If checked, the message text will blink during the 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 the 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 the runtime. If you do not specify the path, the bitmap file must be stored in the application's directory.
Transparent	Select the color that will be transparent in the graphic file, if the En. Transparent check-box is checked.
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 type Message Display.

 **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 the 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 the 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.

- **Pushbuttons** button (): Click to create a pushbutton object using the *Command* dynamic object property 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:

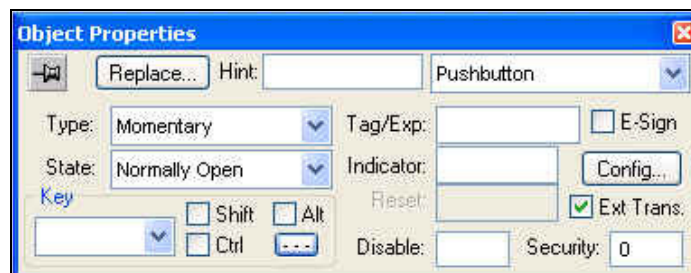
- Rectangular with a faceplate and indicator light
- Rectangular without a faceplate or indicator light (*default*)
- Rectangular with a 3-D
- Rectangular with a floating appearance

To add one or more pre-configured buttons to a screen:

- Click the **Pushbutton** button, and position the mouse (pointer) on the screen.
- 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.

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



Object Properties: Pushbuttons

⇒ **Tip:**

Alternatively, you can right-click on the pushbutton object or highlight the object, press the **Alt+Enter** keys, and select **Properties** from the resulting pop-up menu to open the *Object Properties* dialog.

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.

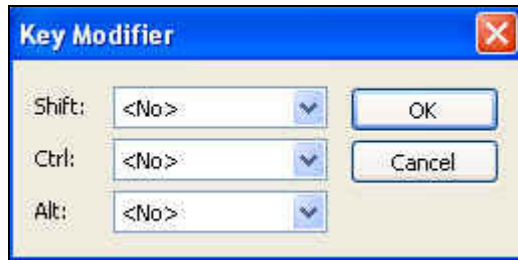
- **Tag/Exp** text box: Type a tag or an expression to accomplish the following:
 - * Type in a tag to receive the **Write Value** from the appropriate state (**Open** or **Closed**) area in the *Configuration* dialog.
 - * Type an expression to execute **On Down**, when you press the pushbutton down.

✎ **Note:**

IWS *does not* write the result of any expression in the **Tag/Exp** field into a tag.

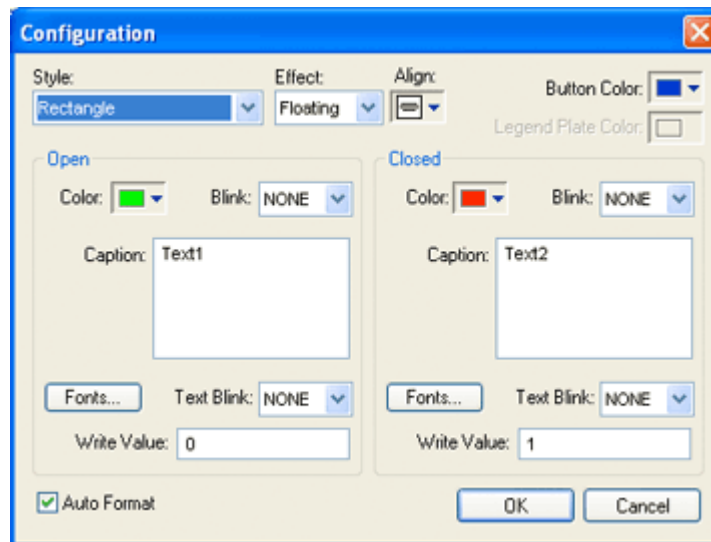
- **Indicator** text box: Type a tag to define an indicator that causes the button to change to a specified color when the tag value matches one of two specified values. You must define both the colors and tag values in the *Configuration* dialog. If you leave this field blank, the indicator changes color automatically when you press the button.
- **E-Sign** check-box: Select (*check*) this option to prompt the user to enter the Electronic Signature before executing the dynamic.
- **Reset** text box (*active for Latched pushbutton type only*): Type a tag to control the button's latched state, as follows:
 - * Type a zero into the tag's value and the button will remain in a latched state after you press it.
 - * Type a nonzero value into the tag 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 (*check*) the box 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.



Key Modifier Dialog

- **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 property.
- **Ext Trans.** check-box: Click (*check*) to translate the text automatically using pre-configured translation worksheets.
- **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:



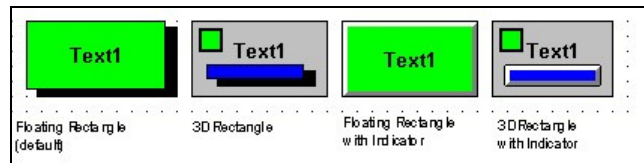
Configuration Dialog

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.

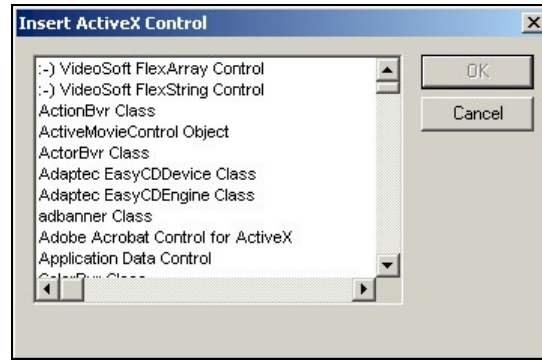
A legend plate encloses a button and indicator light. This field becomes inactive if the pushbutton **Style** does not include an indicator.
- **Open** and **Closed** areas: The following parameters are used to configure the appearance of a pushbutton object in its open and closed states.
 - **Color** 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.

- **ActiveX Control** button (): Click 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 registered on your computer.



Insert ActiveX Control Dialog

Click on one or more components in the list, and then click the **OK** button to close the dialog and display all selected components on your screen.

ActiveX controls are components designed according to a standard. Because IWS 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 the application (e.g. Object Color, FileName, URL, and so forth)
- **Methods:** Functions from the ActiveX object that can be triggered by the application (e.g. open a dialog, execute a calculation, and so forth)
- **Events:** Internal messages that can trigger the execution of expressions in the application (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 the application with the ActiveX control:

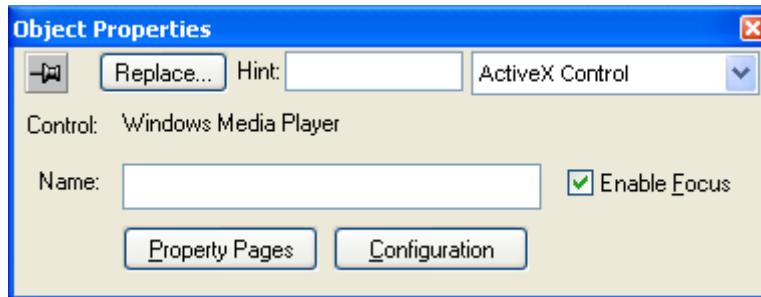
- By using the ActiveX functions [XGet\(\)](#), [XSet\(\)](#) and [XRun\(\)](#)
- OR
- By using the Object Properties window to configure the object

➔ Caution:

- When using ActiveX controls in your application, make sure that the target station (runtime station) has the same ActiveX properly registered. The IWS application files include links to the ActiveX controls; however, the installation of these controls on the target station must be executed manually. Furthermore, when ActiveX controls are used on screens open in remote Web Thin Clients, the ActiveX controls must also be manually installed on the Web Thin Client stations. Consult your ActiveX provider for further information about how to install.

- When configuring applications with ActiveX for CEView, make sure that the ActiveX control used in the application is supported on the platform (Windows CE operating system and process type) where you intend to run the application. Consult your ActiveX provider for further information about the supported platforms.

Double-click the ActiveX control to open the *Object Properties* dialog box.



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, for the application on the Name field (e.g. MyControl). This name is used to reference the object when configuring the ActiveX functions from IWS 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 window depend on the implementation of each ActiveX control. Use this interface to set properties that should not be changed during the runtime (fixed properties).

The Configuration button on the Object Properties window opens dialogs that allow you to do the following:

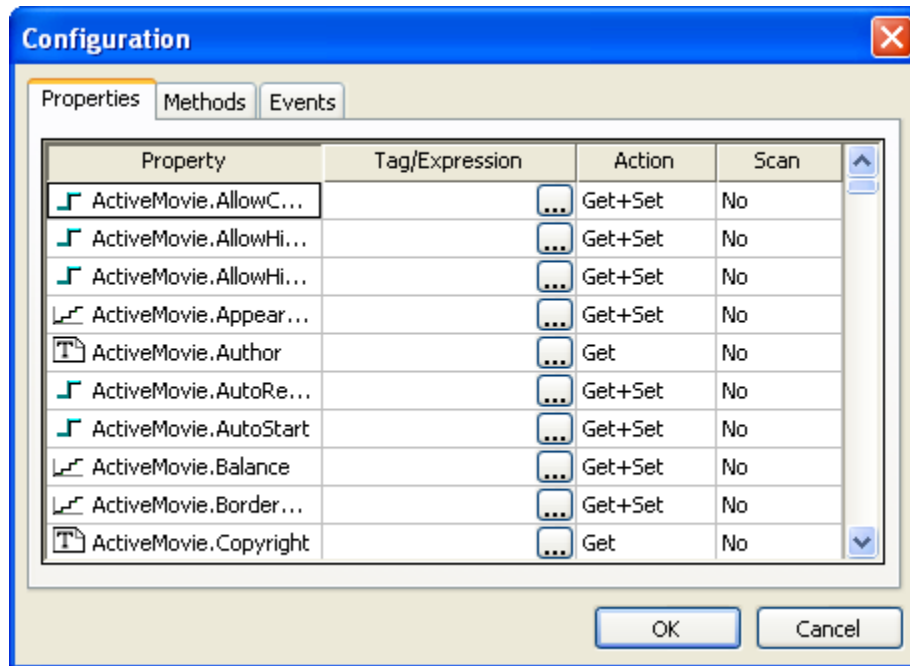
- Associate tags to properties of the ActiveX object
- Trigger methods from the ActiveX object based on tag change
- Configure scripts, which are executed when Events from the ActiveX object occur

The following sections describe how to configure these interfaces.

- Notes:**
- 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 Windows Media Player ActiveX control. Although the name of the properties, methods and events varies depending on each ActiveX control, the configuration interface is the same for any ActiveX control. The concepts described here apply to all of them.

– **Configuring Properties**

The Properties tab provides a grid with the following fields:



Configuration Dialog – Properties Tab

- * **Property:** List 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, and 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 the application), 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:

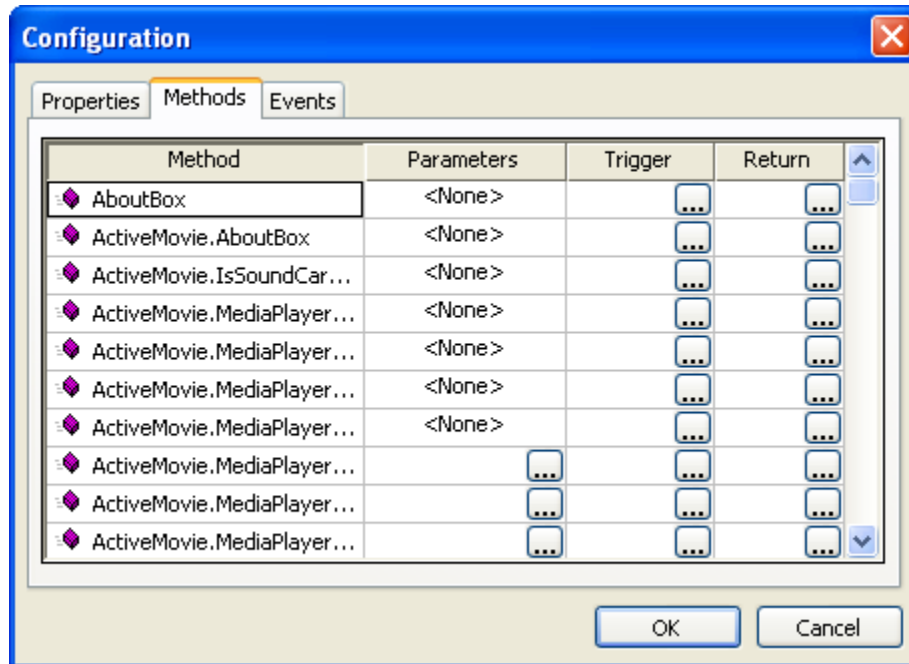
Scan	Description
No	The value of the ActiveX property is read and written to the tag configured in the Tag/Expression field, only when the screen with the ActiveX object is open, and when the ActiveX object sends a message to IWS to update this tag.
Always	IWS keeps polling the value of the ActiveX property 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 (application) 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).

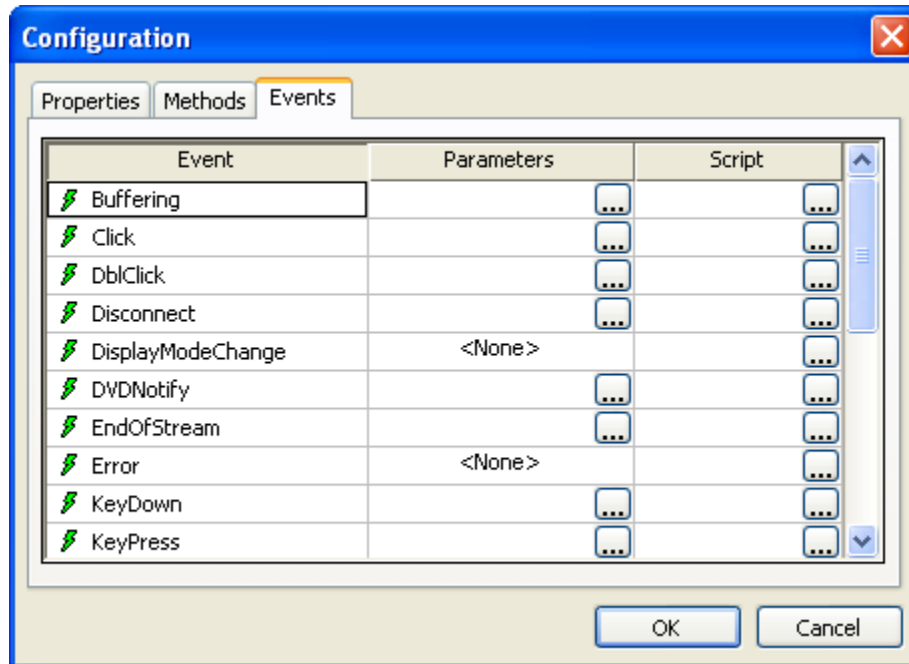
⇒ **Tip:**

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 window displays only the expression configured in the first line of the script.

- **.NET Control** button (): Click to place .NET Framework components on your screen.

.NET Components are designed according to the Microsoft .NET Framework, which is a standard for modular programming technologies. Because IWS is an .NET container, you can configure and run .NET Components in the screens created with IWS. .NET Components provide the following interfaces:

- **Properties:** Variables whose values can be read and/or written for the application (e.g. Object Color, FileName, URL, and so forth)
- **Methods:** Functions from the .NET Framework component that can be triggered by the application (e.g. open a dialog, execute a calculation, and so forth)
- **Events:** Internal messages that can trigger the execution of expressions in the application (e.g. Mouse_Click, Download_Completed, and so forth)

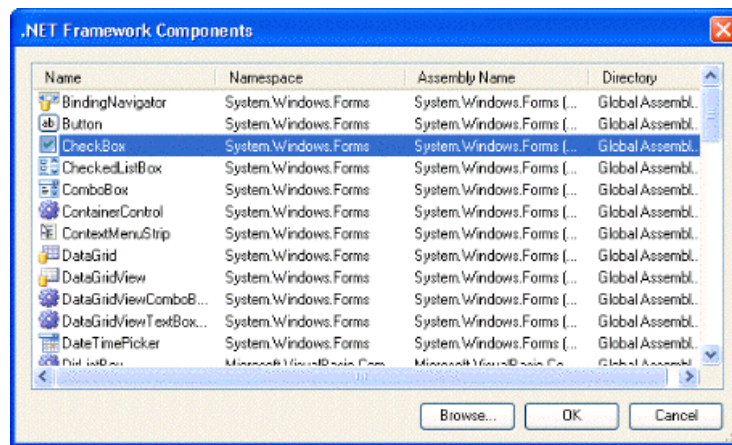
The name of the properties, methods and events supported by each .NET Framework component depends on its own implementation.

Caution:

When using .NET Framework components in your application, make sure that the target station (runtime station) can support the same components and that they are properly installed and registered. The IWS application files include links to the components; however, the installation of these components on the target station must be done separately. Furthermore, when components are used on screens open in remote Web Thin Clients, the components must also be manually installed on the Web 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 Framework component provider for further information about how to install.

To select and place a .NET Control object in your application screen:

1. Click the **.NET Control Object** tool on the *Active Objects* toolbar, or choose **Insert > .NET Control** from the main menu bar.
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 application 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 IWS include unregistered components in the *.NET Framework Components* dialog by editing the *<ApplicationName>.APP* file to add this parameter:

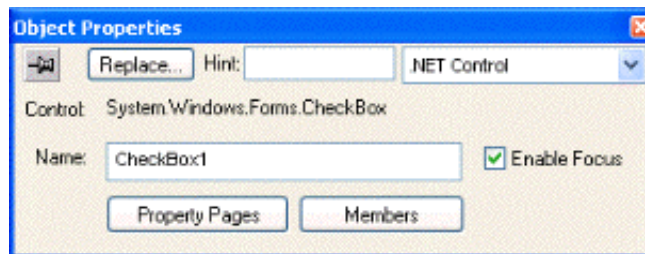
```
[Execution Environment]
DotNetControlPath=<OptionalPath>
```

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 the application 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 window displays the name of the .NET Control. You must assign a name (alias) to the .NET Control, for the application on the Name field (e.g. CheckBox1). This name is used to reference the component when using the scripting languages.

📌 **Note:**

You should not configure two .NET Components 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 window depend on the

implementation of each .NET Component. Use this interface to set properties that should not be changed during the runtime (fixed properties).

The **Members** button on the *Object Properties* window opens 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.



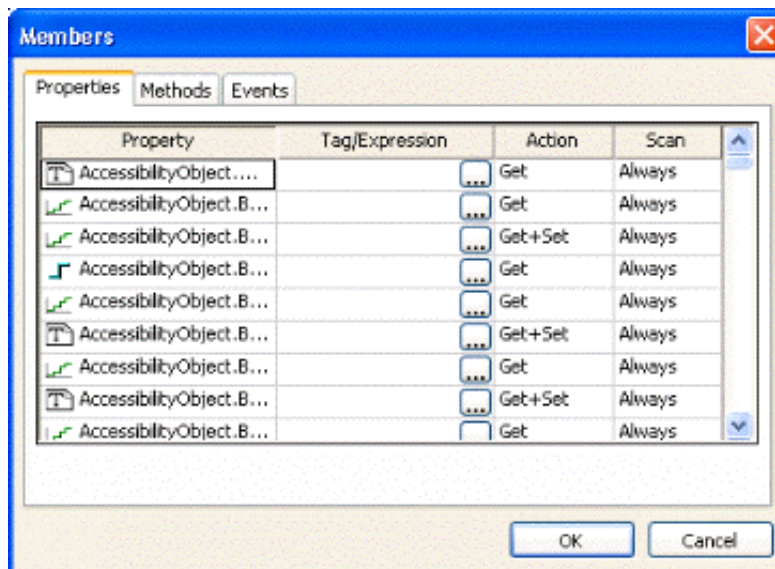
Notes:

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:

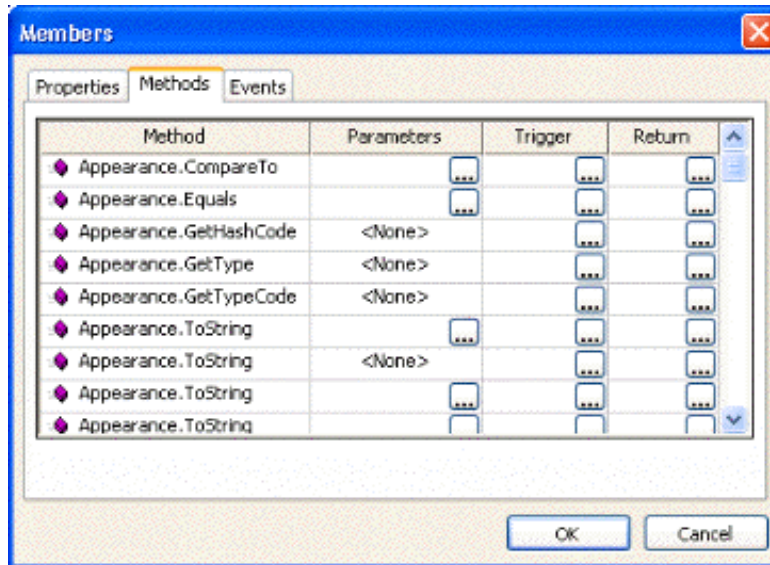
Action	Description
Get	Read the value of the property and write it to the tag configured in the Tag/Expression field.
Set	Write the value from the tag or expression configured in the Tag/Expression field into the property.
Get+Set	Executes both actions (Get and Set). However, when opening a screen with 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 the application), 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:**

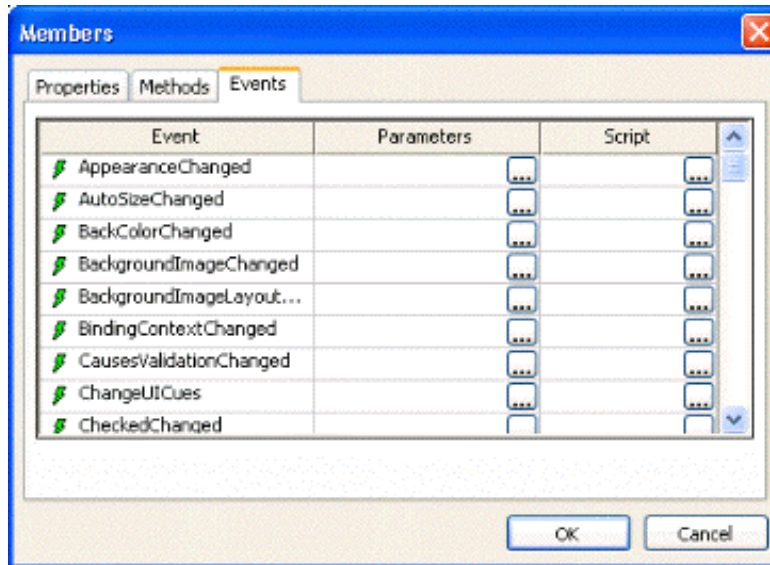
When you click the Browse button (⋮), it will display the list of parameters supported by the method, allowing you to associate one tag with each parameter.

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).

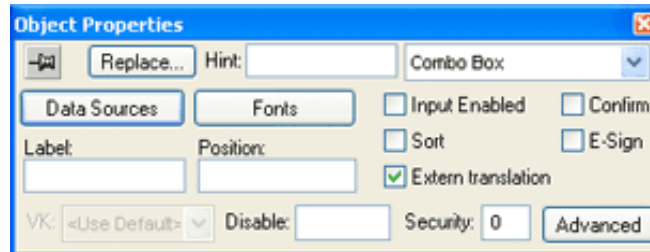
- * **Script:** The script configured in this field will be executed when the event is triggered by the .NET Component.

⇒ **Tip:**

When you click the Browse button (⋮), it will display a dialog with the complete script associated with the event. The main dialog window displays only the expression configured in the first line of the script.

- **Combo-Box** button (): Click to select a single label from a combo-box list of labels. If the list is longer than the space allotted, IWS enables a scroll bar 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:

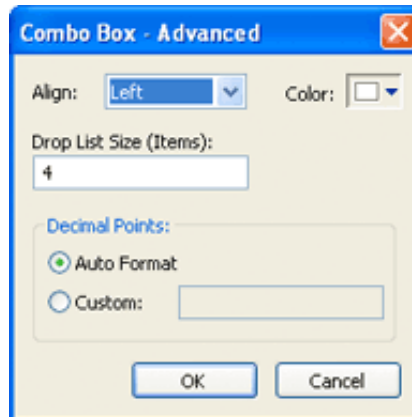


Object Properties: Combo Box

You can use this dialog to set the following parameters:

- **Data Sources** button: Click to open the *Data Sources* dialog (see below).
- **Fonts** button: Click to open a standard *Font* dialog. Use this dialog to change the characteristics of a message font.
- **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.
- **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.
- **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.
- **Extern translation** checkbox: Click (check) to enable automatic translation of the combo box labels using the Translation Tool.
- **Confirm** checkbox: Click (check) to prompt an operator to confirm a command during runtime.
- **E-Sign** checkbox: When this option is checked, the user will be prompted to enter the Electronic Signature before executing the dynamic.
- **VK**: Virtual Keyboard type used for this object. You need to enable the Virtual Keyboard option on the **Project** → **Settings** → **Runtime Desktop** interface before configuring the Virtual Keyboard for this interface.
- **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 property. If you disable the combo box, it appears grayed out during runtime.
- **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 property remains active.

- **Advanced** button: Click to open the *Combo Box - Advanced* dialog:



- o **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.
- o **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.
- o **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.

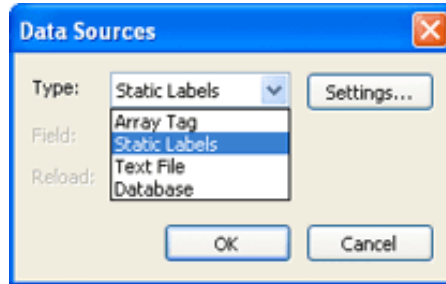
 **Note:**

If this number is less than the *total* number of items in the list, then the drop list will also scroll.

- o *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.

DATA SOURCES

Use the *Data Sources* dialog to configure the items/labels that will be displayed in the Combo Box object.

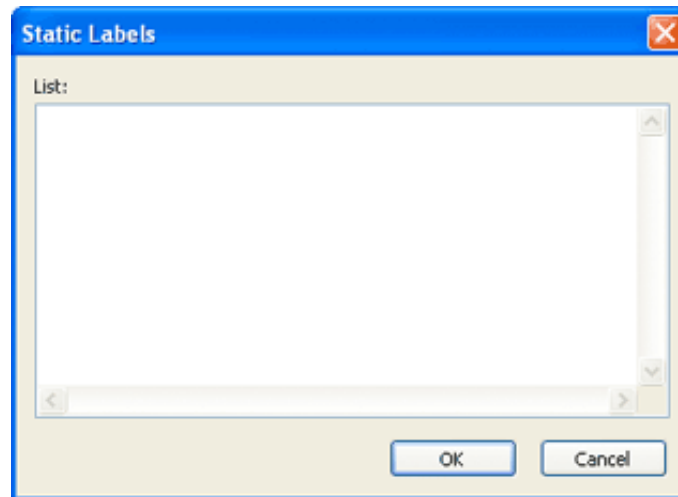


Data Sources dialog

- **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 the **Type** is set to **Static Labels**, you can configure the following settings:



Enter your labels — with one label per line — just as if you were editing a plain text file. The labels are not sorted in any way, so be sure to put them in the order you want them displayed during runtime.

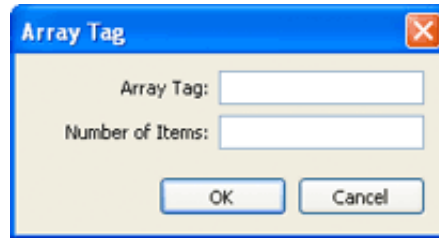
Note:

The first line is designated as *position 0*, the second line is designated as *position 1*, and so forth.

Click **OK** when you're done.

Type: Array Tag

When the **Type** is set to **Array Tag**, you can configure the following settings:

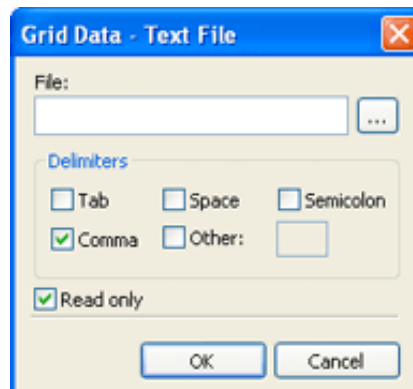


- **Array Tag:** Enter the name of an array tag of String type that contains the items for the combo box.
- **Number of Items:** Specify how much of the array should be displayed in the combo box. Keeping in mind that the combo box counts *array index 0* as the first item, if you enter a value of 4, then the combo box will display *array index 0* through *array index 3* — which is a total of four labels.

Click **OK** when you're done.

Type: Text File

When the **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 the application folder, then you can omit the path in the name.)

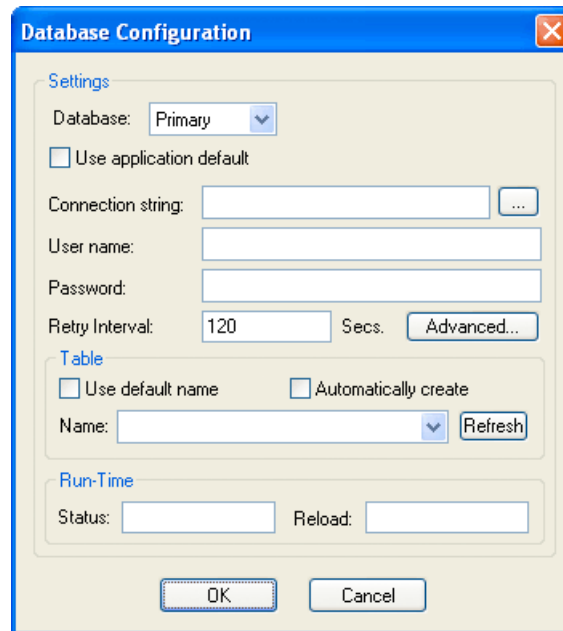
⇒ **Tip:** You can configure tag names between curly brackets (e.g. `{TagName}`) in the **File** field.

- **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:



For more information, please see “Database Configuration Dialog Window” on page 6–14.

- **Alarm** button (): Click to specify an area of the screen in which to display alarm messages.

Next, click in the screen and drag the mouse to designate an area for the messages. When you are finished, double-click on the object to open the *Object Properties* dialog:




Object Properties: Alarm

You can use this dialog to specify the following parameters:

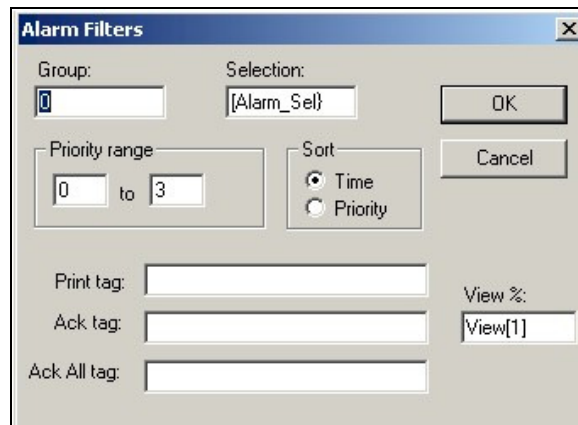
- **On Line** button: Click (*enable*) this button to display on-line alarm messages for the object.
- **History** button: Click (*enable*) this button to display alarm messages from the history files.

⚠ Caution:
You may have to use the **Save to Disk** option, located on the *Alarm* worksheet (**Tasks** tab), to save alarm messages specified as history files.

- **Border** combo-box: Click to define a color for the alarm message border. When the *Color* dialog displays, click on a color to select it, and then close the dialog.
- **Win** combo-box: Click to define a background color for the *Alarm Message* dialog. When the *Color* dialog displays, click on a color to select it, and then close the dialog.
- **PgUp** field: In the runtime, toggling the tag in this field triggers a page up command to the list.
- **PgDown** field: In the runtime, toggling the tag in this field triggers a page down command to the list.
- **Message Format** area: Use the following parameters to define a format for alarm messages. You can format alarms to include dates, times (hours), names, tags, and messages.
- **Font** button: Click to open the *Fonts* dialog and specify a style, size, color, and font type for the alarm message text.
- **DD,MM,YY** check-boxes: Click (*check*) to display the alarm date in the message text.
- **HH,MM,SS,MSS** check-boxes: Click (*check*) to display the alarm time in the message text.

 **Note:**
If you check the **DD,MM,YY** and/or **HH,MM,SS,MSS** boxes, an asterisk (*) character displays between the alarm's date/time and the alarm message.

- **Tag** field: Specify how many characters to allow for a tag name.
- **Message** field: Specify how many characters to allow for a message.
- **Ack** check-box: Click (*check*) to add the acknowledgement time to an alarm message.
- **End** check-box: Click (*check*) to add the normalization time to an alarm message.
- **Selection** button: Click to open the *Alarm Filters* dialog, which allows you to specify filters for alarm messages.



Alarm Filters Dialog

Use the following parameters to specify these filters:

- **Group** field: Enter a value to select and display alarm groups in the alarm summary object.
 - * Enter a zero in the field to select all alarm groups.
 - * Enter any value other than zero to select a specific alarm group.
- **Selection** field: Type the character string you specified in the **Selection** column on the *Alarm* worksheet (**Tasks** tab). IWS filters display alarms by matching this character string against the string specified on the *Alarm* worksheet.

**Note:**

This character string must be the same as the character string on the *Alarm* worksheet.

**Tip:**

If you enter the character string between curly brackets ({ }), you can modify the tag value during runtime.

- **Priority Range** fields: Specify a range (based on the alarm priority specified in the **priority** column of the *Alarm* worksheet) in which to filter and display alarm messages.

For example, if you assign alarm priorities 1 to 5 on the *Alarm* worksheet, and then assign 0 to 4 in these **Priority Range** fields, IWS will display alarm priorities 1 to 4 and will not display alarm priority 5.
- **Sort** area: Use the radio buttons to sort and display alarm messages as follows:
- **Time**: Click (*enable*) this button to sort and display alarms according to the time the alarm was received.
- **Priority**: Click (*enable*) this button to sort and display alarms according to the priorities assigned in the **Priority Range** fields.
- **Print Tag** field: Type a tag name to filter and print all alarms when changes are made to that tag.
- **Ack Tag** field: Type a tag name to filter and acknowledge the active alarm (top of alarm object list) when you change that tag.

**Tip:**

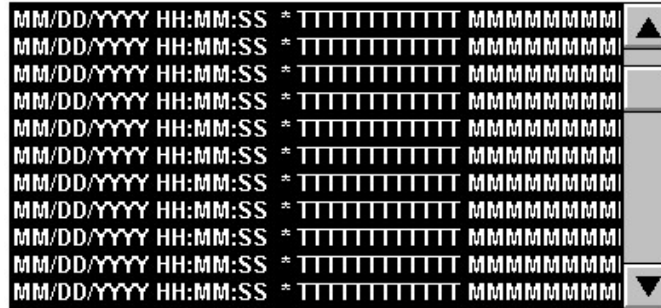
You can use the internal **AckAlr** tag to acknowledge the last alarm from the application.

- **Ack All Tag** field: Type a tag name to acknowledge all filtered, active alarms when you change that tag.


**Tip:**

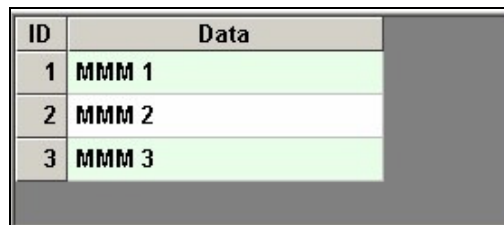
You can use the internal **AckAll** tag to acknowledge all alarms from the application.

- **View %** field: Type a tag in this field (specify a percentage from 0% to 100%) to control how many alarm messages are visible in the alarm list during runtime. You also can use this tag to scroll the alarm list up and down during runtime.



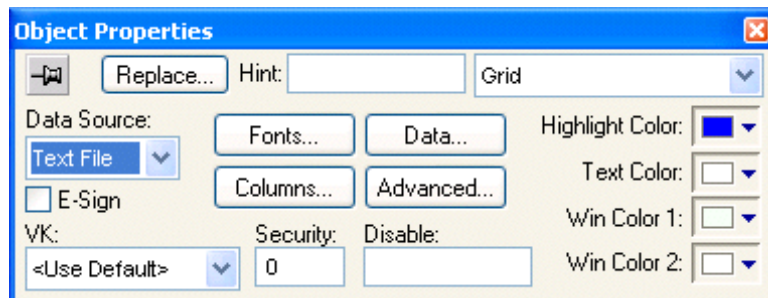
Alarms List with Scroll Bar

- **Grid button** (): 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:
 - ☑ Click the Grid tool.
 - ☑ 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).
 - ☑ Release the mouse button, and the Grid Object will display.



Creating a Grid Object

- ☑ Right-click on the Grid Object, and select Properties from the menu. The Object Properties dialog box will open. Use this dialog to configure the Grid Object's parameters:



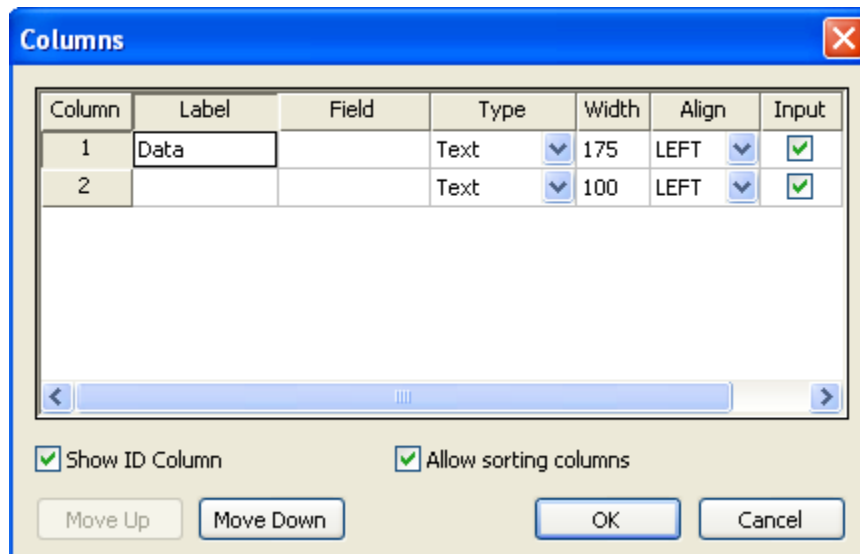
Object Properties: Grid Object

- **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 on the Project Settings Runtime Desktop interface. You can also specify a different virtual keyboard for this Grid Object.
- **Security:** Enter the security system access level required for the object/dynamic.
- **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 the runtime.
- **Text Color:** Select a text color for the selected row, during the 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. You can configure the settings for each column displayed by the Grid object during the runtime, as follows:



Columns Dialog Box

- * **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 the runtime. When the label is blank (i.e.: ""), the


width of the column is set to 0 during the runtime. This option is useful to hide columns during the runtime.

- * **Field:** Enter the name of the field (column) in the SQL Relational Database that the Grid object is linked to. 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 the application's 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 the application to support this feature on the Web Thin Client stations. CEView applications support pictures in bitmap format (*.bmp), but not in icon format (*.ico).
Check-box	Displays check-box interfaces. The check-box will be unchecked if the value read from the file is 0, <NULL> or "FALSE"; otherwise, the check-box 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.

 **Notes:**

- 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 the 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 the 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 the application and configuring

its name on the <ApplicationName>.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:** Enable (check) to allow the user to enter data in this column during the runtime.
- * **Key:** Designate a shortcut for sorting the values in this field. 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 applications for runtime devices that do not provide a mouse or touch-screen interface and only have a keyboard for interacting with the application during the 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>
```

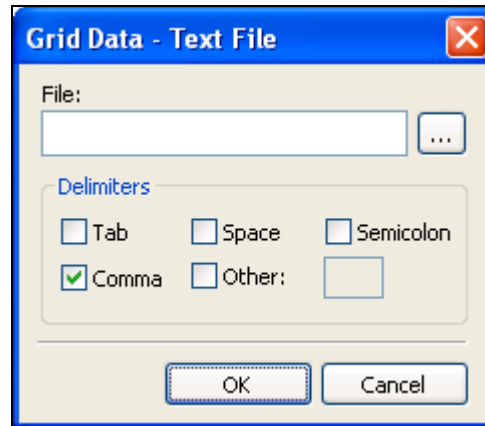
- * **Show ID Column:** Check to display the number of each row, automatically.
- * **Allow sorting columns:** Check to enable the user to sort the values in the columns during the runtime, either by clicking on the label or by using the shortcut configured for each column. This option is disabled if 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:** Click to launch the Data dialog, where you can specify a data source for the Grid object.

This dialog allows you to configure the data source settings, as follows:

- * **Data Source – Text File**
When the Data Source type is set to Text File, you can configure the following settings:



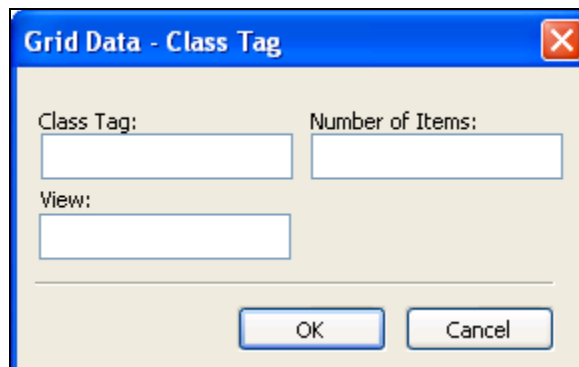
- **File:** Enter the name of the text file source. You can either type the file name and its path or click the ... button to browse for it. (If the file is stored in the application folder, you can omit the path in the name.)
- **Delimiters:** Set the delimiter(s) used in the data source file. For instance, 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.

⇒ **Tip:**

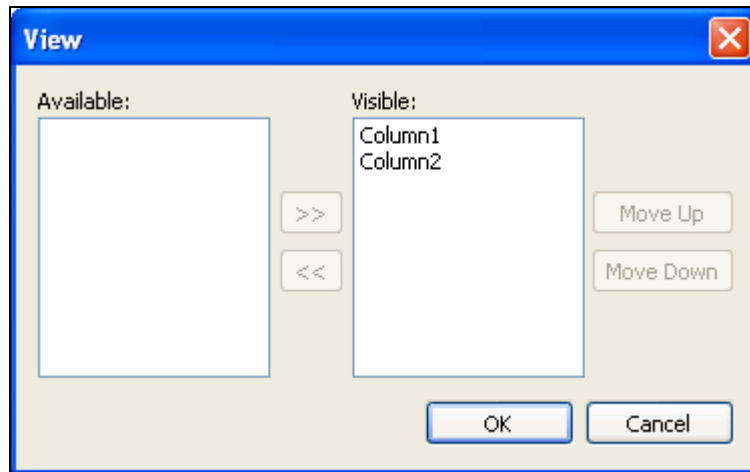
You can configure tag names between curly brackets {TagName} in the File field.

* Data Source – Class Tag

When the Data Source type is set to Class Tag, you can configure the following interface:

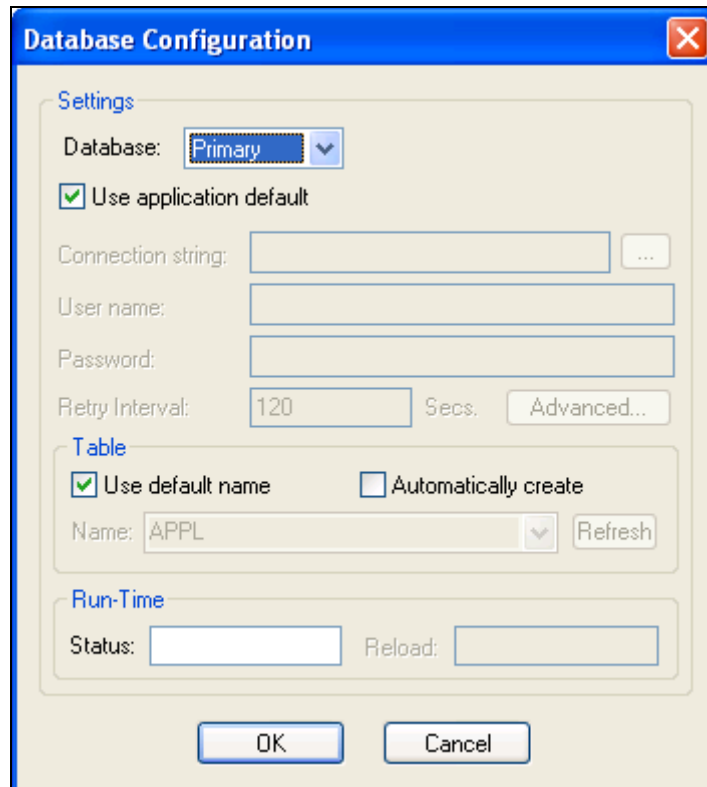


- **Class Tag:** Enter the name of the main class tag source. (Do not specify a specific member of the class tag.) You can specify the initial array position in this field (e.g. Mytag[10]); otherwise, 0 (zero) will be used as the initial position by default.
- **Number of Items:** Enter the number of 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 the runtime, the grid object launches a dialog, allowing the user to show/hide each column or modify their positions.



* Data Source – Database

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



Please refer to the Database Configuration Dialog Window for further information about this dialog.

- **Advanced:** Click to launch the Advanced dialog, where you can configure several settings for the Grid object.

This dialog allows you to configure the advanced settings, as follows:

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 the 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 the runtime. In addition, you can select different rows by writing their values in this tag.
- * **Condition:** This field can be configured with the data filter expression. This expression must follow the basic syntax of <ColumnName> <Comparison Operator> <Value> (e.g. ColumnX > 200). When using Text File or Class Tag for Data Sources, the <ColumnName> is the value specified in the Label. When using Database for the Data Source, the column is the value specified in the Field. (In this case, if the Field is left blank, the column value specified is the Label.)

⇒ **Tips:**

- You can combine several conditions simultaneously in the Condition

filed, using the logic operators AND, OR, and NOT. For example, ColumnAge>'10' OR ColumnName='John' AND ColumnDate>'05/20/2003'.

- You can use wildcards (* and ?) in the Condition field to filter data.
- You can configure tags between curly brackets {TagName} in the Condition field to change the filtering condition during the runtime.

- * **Print:** When the tag configured in this field toggles, the data currently filtered in the object is sent to the printer.
- * **Reload:** When the tag configured in this field is toggled, the object reloads the data from the data source and displays it.
- * **PDF Trigger field:** Type a Tag in this field. When the value of the Tag changes, the data currently filtered in the Alarm/Event Control is distilled to a PDF file and saved to the path specified in the **PDF Filename** field below.
- * **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 applications running on Windows CE or Web Thin Client.


- * **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 the 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 the runtime.
- * **Conditional Check-box:** When this option is checked, the user cannot uncheck a check-box on the Grid during the runtime, unless all preceding check-boxes 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 the runtime, displaying the label of each column.

- * **Show gridlines:** When this option is checked, the gridlines of the Grid object are visible during the runtime.
- * **Ext. translation:** When this option is checked, the text displayed by the Grid object will be susceptible to the Translation Tool during the 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.

⇒ **Tips:**

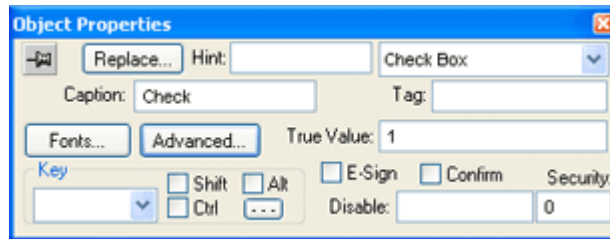
The Export feature is an easy and powerful tool to transfer data from different data sources to tags. After exporting the data to tags, you can use different tasks to manipulate the data, such as the FileWrite() function, or the Recipe or Report tasks to save the data in text files (e.g. CSV files).

- * **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” above.
- **Check Box Object** button (): Click this button to create a check box object on your screen.

The **Check Box Object** is useful to create interfaces where users can enable/disable an option on the display. To create a **Check Box Object**:

- Click in the drawing area and drag the mouse/cursor to draw the check box and its label.
- Release the mouse button when the object is the size you want.

- Double-click on the object to view the *Object Properties* dialog:



Object Properties Dialog: Check Box

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

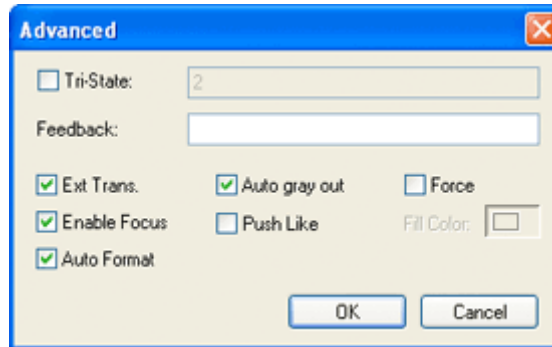
- **Caption**: Specify a caption by typing the text into the text box. You can include a tag by enclosing it in curly brackets (e.g. {*tagname*}).
- **Fonts**: Specify a font style for the caption by clicking the **Fonts** button.
- **E-Sign**: When this option is checked, the user will be prompted to 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 (*check*) the box 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.



Key Modifier Dialog

- **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 the 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** 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 blank, the third state will be displayed when the tag configured in the **Tag** field assumes the value specified in the **Tri-State** field.
- * **Feedback**: Value that indicates the state of the object (TRUE, FALSE or Third-State). If the **Feedback** field is left blank, 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 the runtime by the navigation keys.
- * **Push Like**: When this option is checked the control is displayed as a button, instead of the 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.

There are two main modes of operation for this object: **Normal Mode** and **Tri-State Mode**. These modes are described below:

NORMAL MODE

When the **Tri-State** option is unchecked, the object operates in **Normal Mode**. Therefore, it can assume two states only:

State	Shape	Shape (Push Like)
FALSE	<input type="checkbox"/> Check	<input type="button" value="Check"/>
TRUE	<input checked="" type="checkbox"/> Check	<input type="button" value="Check"/>

Normal Mode States

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. Otherwise, the state is set to FALSE. If the **Feedback** field is left blank, the tag configured in the **Tag** field will be used as the **Feedback** tag.

When the user clicks on the object, the tag configured in the **Tag** field is updated according to the following table:

Current Status	Value written to the tag configured in the "Tag" field when the user clicks on the object
FALSE	Value configured in the True Value field
TRUE	NOT (Value configured in the True Value field)

Note:

When the value configured in **True Value** is a string, the tag configured in the **Tag** field will be toggled between an empty string and the **True Value**. If **True Value** is left in blank, the tag configured in the **Tag** field will be toggled between "UNSELECTED" and an empty string.

TRI-STATE MODE

When the **Tri-State** option is checked, the object operates in **Tri-State Mode**. Therefore, it can assume three states:

State	Shape	Shape (Push Like)
FALSE	<input type="checkbox"/> Check	<input type="button" value="Check"/>
TRUE	<input checked="" type="checkbox"/> Check	<input type="button" value="Check"/>
TRI-STATE	<input checked="" type="checkbox"/> Check	<input type="button" value="Check"/>

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, the tag configured in the **Tag** field will be used as the **Feedback** tag.


⚠ Caution:

The **Tri-State** field must not be configured with the same value as the **True Value** field or with an empty string value.

Current Status	Value written to the tag configured in the "Tag" field when the user clicks on the object
FALSE	Value configured in the True Value field
TRUE	NOT (Value configured in the Tri-State field)
TRI-STATE	NOT (Value configured in the True Value field)

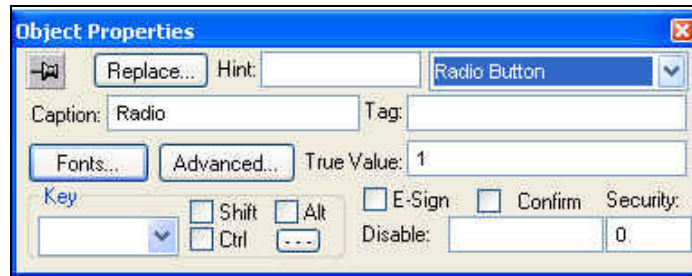
📌 Note:

- If "NOT (Value configured in the **True Value** field)" is equal to **Tri-State**, the value assigned to the tag configured in the **Tag** field will be the minimum signed integer value different from **True Value**.
- When **True Value** is a string, "NOT (Value configured in the **True Value** field)" will result in an empty string. If **True Value** is an empty string, "NOT (Value configured in the **True Value** field)" will result in "UNSELECTED".

- **Radio Button Object** button (): Click this button to create a check box object on your screen.

The **Radio Button Object** is useful to create interfaces where users can choose one option from multiple display options. To create a **Radio Button Object**:

- Click in the drawing area and drag the mouse/cursor to draw the radio button and its label.
- Release the mouse button when the object is the size you want.
- Double-click on the object to view the *Object Properties* dialog:



Object Properties Dialog: Radio Button

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

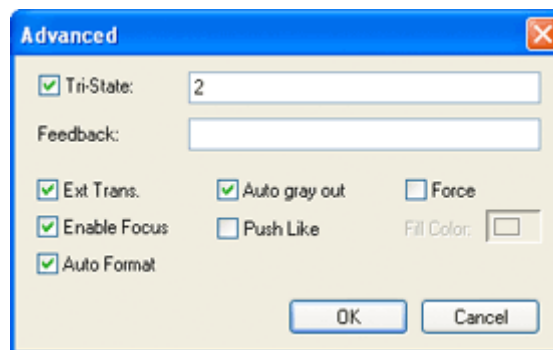
- **Caption**: Specify a caption by typing the text into the text box. You can include a tag by enclosing it in curly brackets (e.g. { *tagname* }).
- **Fonts**: Specify a font style for the caption by clicking the **Fonts** button.
- **E-Sign**: When this option is checked, the user will be prompted to enter the Electronic Signature before executing the command.
- **Confirm** 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 radio button.
- 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 (*check*) the box 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.



Key Modifier Dialog

- **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 the 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 blank, the third state will be displayed when the tag configured in the **Tag** field assumes the value specified in the **Tri-State** field.

- * **Feedback:** Value that indicates the state of the object (TRUE, FALSE or Third-State). If the **Feedback** field is left blank, 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 the runtime by the navigation keys.
- * **Push Like:** When this option is checked the control is displayed as a button, instead of the 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.

There are two main modes of operation for this object: **Normal Mode** and **Tri-State Mode**. These modes are described below:

NORMAL MODE

When the **Tri-State** option is unchecked, the object operates in **Normal Mode**. Therefore, it can assume two states only:

State	Shape	Shape (Push Like)
FALSE	<input type="radio"/> Radio	<input type="button" value="Radio"/>
TRUE	<input checked="" type="radio"/> Radio	<input type="button" value="Radio"/>

Normal Mode States

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. Otherwise, the state is set to FALSE. If the **Feedback** field is left blank, the tag configured in the **Tag** field will be used as the **Feedback** tag.

When the user clicks on the object, the tag configured in the **Tag** field is updated with the value configured in the **True Value** field.

TRI-STATE MODE

When the **Tri-State** option is checked, the object operates in **Tri-State Mode**. Therefore, it can assume three states:

State	Shape	Shape (Push Like)

State	Shape	Shape (Push Like)
FALSE	<input type="radio"/> Radio	<input type="radio"/> Radio
TRUE	<input checked="" type="radio"/> Radio	<input type="radio"/> Radio
TRI-STATE	<input checked="" type="radio"/> Radio	<input type="radio"/> Radio

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, the tag configured in the **Tag** field will be used as the **Feedback** tag.

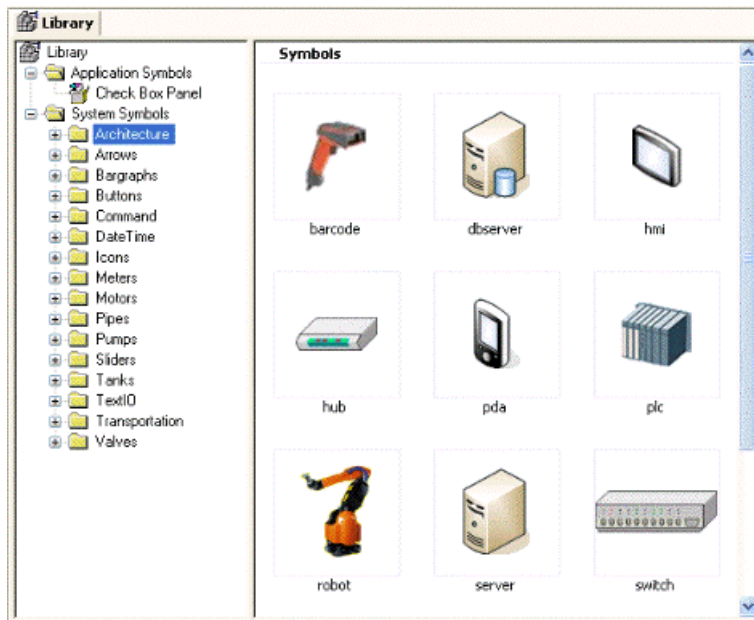
Caution:
 The **Tri-State** field must not be configured with the same value as the **True Value** field or with an empty string value.

Current Status	Value written to the tag configured in the "Tag" field when the user clicks on the object
FALSE	Value configured in the True Value field
TRUE	NOT (Value configured in the Tri-State field)
TRI-STATE	NOT (Value configured in the True Value field)

Using the Library

Symbols are reusable objects (or groups of objects) that you can store for reuse. IWS provides access to an extensive symbol library that enables you to add and reuse symbols quickly and easily. You also can modify existing symbols in just a few seconds.


From the menu bar, select **View > Library** to open the following library of previously configured objects and symbols, including several objects with dynamics:



Library

You can add symbols from this library to your application screen, which saves time during development. You also can upgrade this library with new symbols by right-clicking on a screen button (in the *Workspace*) and choosing the **Send to library** option from the resulting pop-up menu. The application inserts the screen into the library with all its objects.

 **Note:**

Using the **Library** menu option is the same as using the **Library** button  on the *Standard* toolbar.

To add an image to the display:

- Click on a category in the left panel to preview available images.
- Double-click on a category to import a copy of the image to the display screen, which keeps your image and closes the *Libraries* dialog.
- Click anywhere in the display screen to place the selected image.

Note:
Most symbols have predefined properties. To change these properties, use the **Replace** tab located on the *Object Properties* dialog. You can add a user screen to the *Symbol* library. Develop the screen as **.scr** and copy it to the **\LIB** directory where you installed IWS.

Using Paste Link

From the main menu bar, select **Edit > Paste Link** to paste a *linked object* (only **.BMP** files) onto the active screen, while maintaining a connection to the source. A linked object is information (the object) created in a source file (another project or another screen). **Paste Link** automatically updates a linked screen object whenever you update the source file, but the linked object does not become part of the screen. If you put a linked file in an application folder or a subfolder, you can download the file with the application to the runtime workstation.

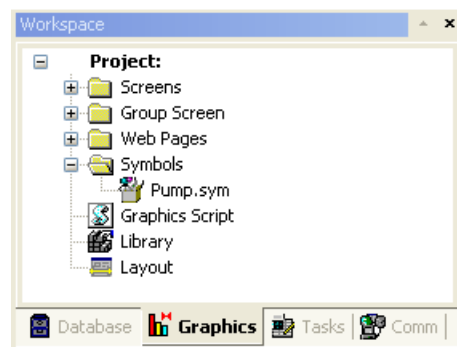
Note:
This option does not work in Windows CE applications.

Symbols Folder

A Symbol is an Object (or Group of Objects) that is saved to the *Symbols* folder (in the *Graphics* tab of the Workspace), so that you can reuse it again and again in your application 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.

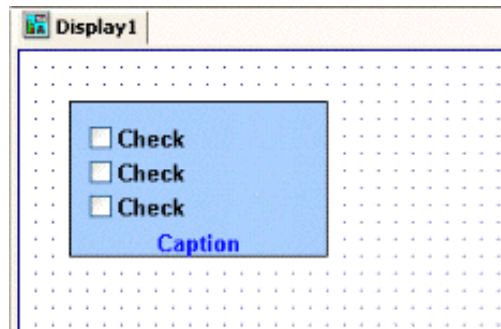


Symbols Folder in the Graphics Tab

CREATING A MASTER SYMBOL

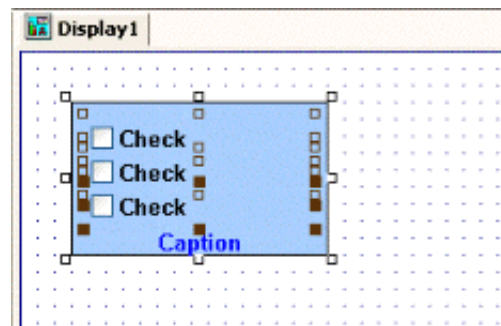
To create a Master Symbol and save it to the *Symbols* folder:

1. Design your Symbol just as you would normally draw an application 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.

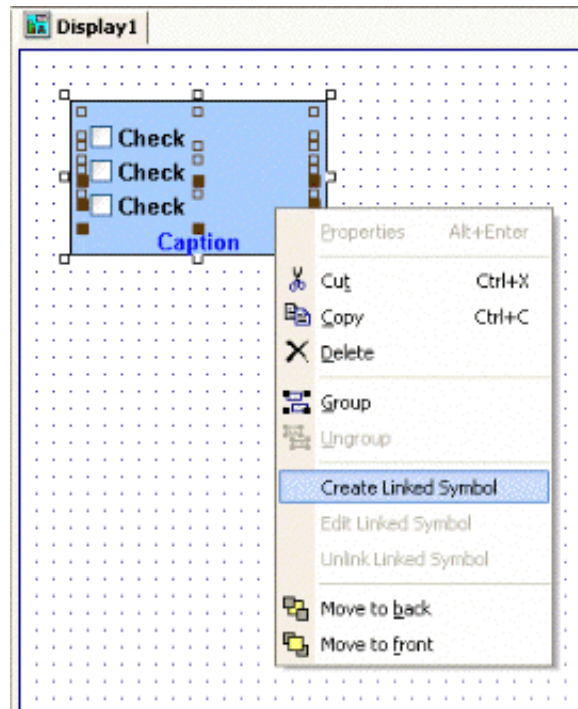


Selecting the Objects

Note:

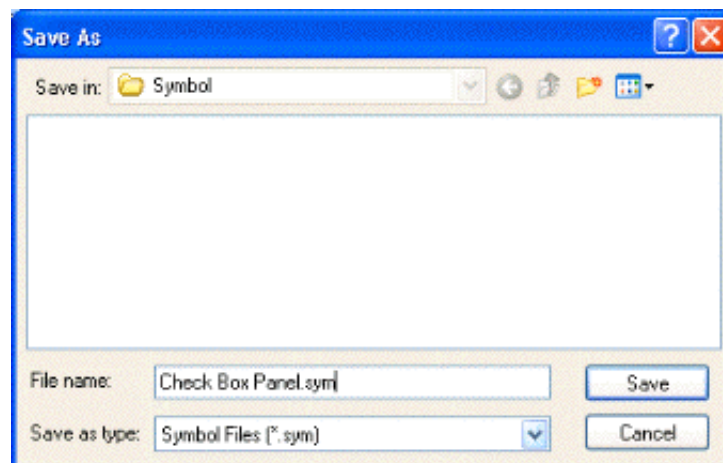
It is not necessary to make a Group out of two or more Objects before saving them as a Symbol. Saving the Objects together as a Symbol effectively groups them as well.

3. Right-click on the selected Object(s) and choose **Create Linked Symbol** from the contextual menu, or choose **Edit > Copy to** from the main menu bar.



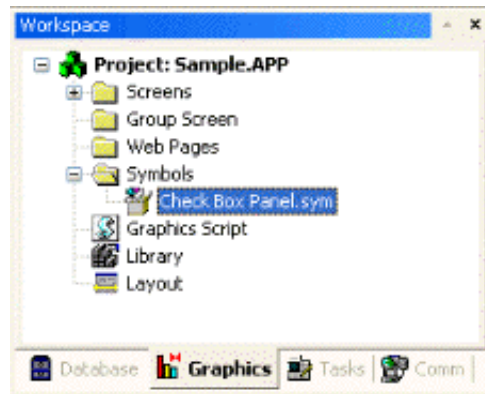
Creating a Linked Symbol

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 directory of your application 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 Workspace.



Symbol File in the Workspace

Note:

The Symbol also appears in the *Application Symbols* folder of the Library.

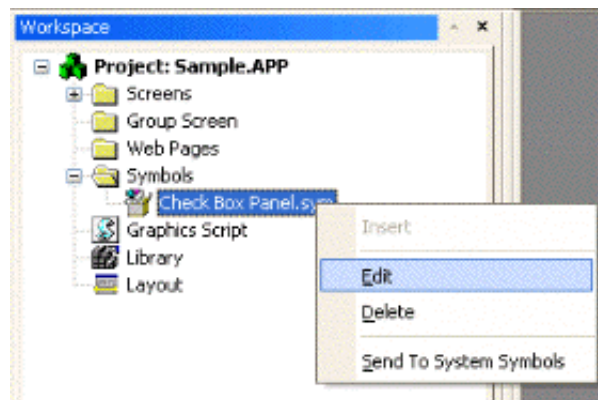
The Symbol is ready to be reused in your project, but the way it is currently saved, every copy will have identical properties. You must now define Custom Properties on the Symbol — that is, the properties you want to be able to customize each time you reuse the Symbol.

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 modifications you make to the Master Symbol will automatically propagate to every linked copy in every application project.

To edit a Symbol:

1. Right-click on the Symbol file in the Symbols folder, and then choose **Edit** from the contextual menu.

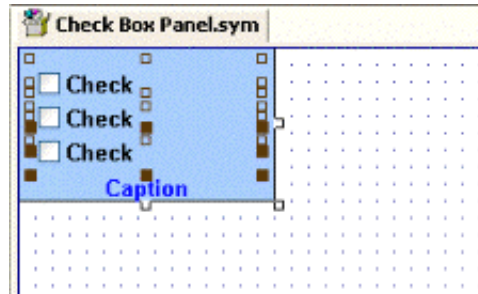


Editing the Symbol File

Tip:

You can also right-click on any instance of the Symbol and choose **Edit Linked Symbol** from the contextual menu.

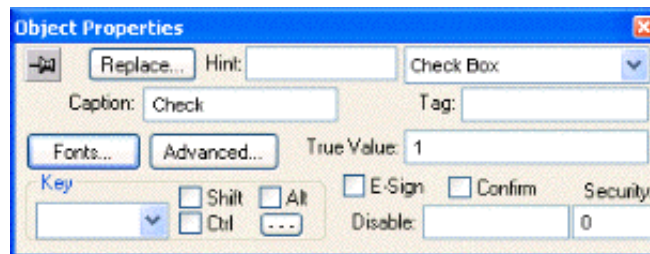
The Symbol file is opened for editing in its own window. This Symbol Editor works in the same way as a regular Screen Editor, except that every Object in the window is part of the Symbol. If you add, move or delete Objects in the Symbol Editor, then you may change the size or shape of the Symbol and 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 of the three Check Boxes and the Tags which the Check Boxes enable/disable, as well as the caption of 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 following syntax:

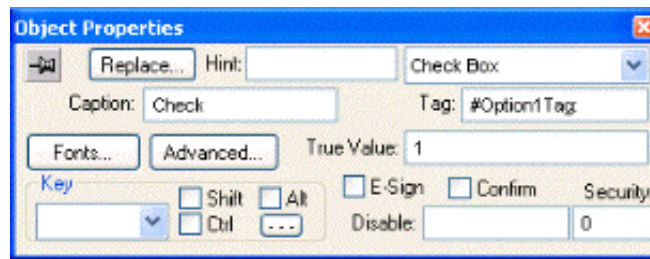
#<Label>: <Default>

...where **<Label>** is a name to identify the property, when you are completing the properties on an instance of the Symbol (see below), and **<Default>** is an optional default value for the property.

Note:

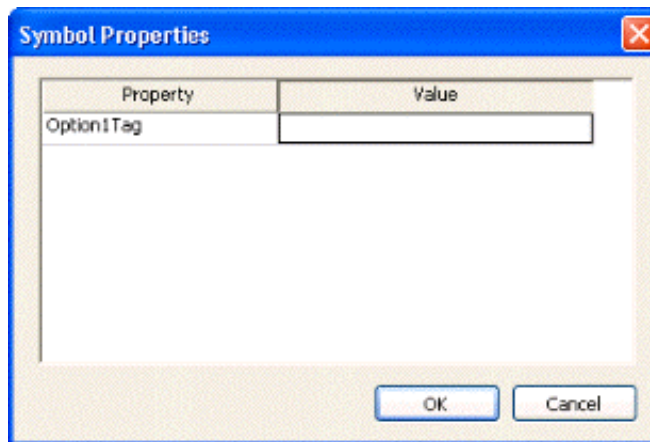
All standard syntax applies to **<Default>**, including Tag names, indirect Tags, Arrays, strings, numerical and boolean values, and scripting expressions. Also, even if you do not want to assign a default value to the Custom Property, you must type the colon character (:) after the **<Label>**.

In this example, we want to be able to customize which Tag the Check Box will enable/disable when it is clicked; in the **Tag** field, type **#Option1Tag**: as shown.



Defining a Custom Property for the Tag Field

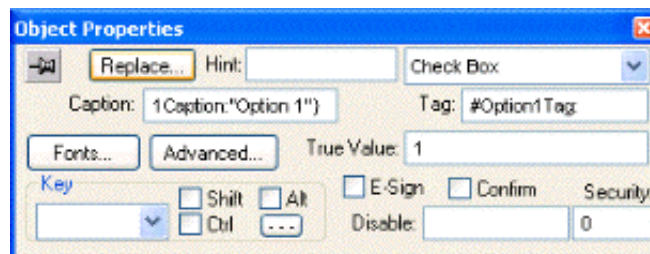
When you go to complete the properties on an instance of the Symbol, the **Option1Tag** property will appear like this:



Custom Properties on a Symbol

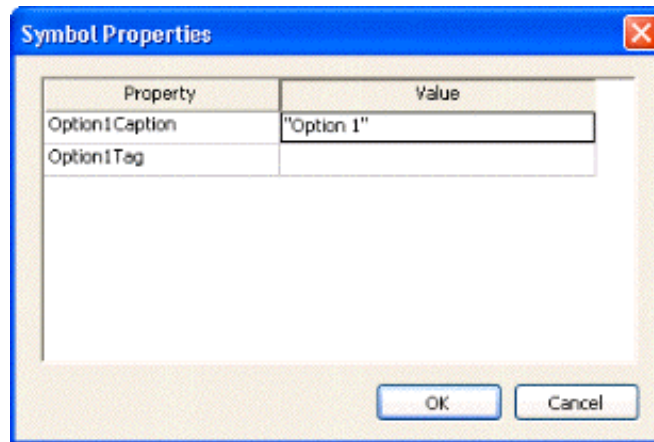
But more about that later...

4. Depending on the context, some Object properties require a specific type of value like a string, boolean or numerical value. For these properties, you must enclose the Custom Property definition in curly brackets "{ }". In this example, the **Caption** field requires a string, so type **{#Option1Caption:"Option 1"}** as shown.



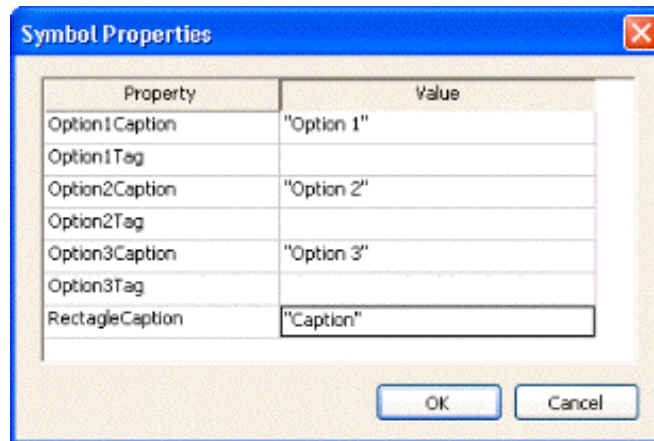
Defining a Custom Property for the Caption Field

Again, when you go to complete the properties on an instance of the Symbol, the **Option1Caption** property 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

- Save the Symbol and close the Symbol Editor.
- From the main menu bar, choose **Tools > Verify Application**. This will update all existing instances of the Symbol in your application project.

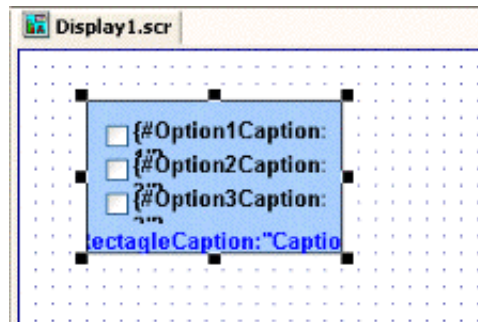
INSERTING A SYMBOL IN A SCREEN

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

- Open the desired Screen file (.scr) from the Screens folder, or insert a new Screen. The Screen file is opened for editing.
- Insert the desired Symbol in the Screen — there are several ways to do this:
 - Double-click on the Symbol file (.sym) in the Symbols folder;
 - Right-click on the Symbol file and choose **Insert** from the contextual menu;
 - Right-click in the Screen Editor where you want to insert the Symbol, choose **Insert Linked Symbol** from the contextual menu, and select the Symbol file using the standard *Open File* dialog; or

- Select the Symbol from the Library.

Regardless of how it is inserted, the Symbol is placed in the Screen as shown.



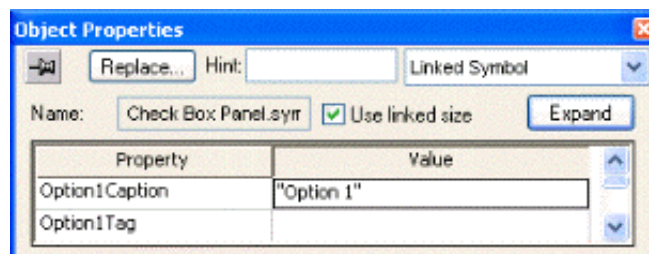
Symbol Placed in a Screen

Note:

You can also insert Symbols using the menu option **Edit > Paste From**. However, when using this menu option, the inserted Symbol does not keep any link with the Master Symbol — it is just a simple copy of the Object (or Group of Objects) used to create the Master Symbol.

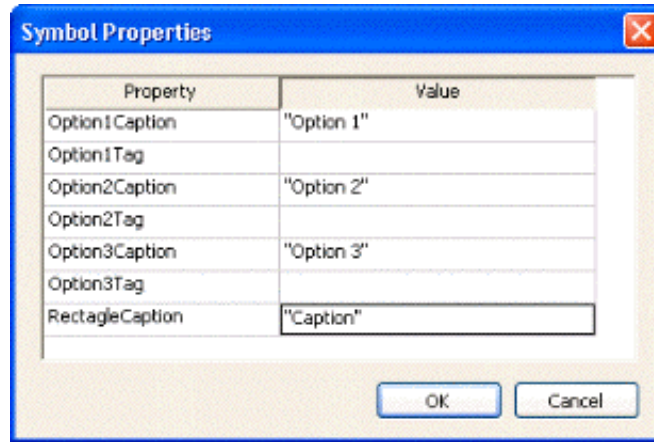
Once the Symbol is inserted, you can manipulate it like any other Object in the Screen. You can align and distribute it with other Objects, and you can apply Dynamic Properties to it. However, the first thing to do is complete the Custom Properties for this instance of the Symbol.

3. Open the Object Properties for the Symbol.



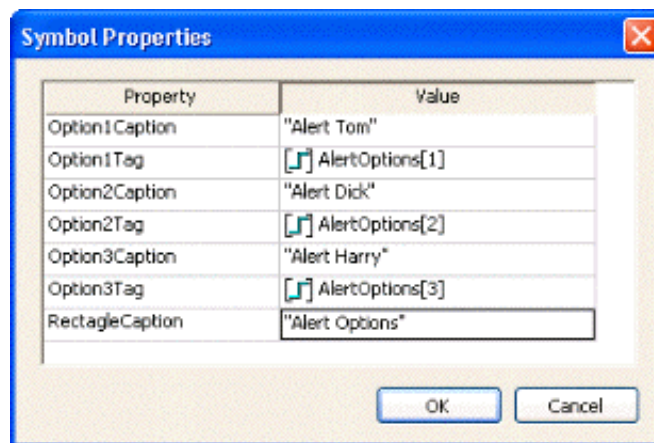
Object Properties Dialog for the Symbol

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



Symbol Properties Dialog for the Symbol

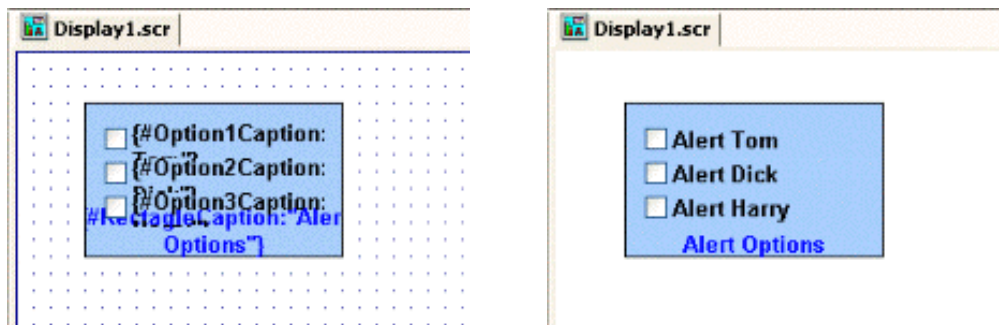
- Enter the property values as needed. In this example, the three Check Boxes are used to determine whether to alert Tom, Dick and/or Harry. The captions are updated accordingly, and the Check Box tags are configured with the first three indices of a boolean array called **AlertOptions**.



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.



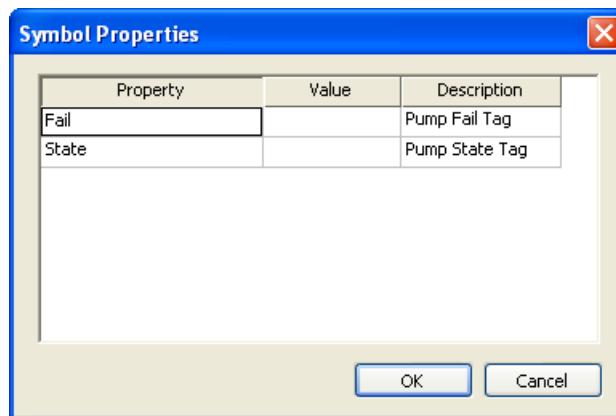
Symbol During Editing (left) and Runtime (right)

Note:

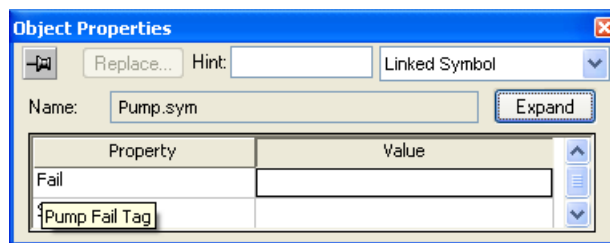
Remember, the completed Custom Properties on each instance of a Symbol are independent from every other instance of that Symbol, but if you make any changes to the Master Symbol file, then those changes automatically propagate to every instance.

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 contextual 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:

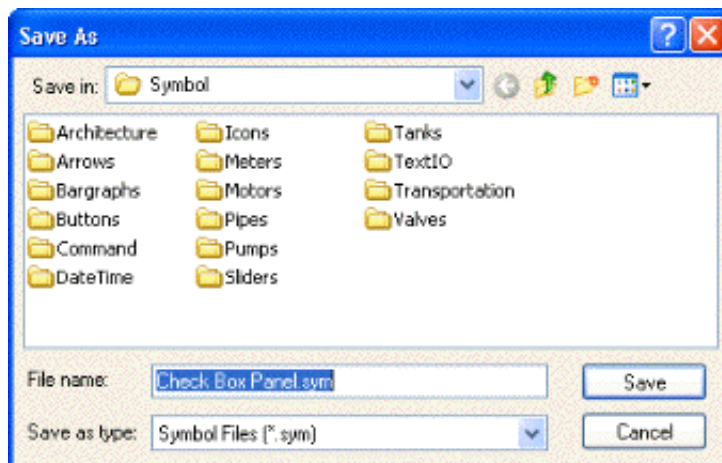


Tooltip Showing Description of the Property

MAKING A USER-MADE SYMBOL AVAILABLE TO OTHER APPLICATION PROJECTS

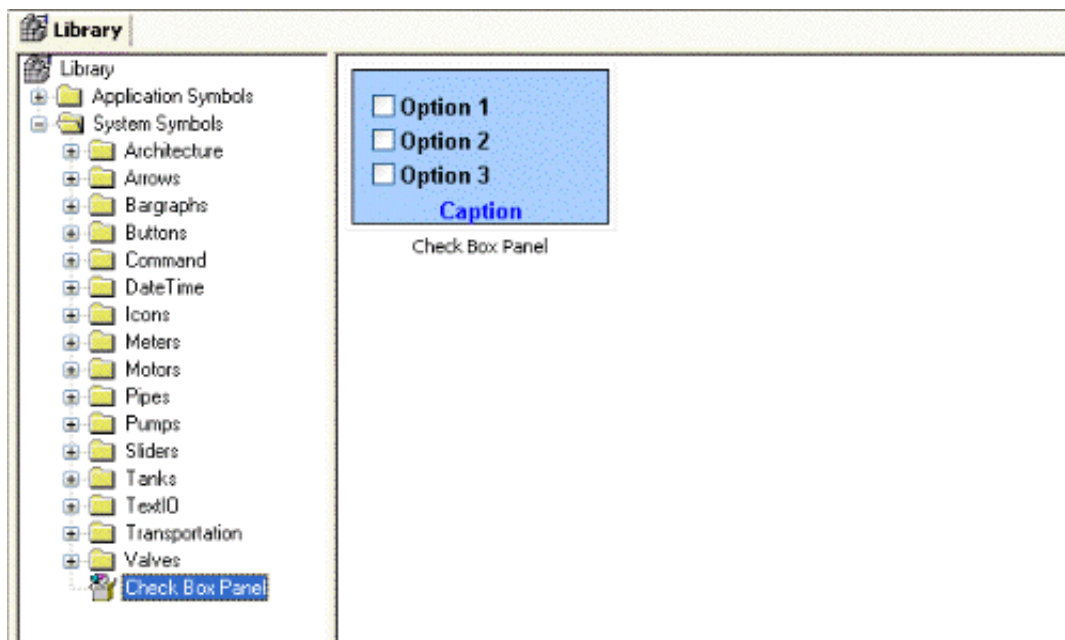
User-made Symbols are normally available only in the application project where they were initially created and saved. However, you can send a user-made Symbol to the System Symbols directory of the Library, to make it available to all application projects:

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



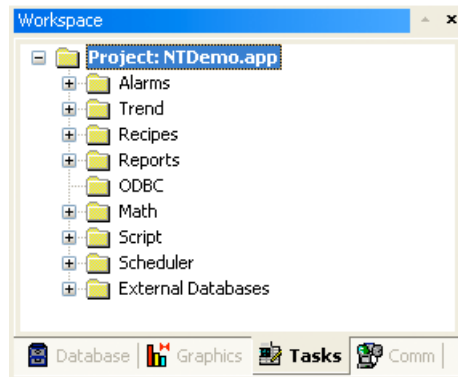
Symbol in the Library

Chapter 8: Configuring Task Worksheets

This chapter provides information on configuring *Task* worksheets. A *task* controls how different tags in the database will be used in relation to your application.

Before continuing our application development, we must create a script to simulate certain variables. Because these variables must be updated constantly, we will create this simulation in a *Math* worksheet. In a real world application, these variables would be coming from field equipment such as a PLC or a Soft Control.

Use the **Tasks** tab to access all task worksheets in the current application.



Workspace: Tasks Tab

This tab contains the following task folders:

- **Alarms:** Contains the *Alarm* worksheets used to configure alarm groups and the tags related to each alarm group in the application. You also use the **Alarm** task to define the alarm messages generated by IWS.
- **Trend:** Contains the *Trend* worksheets used to configure history groups that store trend curves for the application. 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. IWS 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 application 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. **Report** 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 IWS and any database supporting the ODBC interface.
- **Math:** Contains the *Math* worksheets used to configure and implement additional routines to work with different IWS tasks. IWS 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 application.

- **Scheduler:** Contains the *Scheduler* worksheets used to configure events using defined mathematical expressions, which are executed according to time, date, or other monitored event.

To open *Task* worksheets for editing, double-click the task button and the worksheet will display in the *Screen/Display* window.

Configuring an Alarms Task

The Alarms folder enables you to configure alarm groups and tags related to each group. The Alarm task defines the alarm messages generated by IWS. 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).

To insert an Alarm Worksheet, right-click on the Alarms folder and select the option **Insert** from the pop-up menu. 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.

The Alarm task is executed by the **Background Task** runtime task (BGTask). The Alarm task handles the status of all alarms and save the alarm messages to the history, if configured to do so. However, the Alarm task does not display the alarm messages to the operator. The Alarm/Events control object, available on the Active Objects toolbar from the screen editor, must be created and configured in a screen in order to display the alarms to the operator.

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).
- **Body:** Configure alarm messages and associate them to conditions linked to tags.

	Tag Name	Type	Limit	Message	Priority	Selection
*		HIHI				
*		HIHI				
*		HIHI				
*		HIHI				
*		HIHI				

Alarm Worksheet

Notes:

- You can configure the Alarm Group to sent notifications by email automatically, based on alarm events. For further details, see *Email Settings* below.
- The alarm properties associated to each tag (configured in the body of the alarm group) can also be edited by the Tag Properties dialog, launched by the Tag Properties toolbar. 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 version 6.1 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 application (.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).

HEADER SETTINGS

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 the 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.	Check-box
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.	Check-box

Field	Remarks	Syntax
On Line > Beep	When checked, the computer keeps beeping while there are alarm(s) to be acknowledged, currently active.	Check-box
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 <ApplicationName>.APP file: [AlarmLog] Device=<PrinterPath/PrinterName>	Check-box
History > Save to Disk	When checked, the alarm messages are stored in the history log when they become active.	Check-box
History > Generate Ack Messages	When checked, the alarm messages are stored in the history log when they are acknowledged.	Check-box
History > Generate Norm Messages	When checked, the alarm messages are stored in the history log when they become normalized.	Check-box
Colors in Alarm Controls > Enable	When checked, the alarms configured in this group will be displayed with the colors assigned to each alarm state (Start, Ack or Norm), according to the colors configured in the Alarm Group.	Color-box
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: - Start: Alarm active and not acknowledged - Ack: Alarm active and acknowledged - Norm: Alarm no longer active and not acknowledged.	Color-box

BODY SETTINGS

Field	Remarks	Syntax
Tag Name	Name of the tag associated with the alarm.	Tag

Field	Remarks	Syntax
Type	<p>Type of the alarm:</p> <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. ▪ Lo: Activates the alarm if the tag value is equal or lower than the limit. ▪ 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. ▪ 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. <p>When using the types Rate, DevP, and DevM it is necessary to configure additional settings by the Tag Properties dialog, launched by the Tag Properties toolbar.</p>	Combo-box
Limit	Limit associated with each alarm. The limits can be modified dynamically during the 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 chars)

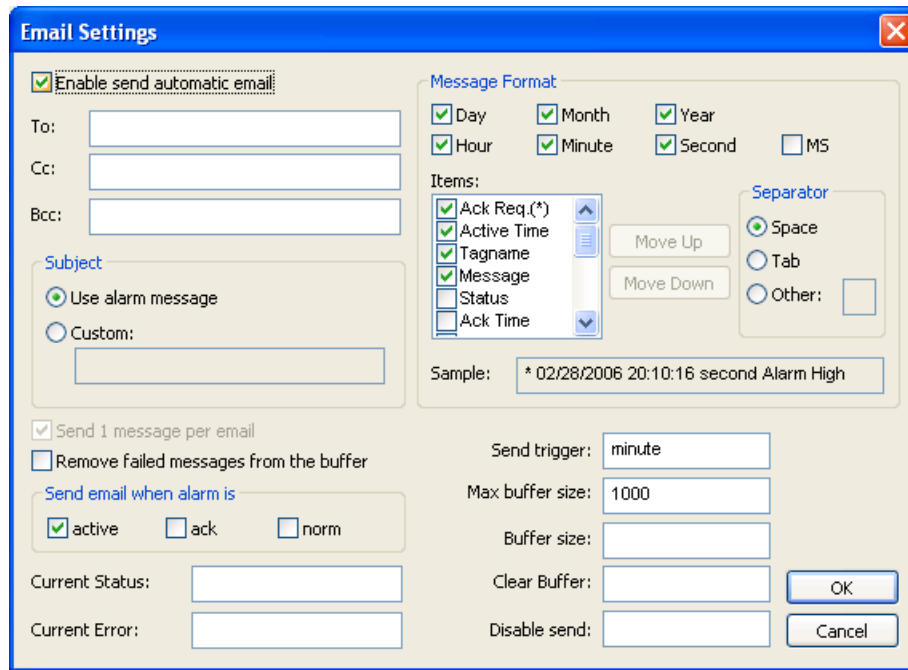
EMAIL SETTINGS

InduSoft Web Studio 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.

Caution:

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 application

(e.g.: SMTP server, user name, password, and so forth).



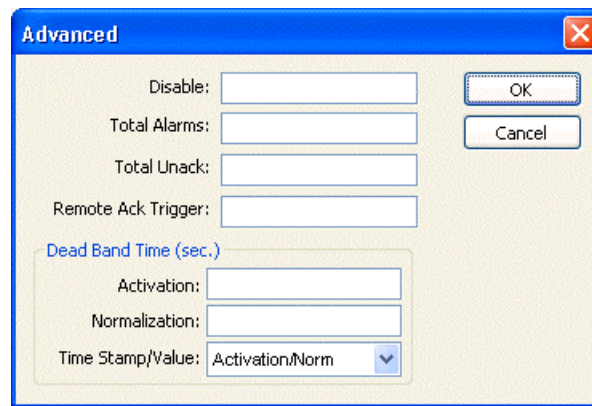
Alarm Worksheet: Email Settings

Field	Remarks	Syntax
Enable send automatic email	Name of the tag associated with the alarm.	Check-box
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 and Text (up to 1024 characters)
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”.	Check-box

Field	Remarks	Syntax
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.	Check-box
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.	Check-box
Current Status	The tag configured in this field, if any, is updated with the current status of the current or last email that the application 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) ▪ 2: Last email was sent successfully. ▪ 3: There was an error sending the last email. 	Tag
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 application attempted to send. Therefore, when configuring a tag in this field, this tag must be a String type.	Tag
Message Format	This interface allows you to configure the actual format of the message sent by email, based on the alarm event(s): <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. While you configure these settings, the Sample field displays an example of the format of the message according to the settings being configured.	Check-box 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

Field	Remarks	Syntax
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



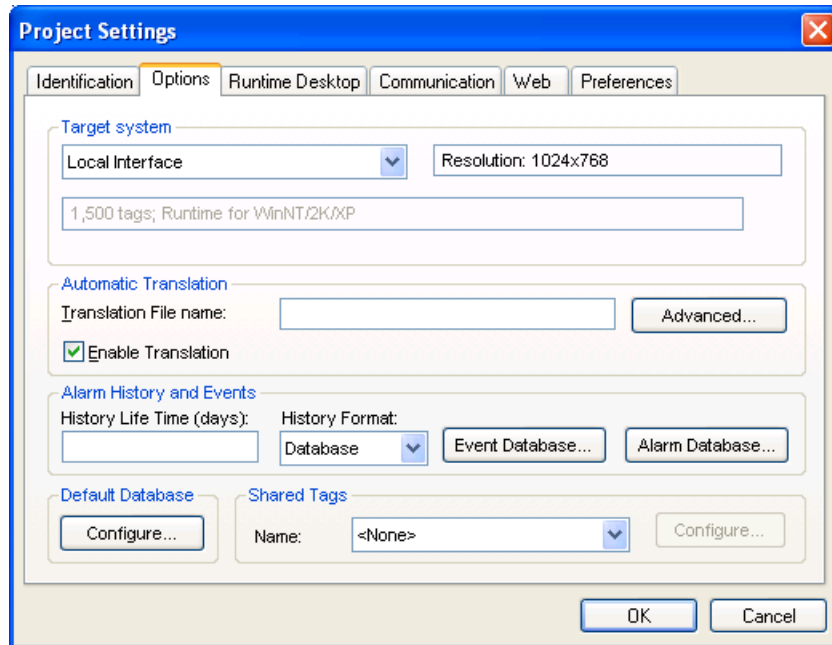
Alarm Worksheet: Advanced

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 becomes 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 HISTORY

The alarm history can be saved either with the InduSoft Web Studio proprietary format or to an external SQL Relational database by ADO. The alarm history settings can be configured by the **Project Settings > Options** dialog.



Project Settings: Options Tab

The Alarm History and Events interface allows you to configure the following settings:

Field	Remarks	Syntax
History Life Time (days)	The history for alarm/event messages older than the number of days specified in this field are automatically deleted, following a FIFO (First-In, First-Out) behavior. If this field is left in blank, the history of alarms/events is not deleted automatically.	Number

Field	Remarks	Syntax
History Format	<p>Defines the format of the history of alarms/events:</p> <ul style="list-style-type: none"> ▪ Proprietary: Saves the history in the proprietary format. The alarm messages are saved in a text file with the name ALxxyyzz.ALH, where: <ul style="list-style-type: none"> ○ xx: Last two digits of the current year. ○ yy: month. ○ dd: day <p>There will be one history file for each day. By default, the alarm history files created with the proprietary format are stored in the \Alarm sub-folder of the application. However, it is possible to direct the alarms to a different directory by using the <code>SetAlarmPath()</code> function from the built-in language.</p> <ul style="list-style-type: none"> ▪ Database: Saves the history in a third-party SQL Relational Database (e.g.: SQL Server). InduSoft Web Studio does not provide the database itself. 	Combo-box
Alarm Database	<p>When selecting Database as the format for the history of alarms, the specific settings to interface with the third-party SQL Relational Database can be configured by the dialog launched when pressing this button. For further details about support for third-party SQL Relational Databases, see <i>Chapter 17: IWS Database Interface</i>.</p>	-

When saving the alarm history in the proprietary format, each alarm event is saved in a new line, using the pipe character (“|”) to delimiter the different 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
P1|P2|P3|P4|P5|P6|P7|P8|P9|P10|P11|P12|P13|P14|P15|P16|P17|P18|P19|P20|P21
.
.
P1|P2|P3|P4|P5|P6|P7|P8|P9|P10|P11|P12|P13|P14|P15|P16|P17|P18|P19|P20|P21
```

The format of the history both in proprietary format and in the SQL Relational Database format is described in the following table:

Proprietary Format	SQL Relational Database		Remarks	File Version
	Field Number	Field Name		
P1	-	-	File version (Current = 003)	001
P2	Al_Start_Time	Date/Time	Start Date (DD/MM/YYYY)	001
P3			Start Time (HH:MM:SS)	001
P4	Al_Tag	Text	Tag Name	001

Proprietary Format	SQL Relational Database		Remarks	File Version
	Field Number	Field Name		
P5	AI_Message	Text	Alarm Message	001
P6	AI_Ack	Number	Ack, where: <ul style="list-style-type: none"> ▪ 0 Indicates the alarm was acknowledged or does not require acknowledgment ▪ 1 Indicates the alarm was not acknowledged 	001
P7	AI_Active	Number	Active, where: <ul style="list-style-type: none"> ▪ 0 Indicates the alarm is not active ▪ 1 Indicates the alarm is active 	001
P8	AI_Tag_Value	Number	Tag Value when the event occurred	001
P9	AI_Group	Number	Alarm Group Number	001
P10	AI_Priority	Number	Priority Number	001
P11	AI_Selection	Text	Selection	001
P12	AI_Type	Number	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) ▪ 32 is DevP ▪ 64 is DevM 	001
P13	AI_Ack_Req	Number	Ack required, where: <ul style="list-style-type: none"> ▪ 0 Alarm requires acknowledgement (Ack) ▪ 1 Alarm does not require acknowledgement 	001
P14	AI_Norm_Time	Date/Time	Normalization Date (DD/MM/YYYY)	001
P15			Normalization Time (HH:MM:SS)	001
P16	AI_Ack_Time	Date/Time	Ack Date (DD/MM/YYYY)	001
P17			Ack Time (HH:MM:SS)	001
P18	AI_User		User Name	002
P19	AI_User_Comment		Comment	002
P20	AI_User_Full		User Full Name	003
P21	AI_Station		Station	003

Proprietary Format	SQL Relational Database		Remarks	File Version
	Field Number	Field Name		
P22	AI_Prev_Tag_Value	Number	Previous Value	003
P23	Bias	Number	Time Zone Bias	003
-	AI_Start_Time_ms	Number	Number of milliseconds for the Start Time timestamp. This field is used when the database does not support ms in a TimeStamp field.	003
-	AI_Norm_Time_ms	Number	Number of milliseconds for the Norm Time timestamp. This field is used when the database does not support ms in a TimeStamp field.	003
-	AI_Ack_Time_ms	Number	Number of milliseconds for the Ack Time timestamp. This field is used when the database does not support ms in a TimeStamp field.	003
-	AI_Deleted	Number	<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	Date/Time	Timestamp of the last update for this event.	003
-	Last_Update_ms	Number	Number of milliseconds for the Last Event timestamp. This field is used when the database does not support ms in a TimeStamp field.	003

For detailed information about saving alarm history files, including instructions for configuring database settings, see *Chapter 17: IWS Database Interface*.

Configuring a Trend Task

The *Trend* folder enables you to configure history groups that store trend curves. 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. IWS stores the samples in a binary history file (*.hst), and shows both history and on-line samples in a screen trend graph.

To show a trend graph on the screen, you must click the **Trend** tool on the *Active Objects* toolbar to create a trend object.

Use one of the following methods to insert a new *Trend* worksheet:

- Right-click on the *Trend* folder and select **Insert** from the pop-up.
- Select **File** → **New** from the menu bar or click on the **New** tool on the *Standard* toolbar to display the *New Document* dialog. Click **Trend Worksheet**, and then click **OK**.

A new worksheet displays, as follows:

	Tag Name	Dead Band	Column
1			
2			
3			
4			
5			

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 type from the list. The following options are available:
 - **Proprietary**
 - * **File Format:** Binary
 - * **Default Path:** ...\\Hst\GGYYDDMM.HST , where:

 - YY = Two last digits of the year
 - MM = Month
 - DD = Day

Note: IWS provides the HST2TXT.EXE and TXT2HIST.EXE programs, which enable you to convert trend history files saved on proprietary format (.hst) to text files (.txt) and vice versa. For more information about these programs, see “Converting Trend History Files from Binary to Text” on page 8–21 and “Converting Trend History Files from Text to Binary,” on page 8–22.

– **Database**

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

Note: For more information about the structure of the Database table that IWS uses to save history files, see Data Saved in Trend History File.

- **Database Configuration:** Opens the Database Configuration dialog to enter the requisite settings to link IWS to an external SQL Relational Database, for the purpose of saving the trend history.

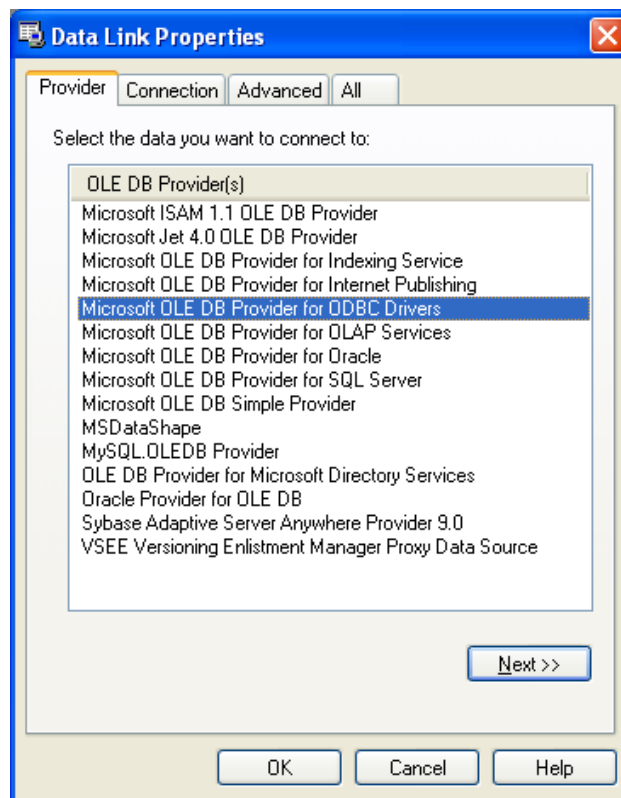
Database Configuration Window

- **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.

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).

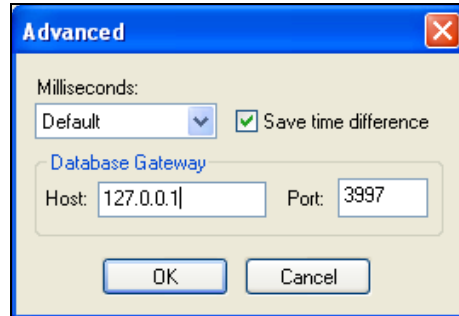
- **Use application default** check-box: When this option is checked, IWS uses the settings configured in the Default Database for the task that is being configured (Connection string, User name, Password, Retry Interval and Advanced Settings). When this option is not checked, you can configure these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

**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 applications, the default value of these settings do not need to be modified and should be kept.



- * **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 by InduSoft, the **Default** option attempts to save the milliseconds in a separate field.

⇒ **Tip:**

The default option for each database is configured in the StudioADO.ini file, stored in the \BIN sub-folder of IWS. See the *Studio Database Gateway* section for information about how to configure the StudioADO.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** check-box: When this option is checked (default), IWS saves the Time Zone configured in the computer where the application 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 Studio 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 Studio database gateway with the same TCP Port. See “Studio Database Gateway” for information about how to configure the Studio ADO Gateway.
 - * **Disable Primary Key**: 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), you should check this field.
- **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. InduSoft 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** check-box: When this option is checked (default), IWS saves and/or retrieve the data in the Table with the default name written in the **Name** field.
 - **Automatically create** check-box: 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.
 - **Refresh** button: If the database configured is currently available, you can press the **Refresh** button to populate the **Name** combo-box with the name of the tables currently available in the database. In this way, you can select the table where the history data should be saved instead of writing the Table name manually in the **Name** field.
- **Run-Time** pane: This area allows you to enter with IWS tags. The following fields are available:
 - **Status** (output) check-box: The tag in this field will receive one of the following values:

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

- **Reload** (output): If you are using in curly brackets in any of the configuration fields, you have to specify the reload tag. 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 and will set the tag back to 0 when it is finished.

For instructions for configuring a default database for all task history (Alarm, Event and Trend), see *Configuring a Default Database for All Task History* on page 6–18.

For more information about saving Trend history, including an explanation of the fields saved, see *Chapter 17: IWS Database Interface*.

- **Save Options** pane: Use the following parameters to define when and how to save trend history files:
 - * **Save On Trigger** check-box 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** check-box: Click (enable) to *always* save the trend sample when a value change occurs in any of the tags from that group.
- **Advanced** button: Click to display the following window:

Trend Advanced Settings

- **Batch** pane: The Batch pane provides the following fields:
 - * **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 <Application>.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 either in Graphical format or Table format. The *Trend* object and *Grid* object sections explain how to display information on these formats.

- **Disk Space Control** pane: The following options are available:
 - * **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 WinCE applications.
 - * **Disable All Data Saving:** Enter with a tag in this field. When the value on this tag is **TRUE** (different from zero), the trend task stops recording data for this worksheet.

Use the **Body** parameters on the *Trend* worksheet as follows:

- **Tag Name** field: Type the tag name to be saved in the history file.

⇒ **Caution:**

After adding or removing tags from a *Trend* worksheet, any history files (*.HST) you previously created will no longer be compatible with the new setting. Consequently, the data from those history files will no longer be displayed by the trend object.

- **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 start the finished application.

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:

- ✓ From a DOS window, change directory (**cd**) to the IWS **Bin** directory.

```
c:\>cd \Program Files\InduSoft Web Studio v6.1\Bin
```
- ✓ 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 plain text **.hdr** (header) file and **.txt** file 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** function in a *Math* worksheet to convert binary files into text format automatically without having to use a DOS window.

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.

```
c:\>cd \Program Files\InduSoft Web Studio v6.1\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.

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 the “Configuring a Trend Task” in *Chapter 8: Configuring Task Worksheets*, 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 <ApplicationName>.APP file, as follows:

```
[Trend]
<DefaultName>=<NewName>

[Trend<Group><PRI | SEC>]
BatchHistory=<TableName>
```

For example:

```
[TREND001PRI]
BatchHistory=MyTableForPrimaryDB
[TREND001SEC]
BatchHistory=MyTableForSecondaryDB
[Trend]
Group_Number=Trend_Worksheet
Batch_Name=Load_Number
```

Configuring a Recipes Task

The *Recipe* folder enables you to configure *Recipe* worksheets for data interchange between the application database and disk files in ASCII, XML, or DBF format; transferring values between files and real-time memory.

You typically use a *Recipe* worksheet used to store process recipes, but you can store any type of information (such as operation logs, passwords, and so forth) in these files. The *Recipes* task reads and writes tag values into files, and it transfers tag values from the application to a file or from a file to the application.

Note:

IWS sequentially increments the number that identifies the *Recipe* worksheet for each newly created worksheet.

Use one of the following methods to create a new *Recipe* worksheet:

- Right-click on the *Recipes* folder, and then click the prompt screen.
- Select **File > New** from the menu bar or click on the **New** button on the *Standard* toolbar to open the New *Document* dialog. Select **Recipe Worksheet** and click **OK**.

A new *Recipe* worksheet displays:

	Tag Name	Number of Elements	
1			
2			
3			
4			
5			

Recipe Worksheet

The *Recipe* 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.
- **Save As XML**: Click (*check*) to save information in XML format, or (*uncheck*) to save in the **.DAT** format.

Caution:

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 application running to view the XML data. (You must be running Internet Explorer version 5.0 or higher to view XML data.)

- **File Name** field: Type a file name related to the recipe group, using static text (**File1**) or a dynamic tag value (**{FileNameTag}**).
- **Register Number** field: Type a tag to define the register number to be read or written into a DBF file. IWS enables this field for older applications created to use DBF files, but disables this field for newer projects.
- **Unicode** check-box: 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 IWS automatically enables every array position and class member by default. To configure a specific array position and/or class member, type it in the **Tag Name** field as normal. For example, **level[3].member**.
- **Number of Elements** field: Specify how many positions of the array tag are in use.

Tip:

You can configure a tag name between curly brackets {TagName} in this field, allowing the user to change the Number of Elements configured in the Recipe for each array tag dynamically, during the runtime.

Caution:

When you define an array tag, its initial position is zero, although IWS uses the tag in case of invalid position configuration.

To read or write a recipe group, use the InduSoft Scripting Language **Recipe** function.

Configuring a Reports Task

The *Reports* folder contains a definition of reports (text type) to be sent to a printer or disk. The **Reports** task allows you to configure your own report (text type) with data from the system. The main purpose of this task is to make report creation easier and more efficient.

 **Note:**

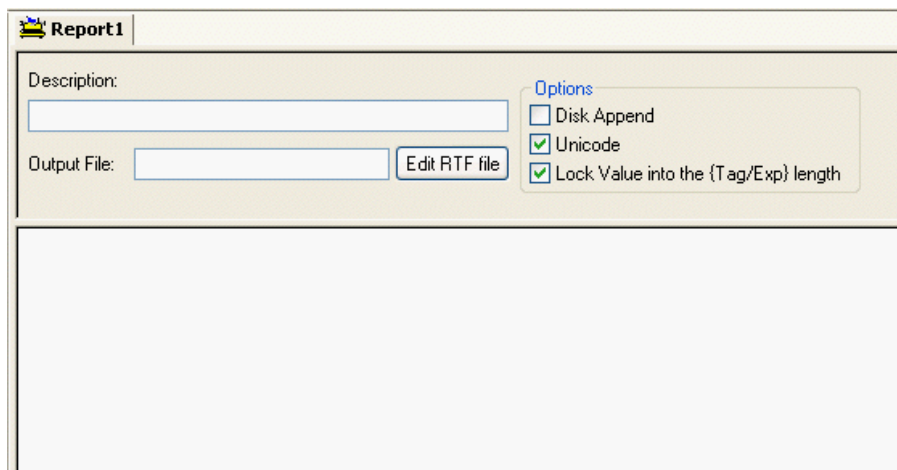
IWS sequentially increments the number that identifies the *Report* worksheet for each newly created worksheet.

To print a report, use an InduSoft Scripting Language function anywhere an expression is allowed.

Use one of the following methods to insert a new *Report* worksheet:

- Right-click on the *Reports* folder and click on the prompt screen.
- Select **File** → **New** from the menu bar or click the **New** icon on the *Standard* toolbar to open the *New Document* dialog. Select **Report Worksheet** and click **OK**.

A new worksheet displays:



The screenshot shows a dialog box titled "Report1". It contains a "Description:" label followed by a text input field. Below this is an "Output File:" label followed by another text input field and a button labeled "Edit RTF file". To the right of the "Output File:" field is an "Options" section with three checkboxes: "Disk Append" (unchecked), "Unicode" (checked), and "Lock Value into the {Tag/Exp} length" (checked).

Report Worksheet

The *Report* worksheet is divided into two areas:

- *Header* area (top section), which contains information for the whole group.
- *Body* area (bottom section), where you define the tag and text to be used in generating the report.

Use the **Header** parameters on this worksheet as follows:

- **Description** field: Type a description of the worksheet for documentation purposes.
- **Disk Append** check-box: When printing to a file,
 - Check the box to add (*append*) 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** check-box: Click (*enable*) to save the report in UNICODE format (two bytes per character) or (*disable*) to save the report in ANSI format (one byte per character).

- **Lock Value into the (Tag/Expression) length** check-box: 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.
- **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** ..., 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.

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 characters 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.

Configuring an ODBC Task

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.

In addition to configuring the ODBC worksheet, you must configure the Windows ODBC standard driver. IWS refers to the User DNS (Data Source Name), which you configure through the *Control Panel*. For more information, refer to your Windows documentation.

Note:

IWS sequentially increments the number that identifies the *ODBC* worksheet for each newly created worksheet.

Use one of the following methods to insert a new ODBC worksheet:

- Right-click on the *ODBC* folder and click on the prompt screen.
- Select **File** → **New** from the menu bar or click the **New** icon on the *Standard* toolbar to open the *New Document* dialog. Select **ODBC Worksheet** and click **OK**.

A new worksheet displays:

	Tag Name	Column
1		
2		
3		
4		
5		

ODBC Worksheet

ODBC worksheets are executed under the *ODBC Client Runtime* task. However, creating a new worksheet does not automatically enable the task; you must use the *Execution Tasks* dialog (**Project** → **Status**) to configure the task to start at runtime. For more information, please see “Starting Runtime Modules on the Target System” on page 6–31.

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.
- *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** field: Type the same Data Source Name (DSN) specified in the Windows *Control Panel* containing information about specific database access. You can change the **Data Source** name during runtime by configuring a tag between curly brackets in this field. For example:
{DSNNameTag}
- **User** field: Type a user name to access to the database. You can change the **User** name during runtime by configuring a tag between curly brackets in this field.
- **Password** field: Type the user's password. You can change the **Password** during runtime by configuring a tag between curly brackets in this field.
- **Table** field: Type a table name in the database. You can change the **Table** name during runtime by configuring a tag between curly brackets in this field.
- **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** 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, in an Excel file, if you type in Column 1, Row 1, the entire column gets defined as column 1).

You must use the Windows *Control Panel* to set up the ODBC interface for Excel files. The procedure is as follows:

- ☑ Click the **Start > Settings > Control Panel**.
- ☑ When the *Control Panel* window displays, double-click on the **ODBC** button to open the *ODBC Data Source Administrator* dialog.
- ☑ In the *ODBC Data Source Administrator* dialog, click **Excel Files** in the *User Data Sources* list, and then click the **Configure** button.
- ☑ 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.
- ☑ Click the **Select Workbook** button to configure the Excel file you want to use.
- ☑ Return to the *ODBC Data Source Administrator* dialog and verify that your User DSN displays in the list. Click **OK** to close the dialog.
- ☑ After configuring the ODBC Windows interface, you must configure the IWS *ODBC* worksheets.
- ☑ From the **Tasks** tab, insert a new *ODBC* worksheet.

- ☑ Be sure you set the **ODBC Runtime** to start automatically from the **Execution Tasks** tab (**Project** → **Project Status**).

To start this configuration, you simply need to run the project. Your application will handle the **Select**, **Next**, **Insert**, **Delete**, and **Update** triggers to allow data to exchange throughout rows in Excel and tags configured in the worksheet.

Consult your Windows documentation for the meaning of specific error codes.

The following is a list of IWS error codes:

Select command

- 1 - Error in the ODBCPREPARE function.
- 2 - Error in the ODBCBINDCOL function.
- 3 - Error in the ODBCEXECUTE function.
- 4 - Error in the ODBCSETCH function.

Next command

- 5 - Error in the ODBCSETCH function.

Insert command

- 6 - Error in the ODBCPREPARE function.
- 7 - Error in the ODBCEXECUTE function.
- 8 - Error in the ODBCCOMMITE function.

Update command

- 9 - Error in the ODBCPREPARE function.
- 10 - Error in the ODBCEXECUTE function.
- 11 - Error in the ODBCCOMMITE function.

Delete command

- 12 - Error in the ODBCPREPARE function.
- 13 - Error in the ODBCEXECUTE function.
- 14 - Error in the ODBCCOMMITE function.

Configuring a Math Task

The *Math* folder allows you to implement additional routines to work with the basic functions of different IWS tasks. A *Math* worksheet contains a group of programming lines that IWS executes as a *Background Task* during runtime. You can configure the *Math* worksheet to provide free environments for logical routines and mathematical calculations needed by the project. For these purposes, the InduSoft Scripting Language is very simple and easy to use.

Note:

IWS sequentially increments the number that identifies the *Math* worksheet for each newly created worksheet.

Use one of the following methods to insert a new *Math* worksheet:

- Right-click on the *Math* folder and then click on the prompt screen.
- Select **File** → **New** from the menu bar or click the **New** button on the *Standard* toolbar to open the *New Document* dialog. Select **Math Worksheet** and click **OK**.

A new worksheet displays:

	Tag Name	Expression
1		
2		
3		
4		
5		

Math Worksheet

The *Math* 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 *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.

⚠ Caution:

IWS 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 send the return value to the specified tag in the **Tag Name** column.

Configuring a Scheduler Task

The *Scheduler* folder generates events with defined mathematical expressions to be executed according to the time, date, or any monitored event.

Note:

IWS sequentially increments the number that identifies the *Scheduler* worksheet for each newly created worksheet. Different scheduler groups have only organizational purposes.

Use one of the following methods to insert a new *Scheduler* worksheet:

- Right-click on *the Scheduler* folder and click on the prompt screen.
- Select **File** → **New** from the menu bar or click the **New** icon on the *Standard* toolbar to open the *New Document* dialog. Select **Scheduler Worksheet** and click **OK**.

A new worksheet displays:

	Event	Trigger	Time	Date	Tag	Expression
1	▼					
2	▼					
3	▼					
4	▼					
5	▼					

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 **Header** 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 day 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.
- **Trigger** field: Type a tag that triggers a change event when the value of this tag changes. When the **Trigger** tag changes, IWS returns the value specified in the **Value** field to the tag. This field is used only by the change event.
- **Time** field: Specify a time interval in which an event must occur when used by clock – hours (0 to 23), minutes (0 to 59), and seconds (0 to 59). You also can use this field to specify a specific time to be used by calendar events.
- **Date** field: Specify a specific date on which a calendar event must occur – day (1 to 31), month (1 to 12), and year (1900 to 2099). If you leave the field blank, the event occurs daily. This field is used only by the calendar event.
- **Tag** field: Type a tag to receive the value returned by the **Expression** field.
- **Expression** field: Type an expression whose return value will be sent to the tag. 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 (logically true) to execute the function. Use an expression value equal to one and the function will not execute (Disable \geq 1).

Configuring an External Databases Task

In addition to ODBC, IWS also supports Microsoft .NET ActiveX Data Objects (ADO.NET) for interfacing between the Application Tags database and other external databases.

Note:

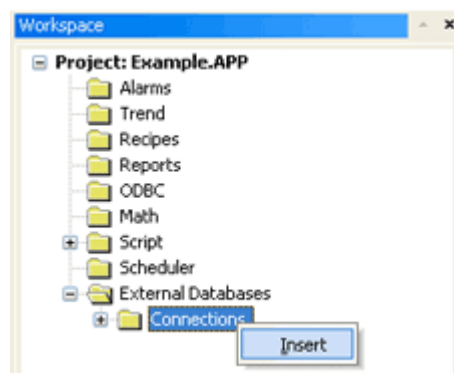
For more information about ADO.NET support in IWS — including how to communicate with remote databases using the Studio Database Gateway software — please see “IWS Database Interface” on page 17-1.

To interface with an external database, you must first configure a connection to the database and then build a worksheet that associates application tags with the database fields.

Database Connections

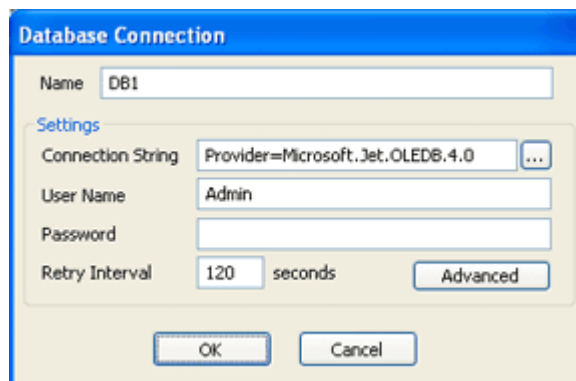
To create a new connection to a target database:

1. In the *Workspace*, open the *External Databases* folder and then right-click on **Connections**.
2. Choose Insert from the pop-up menu.



Inserting a New Database Connection

The Database Connection dialog is displayed.



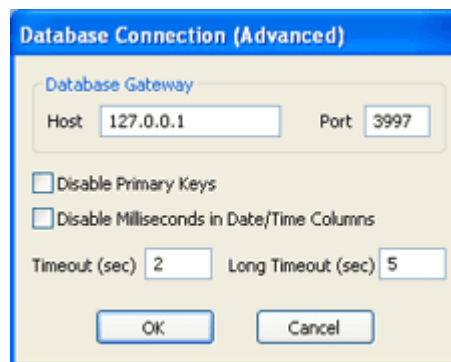
Database Connection dialog

- 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.
- 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 IWS. For more information about using the *Data Link Properties* dialog, please refer to Windows Help.

- 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.
- 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

- In the **Host** field, enter the IP address of the station that is running the Studio 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.
- Click **OK** to close the dialog and save the connection configuration.

Database connections are saved as XML files in the `\<Application Name>\Config` directory. 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...

`\<Application 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:

- Right-click on the *External Databases* folder and choose **Insert** from the pop-up menu; or
- Choose **File** → **New** from the main menu bar (or click the **New** tool on the *Standard* toolbar) to open the *New Document* dialog, select **DB Worksheet**, and click **OK**.

A new worksheet is displayed, as shown below:

	Tag Name	Column
*		
*		
*		
*		
*		

Database worksheets are saved in the `\<Application 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...

`\<Application 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 (**Project** → **Status**) to configure the task to start at runtime. For more information, please see “Starting Runtime Modules on the Target System” on page 6–31.

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:

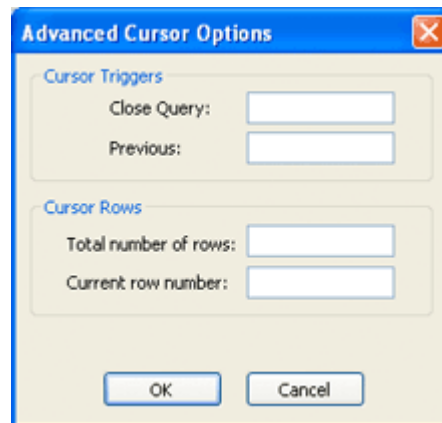
- **Description** field: Enter a description of the worksheet, for documentation purposes.
- **Status** field: Enter a numeric tag that will receive error codes for any database command errors generated during runtime.
- **Completed** field: Enter a numeric tag that will be toggled when database commands are successfully executed.
- **Error Message** field: Enter a tag of String type that will receive detailed error messages.
- **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 dataset 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:**

Table, **Condition** and **SQL Statement** can all be tags of String type, allowing you to change the selection during runtime. However, 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 dataset 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 dataset and copies those values to the tags configured in the worksheet body.

- **Advanced** button: Click to open the *Advanced Cursor Operations* dialog...



- **Close Query** field: Enter any tag; when the value of the tag changes, the cursor releases the selected dataset.
 - **Previous** field: Enter any tag; when the value of the tag changes, the cursor moves to the previous row of the dataset 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 dataset.
 - **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 dataset 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 selected dataset 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 selected dataset 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 application tags to the columns (fields) of the selected dataset. For each row of the body, enter a **Tag Name** and its corresponding **Column**. Which columns are available depends on how the dataset is selected, and how the dataset is selected may change during runtime, so be sure to map all necessary columns.

Chapter 9: Event Settings

This chapter describes InduSoft Web Studio's new 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. InduSoft Web Studio saves all of these events in a log file, which can be retrieved by the *Alarm/Event Control* object.

Event log files are stored in the application's **\Alarm** folder (the same folder where InduSoft Web Studio saves alarm history files). The event log file names must conform to the **evYYMMDD.evt** format, where:

- **YY** represents the last two digits of the year in which the event log file was generated
- **MM** represents the month in which the event log file was generated
- **DD** represents the day on which the event log file was generated

For example, a log file for May 7, 2003 would be **ev030507.evt**.

The event files (***.evt**) are ASCII text files created according to the following format:

```
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
```

P1 = File version (Current > 1)

P2 = Event Type

- . 1 > SECURITY SYSTEM
- . 2 > DISPLAY
- . 3 > RECIPE
- . 4 > REPORT
- . 5 > CUSTOM MESSAGES
- . 6 > SYSTEM WARNING
- . 7 > LOG TAGS

P3 = Event Time (DD/MM/YYYY HH:MM:SS.SSS)

P4 = Tag Name

P5 = Tag Value

P6 = Source (Not used)

P7 = User Name

P8 = User Full Name

P9 = Event Message

P10 = Station

P11 = Comment

P12 = Previous Value

P13 = Time Difference (Bias)

Configuring the Events Settings

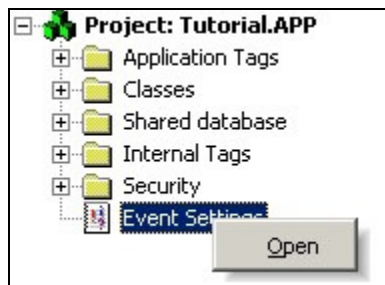
Use the steps to configure the event-retrieval feature:

- ☑ Select the **Database** tab. This tab contains a new icon called **Event Settings**:

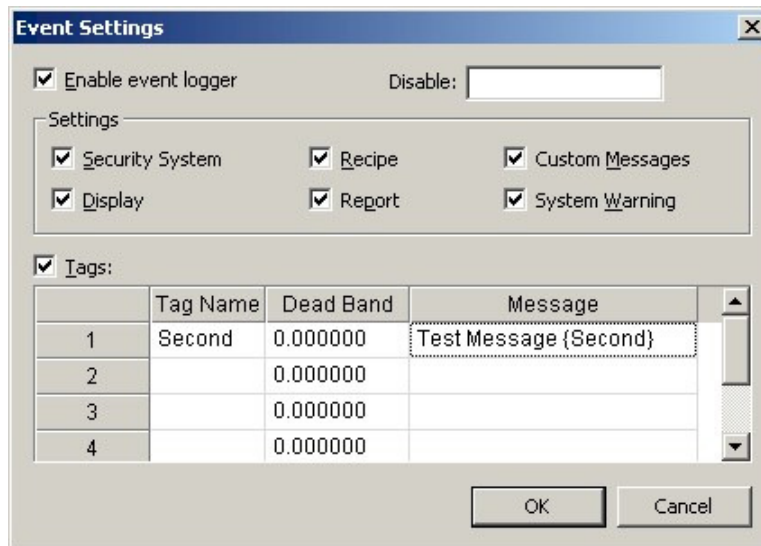


Database Tab: Event Settings

- ☑ Right-click the **Event Settings** icon and select **Open** from the pop-up to open the *Event Settings* dialog:



Selecting Event Settings



Event Settings Dialog

- ☑ Configure the parameters on the *Event Settings* dialog as follows:
 - **Enable event logger** check-box: Enable (*check*) this box to enable event-logging.
 - **Disable** text box: Type a tag into this field. If the tag value is other than 0 (*false*), InduSoft Web Studio automatically disables the *Event Logger*.
 - **Security System** check-box: Enable (*check*) this box to include security system events in the historic event file. IWS logs the following security system events:
 - * Log On / Log Off users
 - * User created/removed using the **CreateUser ()** or **RemoveUser ()** functions
 - * User blocked/unblocked using 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
 - **Display** check-box: Enable (*check*) this box to include screen Open and Close events in the historical event file.
 - **Recipe** check-box: Enable (*check*) this box to include recipe load, save, init, and delete events in the historical event file.
 - **Report** check-box: Enable (*check*) this box to include *reports saved to disk or send to printer* events in the historical event file.
 - **Custom Messages** check-box: Enable (*check*) this box to include events generated by the **SendEvent (strEvent)** function in the historical event file.
 - **System Warning** check-box: Enable (*check*) this box to include general system warnings (such as **Division by zero, Attempted to access invalid array index**, and so forth) in the historical event file. IWS logs the following system warning events:
 - * Errors that occur when sending alarms by email
 - * Tag was blocked/unblocked
 - * Division by zero
 - * Connection/Disconnection of the remote security system
 - **Tags** check-box: Enable (*check*) this box to enable and log tag changes in the historical event file. Configure the tags you want to log in the Tags table as follows:
 - * **Tag Name** column: Type the name of the tag you want to log in the event file.
 - * **Dead Band** column: Type a value for comparing and filtering acceptable changes.

For example, if you specify a **Dead Band** value = 5 for a tag value = 50 and the tag value changes to 52, the system will not register this variation in the event log file, because the variation is less than 5. However, if the tag value change is equal to or greater than 5, the system will save the new value to the history file.
 - * **Message** column: Type a string (message) related to this tag change. You can specify tags in messages using the **{tag name}** syntax.

The **Tags** parameter can be useful if you want to generate a log file of events that are not necessarily alarm conditions (for example, **Motor On, Motor Off**, and so forth).

Use the History Format combo-box to save events data in either the Proprietary history file format from IWS or to an external SQL Relational database. The options for both are as follows:

- **Proprietary**
 - **File Format:** Text (UNICODE). IWS uses the vertical bar character (|) to separate the fields.
 - **Default Path:** ...\ - YY = Two last digits of the year
 - MM = Month
 - DD = Day
- **Database**
 - **Database Type:** Chosen by the User
 - **Default Table Name:** EventHistory

The information saved in the history file is described in the following table.

Field Name	Data Type	Remarks
Version	Integer	This field is created only when the File Format is Proprietary. Current version: 002
Event_Type	Integer	1: SECURITY SYSTEM 2: DISPLAY 3: RECIPE 4: REPORT 5: CUSTOM MESSAGES 6: SYSTEM WARNING 7: LOG TAGS
Event_Time	TimeStamp	Timestamp indicating when the event occurred. When the File Format is Proprietary, IWS saves the Event Time in the following format: MM/DD/YYYY HH:MM:SS.MSS.
Event_Info	String	Tag Name
Value	Real	Tag value when the event occurred
Source	String	Name of the task that generated the event
User	String	User logged when the event occurred.
User_Full	String	Full name of the user logged when the event occurred.
Message	String	Event message
Station	String	Name of the station (computer) where the event occurred.
Comment	String	Comment (optional) typed by the operator when the event occurred. This field only exists for Version >=2
Previous_Value	Real	Tag value that occurred before the event. This field only exists for Version >=2
Deleted	Boolean	0: Event message was not deleted 1: Event message was deleted This field is created only when the File Format is Database.
Bias	Integer	Difference (in minutes) from the Time Stamp columns and the GMT time. This field only exists for Version >=2

Last_Update	TimeStamp	Time Stamp 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 File Format is Database.
--------------------	-----------	--

**Tip:**

When saving the Events in a SQL Relational Database (File Format = Database) you can customize the name of the columns created in the database by editing the <ApplicationName>.APP file as follows:

```
[EventLogger]
<DefaultName>=<NewName>
```

For example:

```
[EventLogger]
Event_Info=Information
Message=Event_Message
```

For detailed information about saving events history files, including instructions for configuring database settings, see *Chapter 17: IWS Database Interface*.

Chapter 10: Communication

This chapter explains how you enable InduSoft Web Studio applications to communicate (exchange data values) with other applications, remote devices (such as a PLC or transmitters), and any devices that implement OPC or DDE Servers.

To enable communication, you configure the *task worksheets* provided by IWS. Instructions for configuring these worksheets are provided the following sections:

- **Configuring a Driver:** Explains how to configure a Driver worksheet to implement a communication protocol (OPC, TCP/IP, or DDE).
- **Configuring OPC:** Explains how to configure an OPC worksheet to manage communication between local or remote OPC Clients and Servers.
- **Configuring TCP/IP:** Explains how to configure a TCP/IP worksheet to manage communication between two IWS applications.
- **Configuring DDE:** Explains how to configure a DDE worksheet to manage communication between local or remote DDE Clients and Servers.

Use the **Comm** tab to access all worksheets configured to establish communication with another device or software using available protocols.



Workspace: Comm Tab

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

Configuring a Driver

A communication driver is a *DLL* containing specific information about the remote equipment and implements the communication protocol. To develop a new communication driver, InduSoft provides a driver toolkit. Consult InduSoft for further information.

The *Drivers* folder 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:

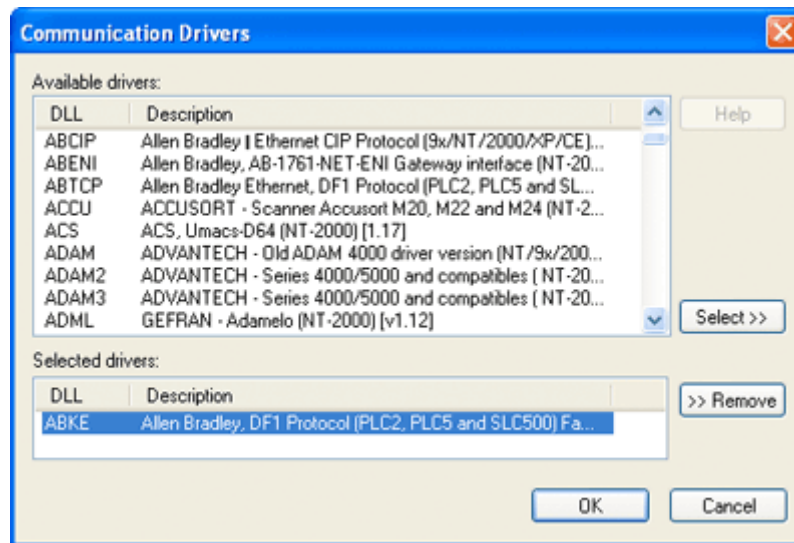
Consult the **Help** menu for a description of the functions and characteristics that are standard for all drivers. When developing an application, you can also refer to the specific documentation provided with each communication driver. This documentation is usually located in the `DRV` directory.

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 InduSoft tags.

Use one of the following methods to add or remove a configured driver:

- Right-click on the *Drivers* folder
- Select **Insert** → **Drivers** from the menu bar

Both methods open a *Communication Drivers* dialog, which displays a list of available drivers.



Communication Drivers dialog

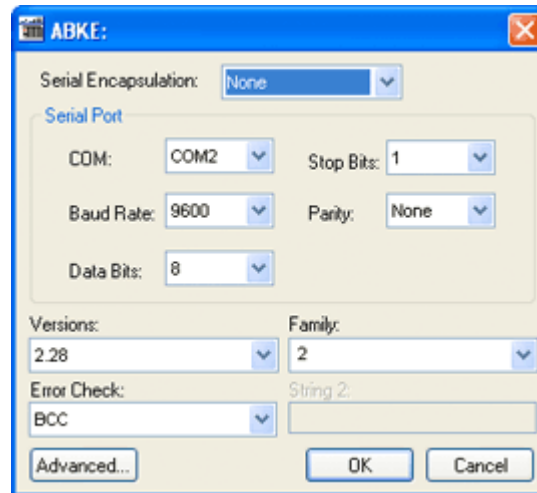
Use the parameters on this dialog, as follows:

- **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

- **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:

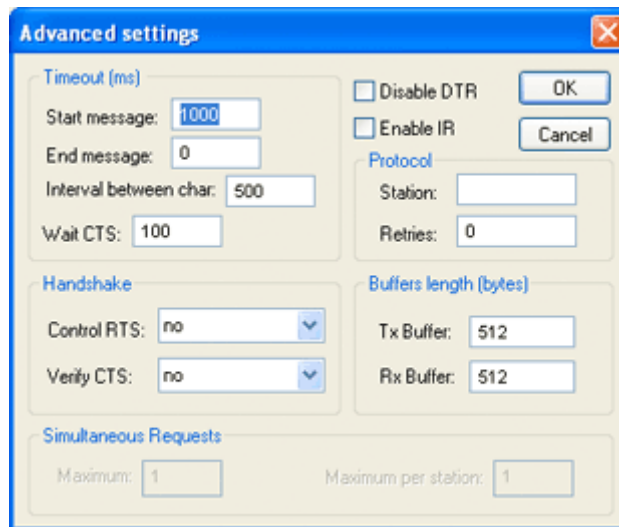
The option **Modem** is not supported for Pocket PC v3.00 or older.

Note:

This section covers only the **Direct** 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 on other encapsulation modes. “Serial Encapsulation Tests” lists the drivers that have been tested with modem, TCP/IP and UDP connections.

- **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 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** check box: Click (*enable*) this box to disable the DTR function (the driver will not set the DTR signal before starting the communication).
- **Enable IR** check box (*available on the Windows CE platform only*): 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
 - **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.
 - **Station** field: Some slave drivers such as the Modbus Slave (MODSL) require a slave network address. Use this field to specify the slave address.
- *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.

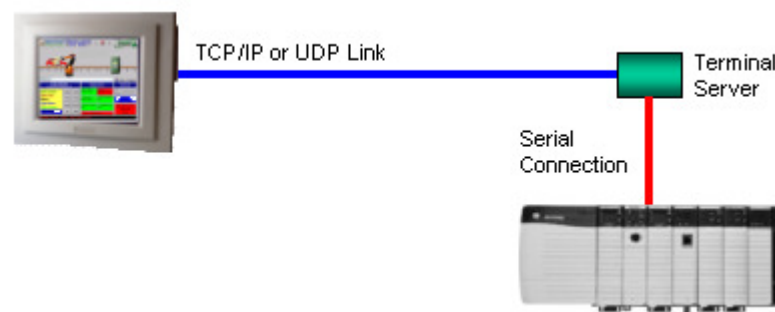
IWS provides two interfaces, which you can use to configure the driver (associating tags from the InduSoft application to device addresses):

- **MAIN DRIVER SHEET:** Provides the easiest method for configuring communication between InduSoft application 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

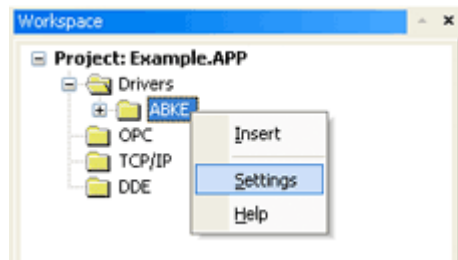
Most of the serial drivers available in IWS allow the use of TCP/IP or UDP/IP encapsulation. The encapsulation mode has been designed to provide communication with serial devices connected to terminal servers on your ethernet or wireless networks. A terminal server can be seen as a virtual serial port. 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:



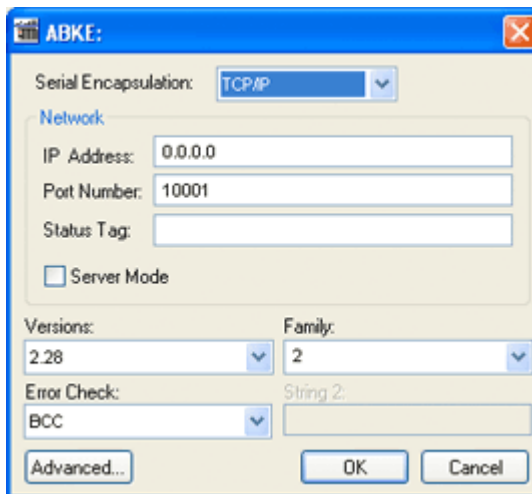
TCP/IP Encapsulation

You can enable the encapsulation by following the steps below:

- ✓ Right-click on the driver's folder, and select Settings from the menu that displays. This will give you access to the communication parameters:



- ✓ In the **Serial Encapsulation** field, select **TCP/IP** or **UDP/IP**:

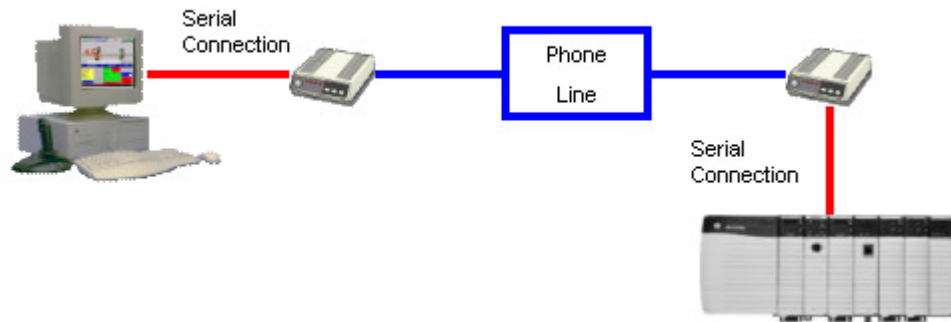


Selecting a Serial Encapsulation

- ✓ The following fields are available:
 - **IP Address** field: Specify the IP Address for the Terminal Server. This field accepts tags between 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

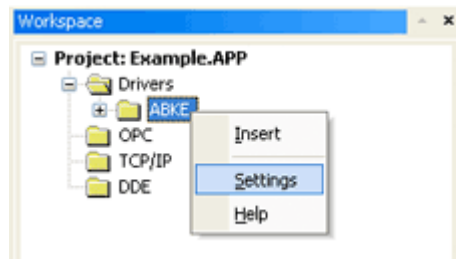
Most of the serial drivers available in IWS allow the use of modem connections. The modem connection has been designed to enable communications with remote serial devices connected through a phone line. The following diagram provides one example of applying this solution:



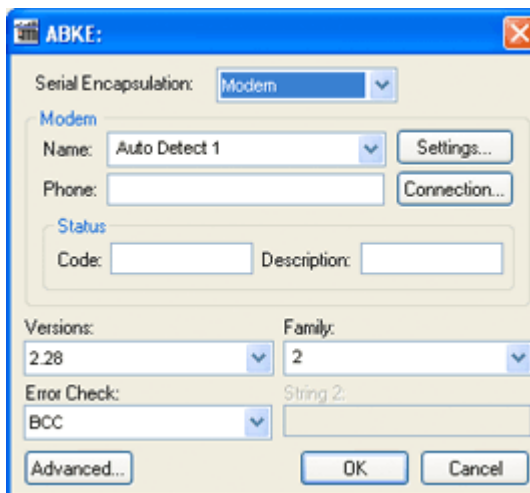
Modem Connection

You can enable the modem connection by following the steps below:

- ☑ Right click on the driver's folder, and select **Settings** from the menu that displays. This will give you access to the communication parameters:



- ☑ In the **Serial Encapsulation** field, select **Modem**:



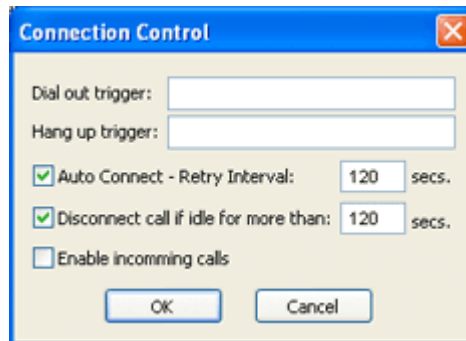
Selecting a Modem

⚠ **Caution:**
The option **Modem** 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 application. 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 application 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 applications. 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:

Code	Description
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 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 if it will work, please contact your distributor.

Driver	Modem	TCP/IP	UDP/IP
MODSL	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
ABKE	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
MODBU	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
OMETH	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

= Item has been tested

= Item has not been tested

Configuring the Driver Worksheets

This section explains how to configure the *MAIN* and *STANDARD DRIVER SHEETS* (or Communication tables) to associate application tags with the device addresses. You can configure multiple *Driver* worksheets—each of which is divided into a *Header* section and *Body* section. The MODBU driver will be used for demonstration purposes.

Note:

Not all drivers require a MAIN DRIVER SHEET. Some drivers only require one *Standard Worksheet*. Consult the driver documentation for specific instructions.

CONFIGURING THE STANDARD DRIVER WORKSHEET

Use the following steps to create a new *Standard Driver Worksheet*:

- ✓ From the IWS development environment, select the **Comm** tab, located below the *Workspace* pane.
- ✓ In the *Workspace* pane, expand the *Drivers* folder and right-click the *MODBU* subfolder.

When the pop-up menu displays, select the **Insert** option:



Inserting a Driver Worksheet

The *Driver* worksheet (*<drivername>.drv*) contains two sections:

- *Header*: Contains all the information about the reads and writes commands
- *Body*: Contains the operator's addresses

Configuring the Header

The *Driver* worksheet header contains configuration information required for the driver's functions. Initially, you must create a new *Driver Configuration* worksheet for each area with which you want to communicate.

The screenshot shows a configuration window with the following fields:

- Description: Coil Status
- Increase read priority
- Read Trigger: RdTr[1]
- Enable Read when Idle: RdEn[1]
- Read Completed: RdCpl[1]
- Read Status: RdSt[1]
- Write Trigger: WrTr[1]
- Enable Write on Tag Change: WrEn[1]
- Write Completed: WrCpl[1]
- Write Status: WrSt[1]
- Station: 1
- Header: 0x1
- Min: []
- Max: []

Header Section of Driver Worksheet

The *header* contains the following fields:

- **Description:** Type a description of the worksheet, such as area types, their ranges, and if the worksheet is **Read**, **Write**, or **Both**. This description displays in the *Workspace*, in the *Drivers* folder.
- **Increase read priority:** For read worksheets (and there can be more read worksheets with the same read trigger or enabled when idle) and a write event happens, the worksheet with the highest priority will be the first worksheet on the next reading called by the read trigger or the “read when idle” event.
- **Read Trigger:** Contains a tag that always generates a read event when value of the tag in the field changes.
- **Enable Read when Idle:** Contains a tag or value that always enables a continuous read when it the value is greater than zero.
- **Read Completed:** Contains a tag value that toggles when a read event is finished.
- **Read Status:** Contains a tag that always has its value filled with an integer value, when a read event finishes. If this value equals zero, the event is completed successfully. If any other value displays, the event completed with an error. You can view the error message in the Logwin module (for NT/2000) or check the **MODBU.MSG** file in the InduSoft Web Studio's **DRV** directory.
- **Write Trigger:** Contains a tag that generates a write event for the entire worksheet, whenever the value of the tag changes.

IMPORTANT!

When using this feature, the driver writes the tag value to the PLCs memory. This operation writes using blocks, from the first worksheet operator up to the last. If there is an operator that has not been declared in the worksheet, and its address is between the first and the last block, the tag will receive the value zero. Therefore, be sure about what you want to write when using this trigger, and verify whether there is any kind of hole in the worksheet that can cause problems for the system or the PLC's program.

- **Enable Write on Tag Change:** When the value of one of the tags in the body is different from the write trigger, IWS writes the changed tag in the worksheet (its value is greater than zero).
- **Write Completed:** Contains a tag value that toggles when a writing event finishes.
- **Write Status:** Contains a tag that always fills with an integer value, when a reading event finishes. If this value is equal to zero, the event is successful. Any other value indicates an error. You can view the error message in the *Logwin* module (for NT/2000) or check the **OMPLC.MSG** file in the InduSoft Web Studio's **\DRV** directory.
- **Station:** Must (if indicated in the driver's help file) contain the CPU's ID, Unit Number, or PLC Address it relates to this specific worksheet. Each driver has a different syntax for this field.

For example, the *GE Fanuc SNP* driver allows you to identify the PLC using all ASCII characters, but the *OMRON Host Link Protocol* allows from only 1 to 31 addresses called *Unit Numbers*.

Typically, you use the address of the PLC in a device network.

You can also enter a tag between curly brackets (for example: {tag})

 **Notes:**

- You cannot test the existence of tags entered inside curly brackets (or entered in a different form from tags in other fields), because they have not been created in the *Tags* database yet. In other words, if you type an uncreated tag, the system cannot work properly.
- **Station** is a string field and must be filled in correctly or the driver will not work properly.

- **Header:** Must contain the worksheet header. This field is extremely important. Each driver has a different syntax for this field; however, you must type something like the operator's type, followed by the initial address.

The following table contains some examples:

Driver	Header	Meaning
MODBUS	4X:100	4X indicates that this worksheet will communicate with the Holding Registers, from the address 100 on. In the AEG 984 case, from the address 400100 on.
OMPLC (Host Link)	IR:0	IR indicates that this worksheet will communicate with the I/O and Internal Relays, from the address 0 on. In the C200H case, from the address IR00000 on.
FANUC (SNP)	%M	%M indicates that this worksheet will communicate with the %M discrete internal operator. There's no initial address to this driver.
ABKE (DF1)	N7:0	N7 indicates that this worksheet will communicate with the N7 file, from the address 0 on. In the PLC-5/40 case, from the address N7:0.
AS511 (Siemens PG Port)	DB5:10	DB5 indicates that this worksheet will communicate with the Data Block number 5, from the Data word 10 on.

So, for each driver this syntax can vary. Most of the time, this is the address of the PLC in a device network.

For example, let's use the MODBUS syntax:

<reference>:<initial address>

Where:

<reference> is the reference with which you want to communicate

For example, if the header is 4X:1, IWS will read the worksheet from 4000001 until the highest offset configured in the Address column.

You can use the following references:

0X: Coil Status

1X: Input Status (read only)

3X: Input Register (read only)

4X: Holding Register

ID: Report Slave (read only)

There are no limits for the initial address, but be careful when specifying address limits. For example, on the PLC there is no 30500. The **Header** field accepts the syntax **3X:500**, but the runtime will not find this register.

Where **Read Only** is indicated, the write functions will not work. It is not safe to specify write for the **Input Status**, **Input Registers**, and the **Report Slave** functions.

This field can also be filled with a tag between curly brackets (for example: {tag}).

 **Note:**

As with the **Station** field, you cannot test the existence of tags entered inside curly brackets (or entered in a different form from tags in other fields), because they have not been created in the *Tags* database yet. In other words, if you type an uncreated tag, the system cannot work properly.

When you first create a new *Driver* worksheet, the field is blank. After you place the cursor on this field (even you try to make it blank again) IWS automatically inserts the default **0X:1** string. From this point on, you cannot make the field blank again. You can however, change the value to another valid header.

- **Min / Max:** Becomes enabled after you enable () the check-box. When selected, this parameter enables a range of values that can be converted into an engineering format. These fields determine the minimum and maximum range of values. For example, memory holds values from 0 to 4095 meaning 0% to 100% in the user interface. This setting takes effect for all tags in the worksheet. In this example, the tag parameters **Min** and **Max** must be set 0 to 100 respectively.

Configuring the Body

The *Driver* worksheet's body section assigns the PLC's memory address to declared tags and handles the engineering units.

	Tag Name	Address	Div	Add
1	TAG_CS[1]	0		
2	TAG_CS[2]	1		
3	TAG_CS[3]	2		
4	TAG_CS[4]	3		
5	TAG_CS[5]	4		
6	TAG_CS[6]	5		
7	TAG_CS[7]	6		
8	TAG_CS[8]	7		

Body Section of Driver Worksheet

The *Body* section contains four columns:

- **Tag Name:** Contains tags used by the communication driver.
- **Address:** Contains addresses to read and write tag values to in the equipment.

As with the **Header** field, this column is different for every driver. Typically, you type the offset from the initial address you configured for the **Header** field. In some cases, you can indicate the specific **Address** bit.

For our driver example case, type the offset from the initial address you configured for the **Header** field. You cannot enter a negative offset—the value 0 will overwrite a negative value.

- **Div / Add / Max / Min:** Configure as follows:

Column	Range of Values	Mean
Div	Any Integer or Real	In read commands: Tag = (Host value) / DIV In write commands: Host value = Tag * DIV
Add	Any Integer or Real	In read commands: Tag = (Host value) + ADD In write commands: Host value = Tag – ADD
Min	Any Integer or Real	Defines the minimum value assigned for the tag, when the corresponding host's value is equal to the value defined in the field Min of the Driver worksheet's Header.
Max	Any Integer or Real	Defines the maximum value assigned for the tag, when the corresponding host's value is equal to the value defined in the field Max of the Driver worksheet's Header.

Notes:

For read operations:

$$\langle \text{tag} \rangle = ((\langle \text{value in the equipment} \rangle) / \text{Div}) + \text{Add}$$

For write operations:

$$\langle \text{value in the equipment} \rangle = (\langle \text{tag} \rangle - \text{Add}) * \text{Div}$$

If you do not configure the columns as specified in the preceding table, the columns will not be configured and the *Driver* worksheet tags will receive the same value as the address configured.

Use the following steps to specify header tags:

- ☑ Specify the following tags in the *Driver* worksheet **Header** fields. All of the tags will be arrays, and you must type each element on each worksheet.

For example, **RdTr [1]** in the **Read Trigger** field of the *ABKE001.DRV* worksheet, and **RdTr [5]** in the *ABKE005.DRV* worksheet, and so forth.

Tag Name	Size	Type	Description
RdTr	0	Boolean	Boolean tag that will be on the "Read Trigger" fields
RdEn	0	Boolean	Boolean tag that will be on the "Enable Read when Idle" fields
RdCpl	0	Boolean	Boolean tag that will be on the "Read Complete" fields
RdSt	0	Integer	Integer tag that will be on the "Read Status" fields
WrTr	0	Boolean	Boolean tag that will be on the "Write Trigger" fields
WrEn	0	Boolean	Boolean tag that will be on the "Enable Write when Idle" fields
WrCpl	0	Boolean	Boolean tag that will be on the "Write Complete" fields
WrSt	0	Integer	Integer tag that will be on the "Write Status" fields
Station	0	String	String tag that will be on the "Header" field
Header	0	String	String tag that will be on the "Station" field

- ☑ Specify **TAG_DRV** as an *Array* tag, size 10, for the communication tags.

- Configure a *Driver* worksheet and a *PLC Driver* screen to look like the following figure:

Description:
 Modbus driver worksheet Increase priority

Read Trigger: Enable Read when Idle: Read Completed: Read Status:

Write Trigger: Enable Write on Tag Change: Write Completed: Write Status:

Station: Header: Min:
 Max:

	Tag Name	Address	Div	Add
3	Tag_DRV[3]	3		
4	Tag_DRV[4]	4		
5	Tag_DRV[5]	5		
6	Tag_DRV[6]	6		
7	Tag_DRV[7]	7		
8	Tag_DRV[8]	8		
9	Tag_DRV[9]	9		
10	Tag_DRV[10]	10		
11				

Configuring the MODBU Driver Worksheet

header = #####

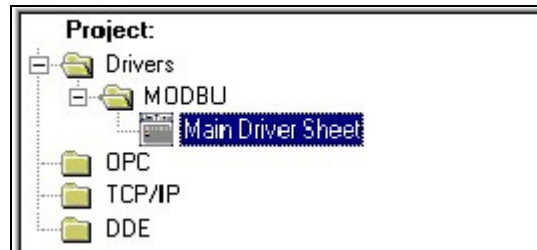
PLC

<input type="button" value="RDTR"/>	rdtr = #####	Tag_DRV[1] = #####
<input type="button" value="RDEN"/>	rden = #####	Tag_DRV[2] = #####
	rdst = #####	Tag_DRV[3] = #####
		Tag_DRV[4] = #####
		Tag_DRV[5] = #####
<input type="button" value="WRTR"/>	wtrg = #####	Tag_DRV[6] = #####
	wren = #####	Tag_DRV[7] = #####
<input type="button" value="WREN"/>	wrst = #####	Tag_DRV[8] = #####
		Tag_DRV[9] = #####
		Tag_DRV[10] = #####

PLC Driver Screen

CONFIGURING THE MAIN DRIVER SHEET (MDS)

When you add the MODBUS driver to your application, the program automatically adds the *MAIN Driver Worksheet (MDS)* to the *MODBUS* driver folder as shown:



Selecting the Main Driver Sheet

You can use the MDS to associate IWS tags to addresses in the PLC. Most MDS parameters are standard for any driver. Use the following instructions to configure the parameters that are specific to the MODBUS driver:

- ☑ Double-click on the *Main Driver Sheet* icon to open the following worksheet:

MODBUS - MAIN DRIVER SHEET

Description:

Disable:

Read Completed: Read Status:

Write Completed: Write Status:

Min: Max:

	Tag Name	Station	I/O Address	Action	Scan	Div	Add
*				Read+Write	Always		
*				Read+Write	Always		
*				Read+Write	Always		
*				Read+Write	Always		
*				Read+Write	Always		

Main Driver Worksheet

- ☑ Configure the following fields on this worksheet:
 - **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. 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 the runtime.

⇒ **Tip:**
 Configuring a string tag between curly brackets in the Station field of the Main Driver Sheet (MDS) is especially useful when configuring applications 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 each PLC register, using the following syntax:

<Type> : **<Address>** (for example, **4X:20**) or

<Type> : **<Address>** . **<Bit>** (for example, **4X:20.6**)

Where:

Type is the register type. Type one of the following: **0X**, **1X**, **3X**, **4X**, **FP**, **FPS**, **FP3**, **FP3S**, **DW**, or **ID**.

Address is the register address of the device.

Bit is the bit number (from 0 – 15) of the word address. This parameter is *optional* and can be combined with 3X- or 4X-type addresses only.



Caution:

You must use a non-zero value in the **Station** field and you cannot leave the field blank.

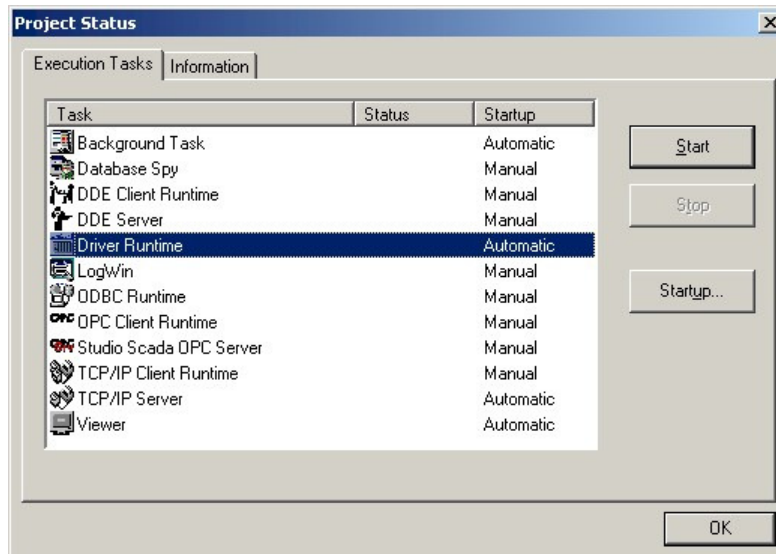
Executing the Driver

After adding the driver to a project, Studio sets the project to execute the driver automatically when you start the runtime environment.

To verify that the driver runtime task is enabled and will start correctly, perform the following steps:

- ✓ Select **Project > Status** from the main menu bar.

The *Project Status* dialog displays:



Project Status Dialog

- ✓ Verify that the *Driver Runtime* task is set to **Automatic**.
 - If the setting is correct, click **OK** to close the dialog.
 - If the *Driver Runtime* task is set to **Manual**, select the **Driver Runtime** line. When the **Startup** button becomes active, click the button to toggle the *Startup* mode to **Automatic**.
- ✓ Click **OK** to close the *Project Status* dialog.
- ✓ Start the application to run the driver.

Configuring OPC

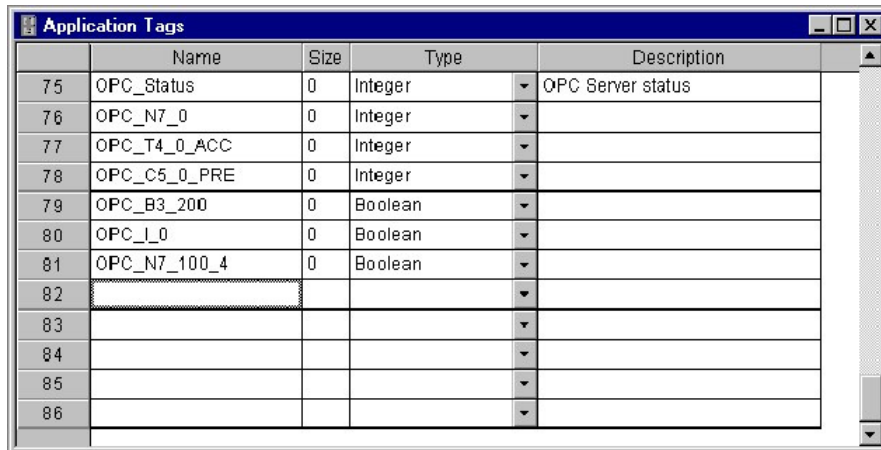
The *OPC* folder allows you to configure OPC interfaces between the application and an OPC server. The InduSoft *OPC Client* module enables the IWS system to communicate with any device that implements an OPC server. Before using the InduSoft OPC Client module, you must install and configure the OPC server on the machines running your application.

On the client machine, use the OPC Client Configuration program to configure the server identifier, communication parameters, and the items you want to connect. To access the client configuration, add a new OPC Client document to the "COMM" table.

Configuring an OPC Client

Use the following steps to configure an OPC Client:

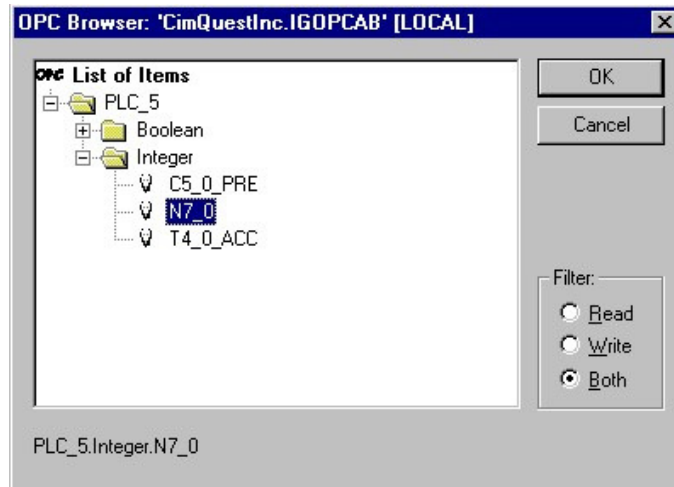
- ✓ Select the **Comm** tab, right-click the *OPC* folder and insert a new *OPC Client* worksheet.
- ✓ Select a registered *OPC Server* (CimQuestInc.IGOPCAB) from the **Server Identifier** combo-box to register *InGear OPC AB*.
- ✓ Create a new set of tags to communicate with the OPC Server, as shown:



	Name	Size	Type	Description
75	OPC_Status	0	Integer	OPC Server status
76	OPC_N7_0	0	Integer	
77	OPC_T4_0_ACC	0	Integer	
78	OPC_C5_0_PRE	0	Integer	
79	OPC_B3_200	0	Boolean	
80	OPC_I_0	0	Boolean	
81	OPC_N7_100_4	0	Boolean	
82				
83				
84				
85				
86				

- ✓ In the *OPC Client* worksheet, type `OPC_Status` in the **OPC Status** field.
- ✓ In the first **Tag Name** column row, type `OPC_N7_0`.

- To associate this tag to the OPC Server item, right click on the **Item** column and click **OPC Browser** and browse all the configured OPC Server items. Select the **N7_0** item.



Select N7_0

Your *OPC Client* worksheet should look like the following:

OPC OPCCL001.OPC

Description: Server Identifier: Disable:

Read Update Rate (ms): Percent Deadband: Status:

Remote Server Name: Read after writing

Accept Tag Name in the Item column

	Tag Name	Item	Scan	Div	Add
*			Always		
*			Always		

OPC Client Worksheet

The *OPC Client* worksheet contains the following fields:

- Description:** Type a description of the OPC module for documentation purposes only. (The OPC Client module 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 zero, to disable communication with the OPC server. Specify zero, or leave the field blank to enable communication.

- **Read Update Rate (ms):** Specify how often the server should update this group (in milliseconds). Specify zero 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:** Current status. Good status is 1.
- **Remote Server Name:** Node name or IP address of server on node network.
- **Read after writing** check-box: Check this check-box force the OPC Client worksheet to execute a synchronous read command. This will read the value from an item from the OPC Server just after writing a value to this item. This option must be used to guarantee synchronization between the value of the tags on IWS and the items from the OPC Server, when the PLC program overwrites the values written by IWS through the OPC Server.
- **Accept Tag Name in the Item column** check-box: 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 the 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:** Type 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 {TagName} in this field, allowing the user to change the item names dynamically, during the runtime.


- **Scan:** Specify one of the following:
 - **Screen:** IWS performs an update when you open a screen containing the specified tag.
 - **Always:** IWS performs an update in the Read Update Rate specified in the worksheet header.
- **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 run-time module, you can choose to run it automatically on start up, or run the module manually by selecting **Project > Status** from the menu bar. After running this program, a small icon displays in your system tray.

To close the OPC client run-time module, right-click the icon in the system tray, and click **Exit**.

 **Note:**
IWS and CEView also provide an OPC Server communication module named *Studio.Scada.OPC*. This module 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 IWS tags (Application Tags, System Tags, and Shared tags) using the OPC interface.

In addition, you can start the OPC Server module automatically when you start the application. Select the OPC Server module in the **Project > Status** dialog, click the **Startup** button, and specify **Automatic**.

 **Tip:**
You can also use the OPC interface to exchange data between remote stations running IWS 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 using Windows CE, 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 Module (NT and CE)" in the Users Guide and Technical Reference Manual.

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 application, 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:

1. DCOM Security Configuration – The *DCOM Tutorial* at http://www.opcactivex.com/Support/DCOM_Config/DCOMConfigUtility/dcomconfigutility.html
2. *Using DCOM with Windows XP + SP2* at http://www.indusoft.com/Using_DCOM_w_WINXPSP2.pdf

- **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 CE 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 Studio distribution, most probably in **<Studio installation folder>\Redist\Wince <x.x>\<processor>**

If you do not find the .dll in the folder for your processor, contact your Studio distributor.

Configuring an OPC Server

The InduSoft Web Studio *OPC Client* module enables the InduSoft Web Studio system to communicate with any device that implements an OPC Server. See the manufacturer's documentation for configuration instructions.

To communicate with a third-party OPC Client and have IWS as the OPC Server, you need to be sure that the **Project > Status > Studio Scada OPC Server** is set to **Automatic**. All of the tags in our database will be automatically available. You can have multiple applications. Only the current application's tags are available.

Configuring TCP/IP

The IWS *TCP/IP Client/Server* modules enable two or more InduSoft applications to keep their databases synchronized. These modules use InduSoft's TCP/IP protocol to make the communication between the applications.

Before using the IWS *TCP/IP Client/Server* modules, you must install and configure the TCP/IP protocol on the machines that will run these modules.

Configuring the Client

On the client machine, you must use the **TCP/IP Client Configuration** worksheet to configure the Server IP address and the tags you want to share with the server.

- ☑ In the *Workspace*, select the **COMM** tab and right-click the *TCP* folder to insert a new *TCP* worksheet.
- ☑ Configure the following fields:
 - **Description:** Type a description of the worksheet for documentation purposes only. The *TCP/IP Client* module ignores this field.
 - **Connection Status:** Type a tag name. The *TCP/IP Client* module updates this tag according to connection status. If the tag value is 0 (zero), then the connection is OK. Otherwise, the Windows Socket library returns an error code.
 - **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 the runtime.
 - **Server IP Address:** Type the server IP Address. The entry can be a string or you can use a tag enclosed by brackets. For example, if you fill this field with {tag_name}, the *TCP/IP Client* module will try to connect to the server indicated by the tag_name tag.
 - **Tag Name:** Type the tags you want to share with the server. If the tag is an array or a class (or both), every element and member is shared. You should type the tag name only in this field — without specifying the index or class member. If you specify an index or a class, the *TCP/IP Client* module ignores it.
 - **Remote Tag:** Type the name of the tag to be linked with the tag specified in the **Tag Name** field. This field is *optional*. If you leave it in blank, the same tag name will be used for both the client and the server.

⚠ Warning:

If you need to share an array, then the tag in the server should contain the same number of elements as the tag in the client. If the tag is a class, then the class definition should be the same in both server and client applications. If you do not follow these rules, unpredictable results will occur.

Setting Custom Parameters

You can configure the following parameters for your *Application Configuration* (.APP) file (select **Project > Settings** on the main menu bar):

- **[TCP] Port:** TCP/IP port number. Default is **1234**. This parameter should be the same in both the client and server machines.
- **SendPeriod:** Time in milliseconds before the client/server module will update the tag values of the other machine. Default is **250**.

- **ConnectRetryTimeout:** Time in seconds before the client should retry to connect to the server. Default is 30.

Only the client module uses the **ConnectionRetryTimeout**.

You can run the *TCP/IP Client Module* automatically or manually. From the main menu bar, select **Project** → **Status**. On the **Execution Tasks** tab, set the **TCP/IP Client Runtime** to **Manual** or **Automatic**.

After running this program, a small icon will display in your system tray.

Configuring the TCP/IP Server

On the server machine, you do not have to configure anything. You just have to run the IWS *TCP/IP Server* module. In the development environment window, go to the *Project Settings* dialog and set the TCP/IP Server to run automatically. From the main menu bar, select **Project** → **Status**. On the **Execution Tasks** tab, set the **TCP/IP Server** to **Automatic**. When running this program, a small icon will display in your system tray.

To close the IWS *TCP/IP Server* module, right-click on the icon in the system tray, and select **Exit**.

Configuring DDE

The *DDE* folder allows you to configure a DDE Client configuration to a DDE Server application such as 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 applications. IWS provides interfaces that run as clients or as servers. See **DDE Client Runtime** and **DDE Server** in the *Execution Tasks (Project → Status)*.

- To run as a server, start the DDE or NetDDE server task as described in *Execution 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 application.

Notes:

- When running NetDDE, IWS accepts the **WRITE** triggers only. To read data, you must configure a `write` command on the server computer.
- By default, the DDE Client module from IWS 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 <ApplicationName>.APP file (you can use Notepad to edit this file):
[Options]
DDEANSI=1

Configuring DDE Client

To open a new *DDE* worksheet, right-click on the *DDE* folder and click the prompt screen.

A new *DDE* worksheet displays:

	Tag Name	Item	
1			
2			
3			
4			
5			

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 application 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

Configuring the DDE Server

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.

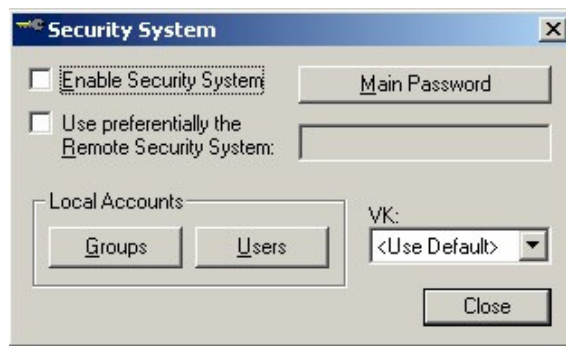
Use the *DDE client* worksheet **Body** parameters to relate each tag to each ITEM part of the DDE server address, as follows:

- **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.

Chapter 11: Configuring a Security System

You use the *Security System* dialog to create groups and users, and to configure their access privileges to InduSoft Web Studio tools and applications.

To access this dialog, simply right-click the *Security* folder on the **Database** tab and select **Settings** from the pop-up menu.



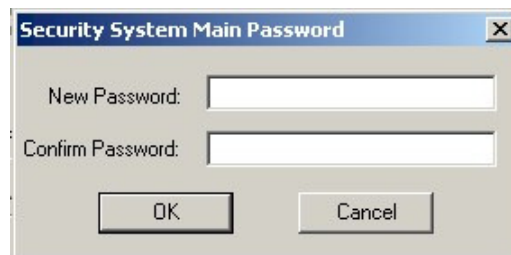
Security System Dialog

This dialog contains the following features:

- **Enable Security System** check-box: Check (✓) this box to enable the IWS Security System.
- **Main Password** button: Opens the *Security System Main Password* dialog so you can define passwords granting access to the security system.
- **Groups** button: Opens a *Groups* dialog, where you create and maintain user groups.
- **Users** button: Opens a *Users* dialog, where you create and maintain users.
- **VK** pane: Virtual Keyboard type used for the *Security System* dialogs (*LogOn*, *Change Password*, and so on). You need to enable the **Virtual Keyboard** option on the **Project > Settings > Runtime Desktop** interface before configuring the Virtual Keyboard for this interface.

Entering a Password

When you click the **Main Password** button, the *Security System Main Password* dialog opens so you can enter a password for accessing the InduSoft Web Studio Security System.



Security System Main Password Dialog

This dialog contains two fields:

- **New Password** field: Type a password.
- **Confirm Password** field: Retype the same password to confirm it.
If the passwords are different, IWS will prompt you to type it again.

IMPORTANT!

After defining your password, you must use that password each time you access the Security System, therefore it is mandatory that you remember it.

Note:

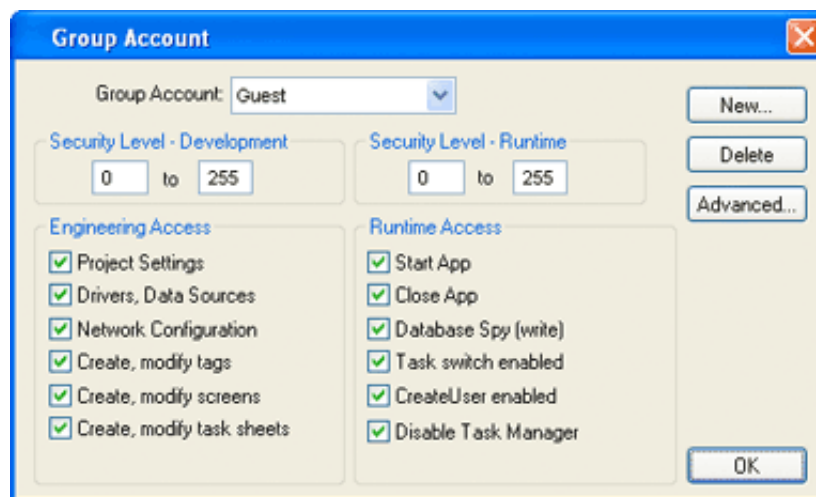
Applications developed in InduSoft Web Studio version 6.1 SP2 or later use case-sensitive passwords — that is, passwords created with both upper and lowercase characters *must* be entered the same way by the user.

Applications developed in earlier versions of IWS use case-insensitive passwords.

Defining Groups

The *Group Account* dialog enables you to create and maintain user groups, enable/disable operations, and set security level ranges for development and runtime systems.

You access this dialog by clicking on the **Groups Account** button on the *Security System* dialog. Alternatively, you can open the *Groups* folder located in the *Security* folder on the **Database** tab, or select **Insert** → **Security Group** from the main menu bar.



Group Account Dialog

The features on this dialog include:

- **Group Account** combo-box: Identifies the group to which a user belongs.

Note:

You cannot delete the *Guest* group (it is the *default* group).

- **Security Level – Development fields and Security Level – Runtime** fields: Defines the security level for a group (0 to 255).

Every object used for data input on a screen (such as input commands, sliders, or screens) has a **Security Level** field. If the object's security level is not in the group security range of the user logged in at the moment, then that object will be disabled. A level 0 (zero) means that the object is always enabled.

- **Engineering Access** check-boxes: Controls which engineering (development) tasks users in this group can access when they log on.

➔ **IMPORTANT!**

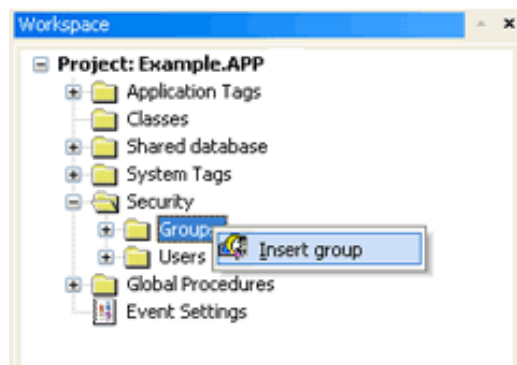
You also can set the security level for documents (such as worksheets and screens) to protect them in the development environment.

- **Runtime Access** check-boxes: Controls which runtime modules users in this group can access when they log on.
- **New** button: Opens the *New Group Account* dialog used to create new groups.
- **Delete** button: Deletes the currently selected user group.

Before setting up your security system, you must decide which groups and users you want to configure. You must define the rights of each group in your environment.

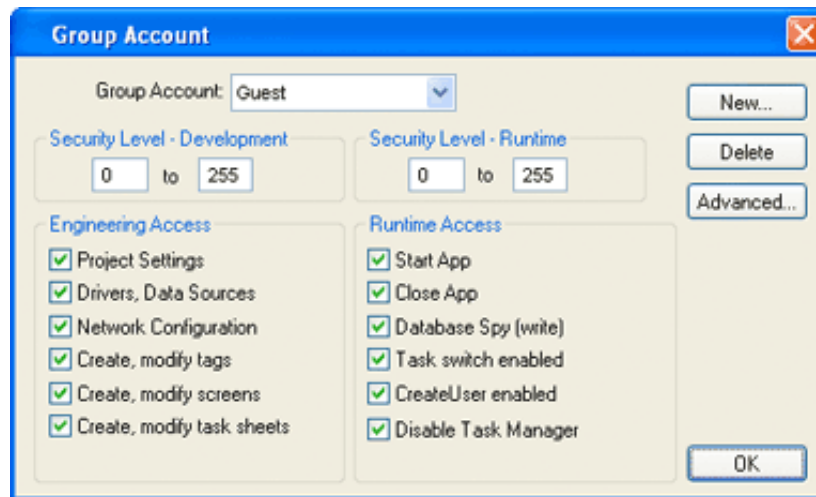
Use the following procedure to create these groups:

- ✓ In the *Workspace*, select the **Database** tab and double-click on the *Security* folder to view the subfolders.
- ✓ Right-click the *Group* folder and select **Insert Group** from the pop-up menu:



Inserting a Group

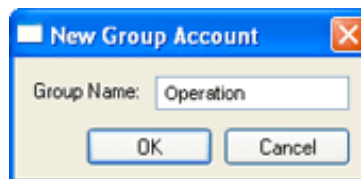
The *Group Account* dialog displays:



Group Account Dialog

Remember, you cannot delete the default group called *Guest*, so you must create a new group, as follows.

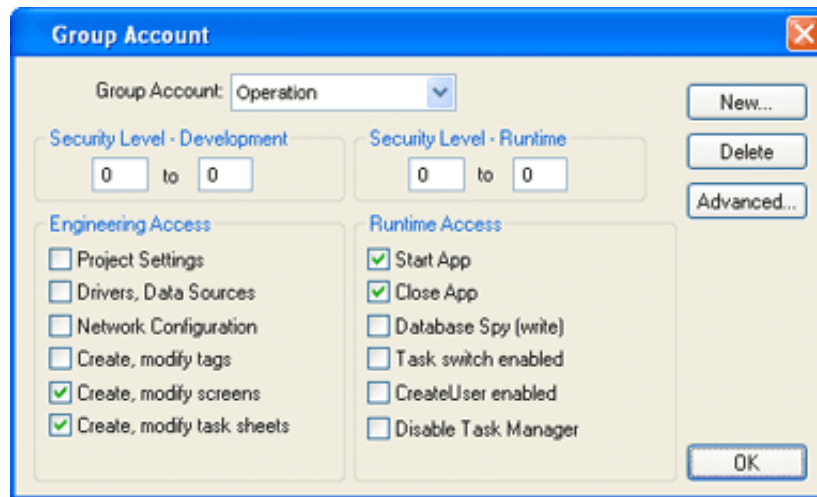
- ✓ Click the **New** button, and when the *New Group Account* dialog displays, type a group name into the field provided. (For this example, type **Operation**.) Click **OK** to close the dialog.



Entering the Group Name

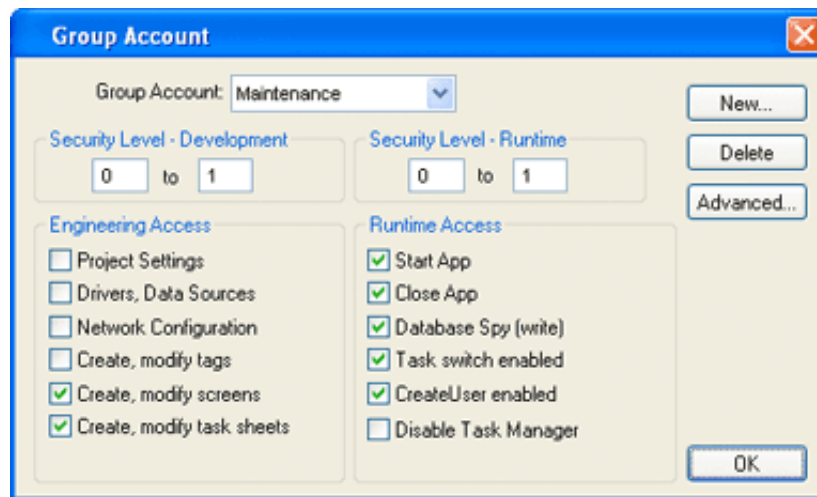
- ✓ Return to the *Group Account* dialog and if the new account name is not already displayed, select **Operation** from the **Group Account** combo-box.

- ☑ Configure the access rights for this group as shown:



Operation Access Rights

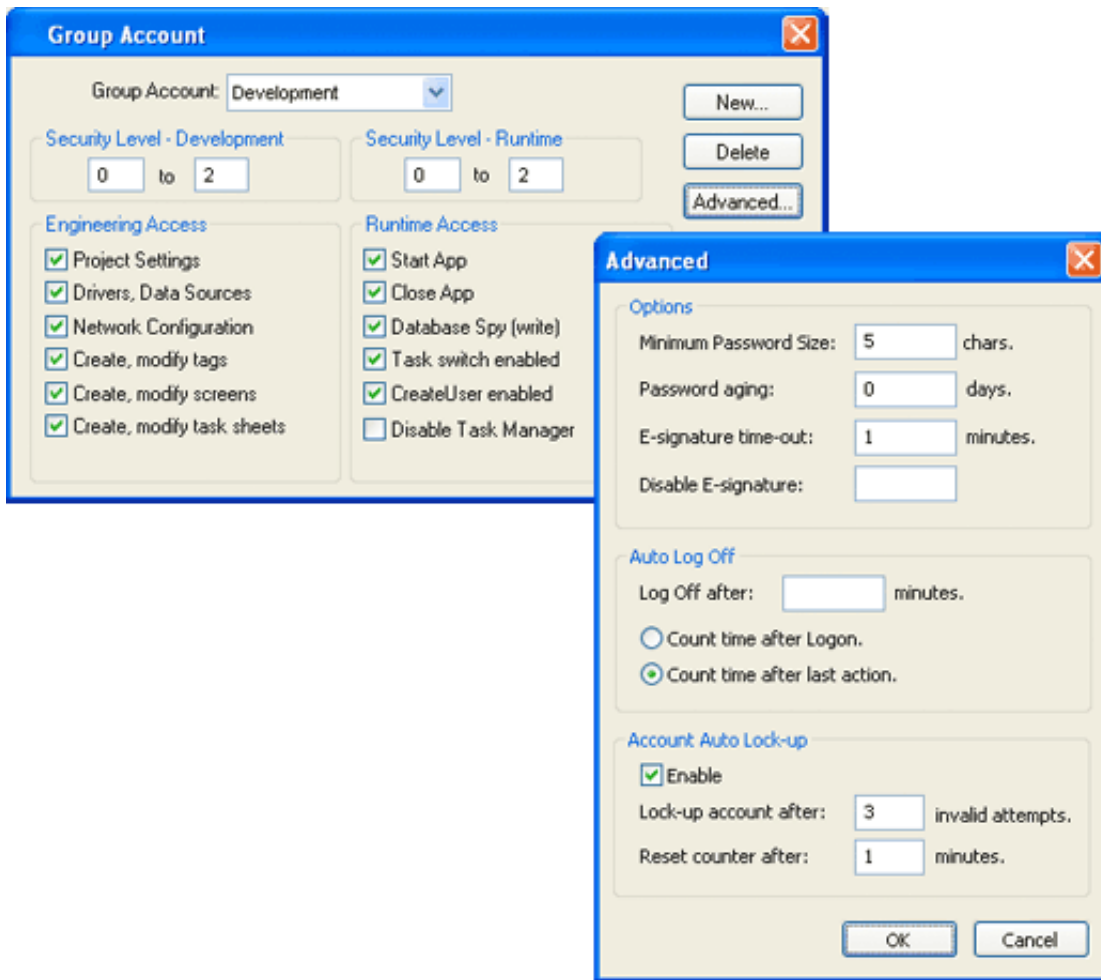
- ☑ Click the **New** button again and create the *Maintenance* group. Click **OK** to close the *New Group Account* dialog.
- ☑ Select **Maintenance** from the **Group Account** combo-box and configure the access rights for the group as follows:



Maintenance Access Rights

- ☑ Finally, repeat the procedure once more to create the *Development* group account.

- ☑ Select **Development** from the **Group Account** combo-box and configure the following access rights:



Development Access Rights

- ☑ Click **OK** to save this configuration.

Notes:

- Each group has a range for the level in development and runtime. In some worksheets (for example, in the *Math* worksheet), you can set an access level to provide the group with access to configure that worksheet.
- When users log into the system they must be associated with a group in the specified access level range (development) for that worksheet.
- You also can configure access levels for buttons so that only authorized users can execute commands (scripts) configured for those buttons in the runtime environment.

Defining Users

Next, you must create new users and associate these users to the group accounts you just created. Use the following steps:

- ✓ In the *Workspace*, select the **Database** tab and double-click on the *Security* folder to view the subfolders.
- ✓ Right-click the *Users* folder and select **Insert User** from the pop-up menu.

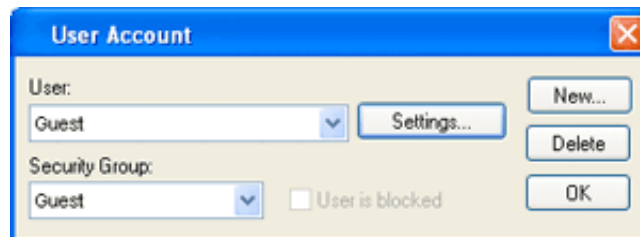


Inserting New Users

Note:

Alternatively, you can access the *User Account* dialog from the *Users* folder located in the *Security* folder on the **Database** tab, or by selecting **Insert** → **User** from the main menu bar.

The *User Account* dialog displays:



User Account Dialog

Remember, you cannot delete the default user called *Guest*, so you must create new users, as follows.

- ✓ Click the **New** button, and when the *New User Account* dialog displays, type a user name into the field provided. (For this example, type **Operator_1**.) Click **OK** to close the dialog.

Creating Operator_1 User

- ✓ To associate this user with a group account, return to the *User Account* dialog and verify that **Operator_1** is displayed in the **User** combo-box.
- ✓ Select **Operation** from the **Security Group** combo-box.

Operator User Account

- ✓ To specify a password for this user, click the **Settings** button, and when the *Settings* dialog displays, type **oper_1**. Click **OK** to close the dialog.
- ✓ Reopen the *User Account* dialog and add the next user as follows:
 - Click the **New** button and when the *New User Account* dialog displays, type **MaintEng_1** into the **User Name** field. Click **OK** to close the dialog.
 - Associate this user to the *Maintenance* group account and then click the **Settings** button to define the **main_1** as their password.

Creating the MaintEng_1 User

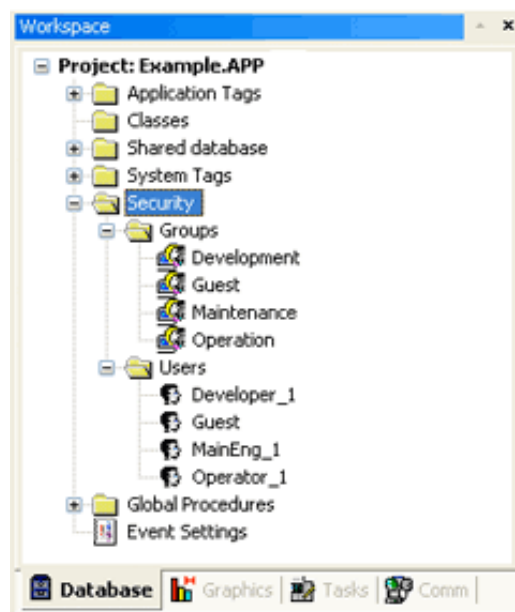
- ✓ Reopen the *User Account* dialog once more and add the last user as follows:
 - Click **New** and create the **Developer_1** user.
 - Associate this new user to the **Development** group account and specify **deve_1** as their password.



Creating the Developer_1 User

- ✓ Click **OK** to save the configuration.

Now, if you expand the *Security* folder, you should be able to open all of the subfolders and verify all the groups and users that you created.



Expanded View of Security Groups and Users

Note:

You also can use the *CreateUser* function to create new users. Within the application, users created in this manner will appear in the *Users* folder.

SETTINGS BUTTON

You can use the **User is blocked** check-box and the **Settings** button (which replaces the **Password** button used previously) to control a user's access to the application. You can access these features from the *User Account* dialog.

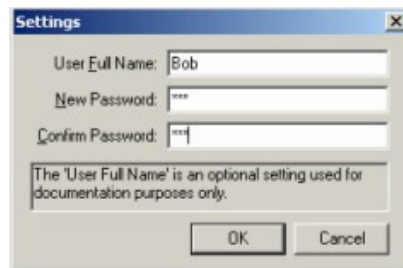
Use the following steps to open this dialog and configure user access:

- ✓ In the *Workspace*, expand the *Security* folder and right-click on a user name.
- ✓ When the pop-up menu displays, select **Properties** to open the *User Account* dialog:



Security – User Properties: User Account Dialog

- ✓ If necessary, click the **User is blocked** check-box to block the selected user.
- ✓ Click the new **Settings** button to open the *Settings* dialog:



User Account: Settings Dialog

- ✓ 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.
- ✓ When you are finished, click **OK** to apply the changes and close the *Settings* dialog.

Note:

Applications developed in InduSoft Web Studio version 6.1 SP2 or later use case-sensitive passwords — that is, passwords created with both upper and lowercase characters *must* be entered the same way by the user.

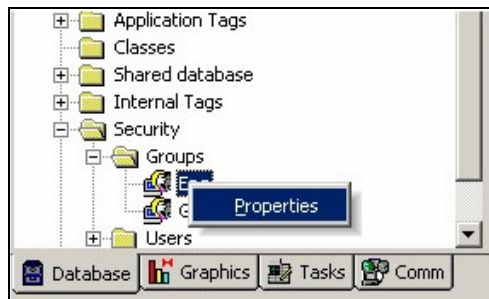
Applications developed in earlier versions of IWS use case-insensitive passwords.

ADVANCED BUTTON

You can use the **Advanced** button to control a user's access to the application. You can access these features from the *User Account* dialog.

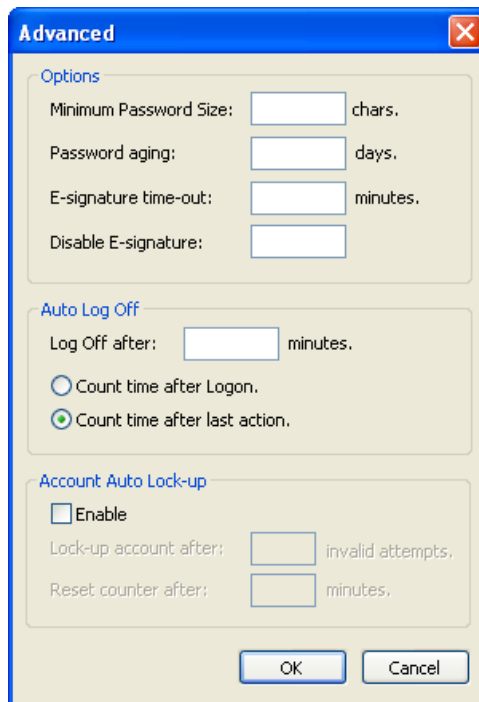
Use the following steps to open this dialog and configure user access:

- ☑ In the *Workspace*, expand the *Security* and *Groups* folders, and then right-click on a group name.
- ☑ When the pop-up menu displays, select **Properties** to open the *Group Account* dialog.



Right-Click Properties

- ☑ Click the **Advanced** button to open the *Advanced* dialog:



Group Account: Advanced Dialog

- ☑ Configure the parameters on the *Advanced* dialog as follows:
 - **Options:** Allow the user to configure the Advanced Options for the Group.

Field	Remarks	Syntax
Minimum Password Size	All users assigned to this group must provide a password containing at least the minimum number of characters. If a user tries to create a password with less than the required number of characters, InduSoft Web Studio will reject the password and display a warning message. Configuring the value 0 in this field allows the users to enter passwords with any number of characters during the runtime.	Tag or Number
Password aging	Longevity (in days) of the password for all users assigned to this group. After the specified number of days, InduSoft Web Studio will force users assigned to this group to change their passwords. When the user tries to log in, the <i>Change Password</i> dialog displays automatically and the user cannot complete the log-in process until they provide a new password. The password never expires when configuring the value 0 in this field.	Tag or Number
E-signature time-out	Time-out period (in minutes) for the E-signature. Users assigned to this group must enter their UserID and password before the specified timeout period expires to execute commands requiring an electronic signature. Before the time-out time expires, the user is asked for a password only – the system automatically assumes the username logged in the last electronic signature. The system resets the time-out counter just after an electronic signature is executed.	Tag or Number
Disable E-signature	When the value in this field is TRUE (different from zero), the E-signature is disabled for the whole application. When configuring a tag in this field, you can enable/disable the E-signature dynamically during the runtime.	Tag or Number



Note:


You can check the number of hours left to expire the password (based on the Password aging setting) by using the built-in function `GetUserPWDAging()`.

Also, by default, the user must type a password different from the previous password, unless the following setting is configured in the **<ApplicationName>.APP** file:

```
[Security]
ChangePasswordMode=1
```

- **Auto Log Off:** Allows you to log-off the current user automatically.

Field	Remarks	Syntax
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.	Tag or Number
Count time after 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.	Radio-button
Count time after 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 or keyboard action) was performed by the current user.	Radio-button

 **Note:**
 When logging off the current user, the Guest user (default user name) is automatically logged on the system. Therefore, it is recommended to restrict the rights for the Guest user.

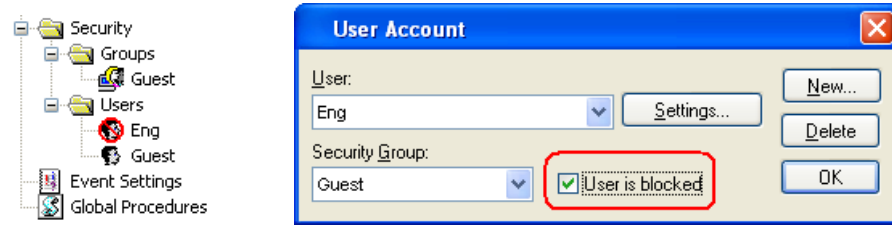
- **Account Auto Lock-up:** Allows you to block a user after attempting to log-on this user with a wrong password for a number of times.

Field	Remarks	Syntax
Enable	When checked, the users assigned to this group can be blocked automatically during the runtime.	Check-box
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, InduSoft Web Studio will lock the user account automatically.	Tag or Number
Reset counter after	Defines how long after an invalid log-on attempt, InduSoft Web Studio will wait (in minutes) until it resets the log-on attempts counter. When typing a valid password, the log-on attempts counter is reset automatically.	Tag or Number

Note:

When a user exceeds the specified number of log-on attempts, InduSoft Web Studio automatically blocks the account and will not reset the counter — even after the **Reset counter after** time expires. The System Administrator must reset the user account by disabling (*unchecking*) the **User is blocked** check-box on the *User Account* dialog or by executing the **UnblockUser ()** function.

A red circle surrounding a user name in the *Workspace* indicates that the user is blocked. In addition, the **User is blocked** box is enabled (*checked*). The following figures are examples that indicate that the *Eng* user is blocked:

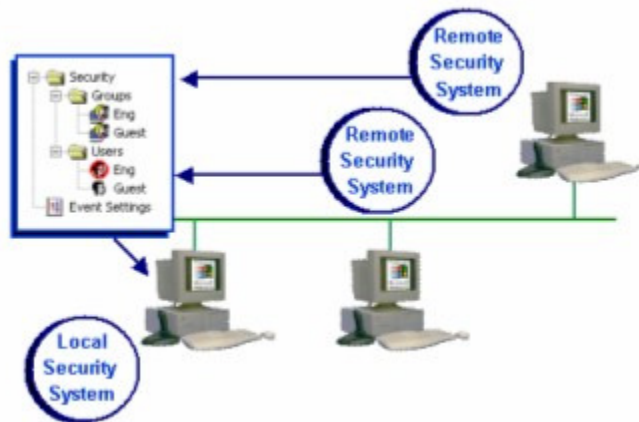


Example: User is Blocked

- ☑ When you are finished, click **OK** to apply your changes and close the *Advanced* dialog.

Remote Security System

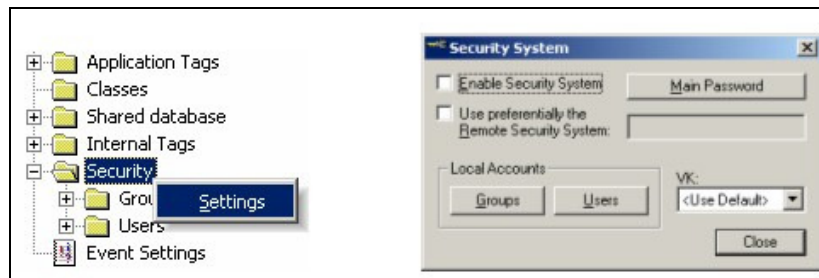
If your system applications connect through a TCP/IP link, it is now possible to designate one of your computer stations as the *Central* security system, from which other stations can use the Users and Groups definitions. The following figure illustrates this configuration:



Remote Security System

Use the following procedure to configure a central security system:

- ✓ Right-click the *Security* folder on the **Database** tab and select **Settings** from the pop-up menu to open the *Security System* dialog:



Right-Click Security Settings

- ✓ Enable (*check*) the new **Use preferentially the Remote Security System** check-box to designate a remote security system.

If the remote applications successfully connect to the security system from the *Server* station, they will use the security system configured on the *Server* station. In this case, any change implemented in the security system of the *Server* station will be assumed automatically by the remote applications. Also, the security system functions (such as **CreateUser()**, **RemoveUser()**, **ChangePassword()**, and so forth) will update the *Server* station's security system — even if the functions are executed from the remote applications. As a result, all applications on a distributed system can share the same security system settings.

If the applications cannot connect because the remote system is not running or cannot be reached, a message (similar to the following) will be logged in the *Output* window and saved in the *event* file:

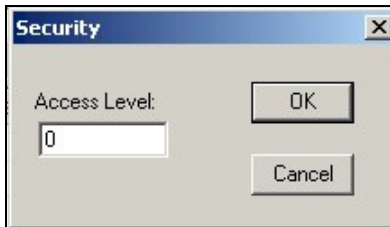
Error connecting to Remote Security Server '192.168.1.255'

In addition, the application(s) will revert to using the local computer's security settings. The remote applications attempt to connect to the *Server* station's security system only when there is an event associated with the security system (such as a user logging on). In other words, there is no polling between the remote applications and the *Server* station during runtime.

Setting the Security Access Level

You can use the **Security Level-Development** check-boxes to set a unique range of access values for each user group. You also can set a unique access range for any InduSoft Web Studio worksheet (*Alarm, Math, Recipe, Report, Scheduler, TCP Client, Trend*, and those not available on CE: *DDE Client, OPC Client, and ODBC*).

If you click on any part of the worksheet body, you can activate the **Edit** → **Access Level** option from the main menu bar, which opens the *Security* dialog so you can assign an **Access Level** to that worksheet.



Security Dialog

Assigning an access level to a worksheet means that a user would have to have an access level that falls within the specified Security Level- Development range to edit that worksheet.

For example, **UserA** of **GroupA** has a security access level range of *0-10* and **UserB** of **GroupB** has a security access level range of *5-15*. To continue the example:

- Math Worksheet 001 has Access Level = 1
- Math Worksheet 002 has Access Level = 7
- Math Worksheet 003 has Access Level = 12
- Math Worksheet 004 has Access Level = 20

Consequently,

- Only **UserA** can access Math Worksheet 001
- Both users can access Math Worksheet 002
- Only **UserB** can access Math Worksheet 003
- Neither user can access Math Worksheet 004

Logging On/Off

After defining the user names and passwords, you use the *Logon* utility (**Project** → **Logon**) to log users on and off.

Alternatively, you can use the to the InduSoft Web Studio Scripting Language activation functions **LOGON** () and **LOGOFF** () to log users on or off.



Log On Dialog

Use the features of this dialog as follows:

- **User Name** field: Enter the user name to log in.
- **Password** field: Enter the user password.
- **Log Off** button: Click to log off the current user.

Note:

When a *Logoff* is executed, the *Guest* user is automatically logged on.

Chapter 12: Testing and Debugging Your Application


This chapter explains how to use different IWS tools to test and debug your project applications.

Testing Your Application

Use the following procedures to test your application:


- **Project > Test Display:** Activates the test display mode, which allows you to configure the application while viewing graphical dynamics online in the development environment. The test display mode does not enable you to use the **Command**, input **Text I/O** dynamics, or execute worksheets.

 **Note:**

Using the **Test Display** menu option is the same as using the  button on the *Execution Control* toolbar.


- **Project > Stop display test:** Stops the test display mode.

 **Note:**

Using the **Stop display test** menu option is the same as using the  button on the *Execution Control* toolbar.

- **Project > Run Application:** Launches the runtime modules specified as **Automatic** on the *Project Status* dialog (**Execution Tasks** tab).
 - When you start the *Viewer* module, it opens the screen(s) currently being edited.
 - If you do not specify any **Automatic** tasks, InduSoft Web Studio will launch the *Viewer* and *BGTask* tasks automatically when you execute **Run Application**.
 - If you are not currently editing screens in the development environment, the *Viewer* module opens the screen specified in the **Startup screen** field on the **Runtime Desktop** tab (*Project Settings* dialog).

 **Note:**


Using the **Run Application** menu option is the same as using the  button on the *Execution Control* toolbar.

 **Caution:**

Run Application affects the application from the target station (configured in the *Execution Environment* dialog). Be sure you know which target station is configured (local or remote) before executing the **Run Application** command.

- **Project > Stop Application:** Stops all runtime tasks.

Caution:
Stop Application affects the application from the target station (configured in the *Execution Environment* dialog). Be sure you know which target station is configured (local or remote) before executing the **Stop Application** command.

Note:
 Using the **Stop Application** menu option is the same as using the  button on the *Execution Control* toolbar.

Debugging Applications from the Database Spy

- Notes:**
- The *Output Window* is *dockable*, which means you can move it to another location in the development environment. Simply click once on the titlebar and drag it to a new location. Release the mouse button to attach or *dock* the window to its new location.
 - For a description of the *Database Spy* window and its components, review “Using the Database Spy” in *Chapter 3: Working with the IWS Development Environment*.

The *Database Spy* is a very useful debugging tool because it enables you to:

- Force values to database tags and monitor the results
- Execute different functions or equations



Name	Value	Quality	Continuous
a	1	GOOD	Yes
b	1	GOOD	Yes
c	1	GOOD	Yes
d	1	GOOD	Yes

Sample Database Spy Window

The *Database Spy* contains four **DB** tabs where you can enter groups of tags, functions, and/or expressions you want to test and debug.

Each **DB** tab contains a spreadsheet with the following components

- Name:** Displays the tagname and/or function being evaluated on this line.
- Value:** Displays returned values and equation results
- Quality:** Displays a quality evaluation (**Good** or **Bad**) of the tag or function source
- Continuous:** Displays whether IWS is continuously re-evaluating the tag, function, or equation.

Creating different groups enables you to manage multiple testing/debugging tasks. For example, you can create one set of tags on **DB1** to test a recipe function, and create another group of tags on **DB2** to test a trending function.

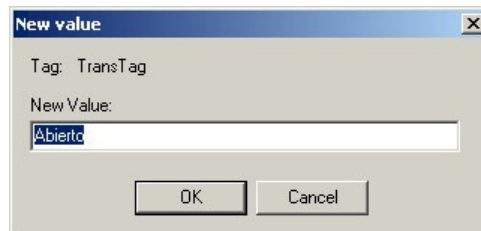
To use test your application using the *Database Spy*, use the following process to configure the spreadsheet:

- ☑ Double-click on an empty line in the spreadsheet to open the *Object Finder* dialog. You use the *Object Finder* dialog to add tags, expressions, or functions to the data collection process. You can use the *New Tag* dialog to force values to test a tag or collect data for a tag.
- ☑ When you right-click on a line in the *Database Spy* window, a pop-up menu displays:



Right-click to Display Options

- **New Value:** Use this option to change the current value of the selected tag. Selecting this option opens the *New value* dialog. Type a value into the **New Value** text box and then click **OK**.



New Value Dialog

- **Continuous:** Use this option to enable or disable whether IWS continuously re-evaluates the selected expression or tag.
- **Toggle:** Use this option to toggle the tag value between 0 and 1, or to reinitiate (renew) a function or expression
- **Insert Tag or Expression:** Use this option to insert a new tag or expression.
- **Remove Selection:** Use this option to delete the selected expression.
- **Hide Window:** Select to close (hide) the *Database Spy* window.

To reopen the window, you can:

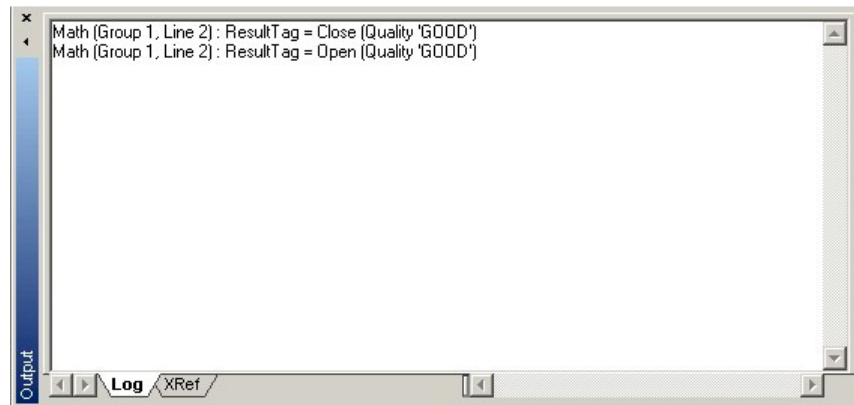
- * Click the **Toggle Database Spy Window** icon on the *Standard* toolbar
- * Press **Alt+2** from the keyboard
- * Select **View** → **Toolbars** → **Database Spy** to re-open the window.

If you are having trouble with any function, copy the function and use the *Database Spy* trigger to test and debug the function instead of running, tweaking, and then running the function again and again in the development environment.

Note:
Some functions, such as `report ()`, `recipe ()`, and `math ()`, must be executed as *Background Tasks* during runtime to work properly.

Debugging Applications from the Output Window

Use the *Output (LogWin)* window (located next to the *Database Spy*) to view debugging messages provided by IWS. (**Note:** You can double-click on a line item to open that worksheet or screen.)



Sample Output Window

For example, you can check the serial communication with a PLC.

Notes:

- The *Output* window is *dockable*, which means you can move it to another location in the development environment. Simply click once on the titlebar and drag it to a new location. Release the mouse button to attach or *dock* the window to its new location.
- See *Chapter 3: Working with the IWS Development Environment* for a detailed description of the *Output* window.

The *Output* window contains the following tabs:

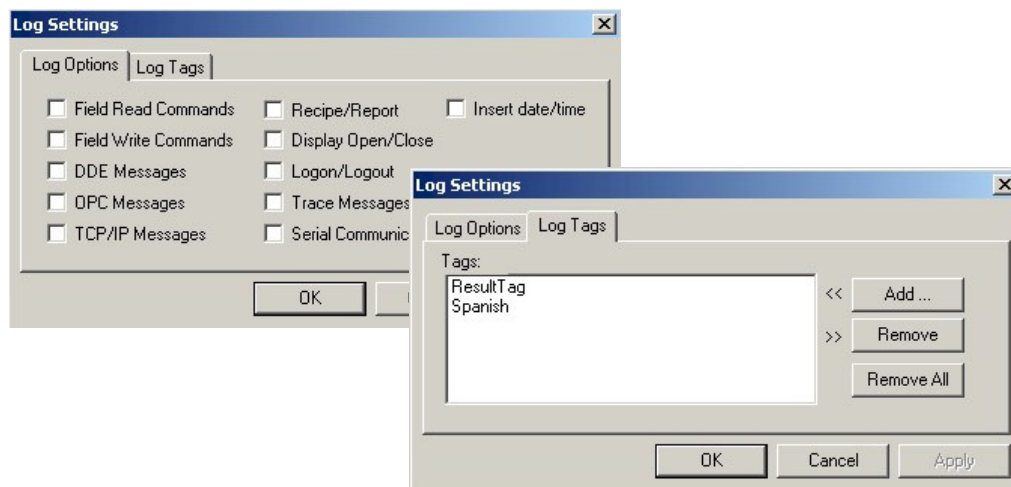
- **Log** tab: Similar in function to the *LogWin* module, but provides limited functionality to reduce time and memory usage within the dev environment. It uses the first-in, first-out (FIFO) principle to manage space.

Right-click in the *Output* window and a pop-up menu displays:



Right-click to Display Options

- **Settings:** Opens the *Log Settings* dialog where you specify options and tags you want IWS to look for and log for debugging purposes.



Log Settings Dialog

Use the two tabs on this dialog to specify the kind of information you want IWS to log and display in *Output* window.

- * **Log Options** tab: Enable (☑) the check-boxes to log the following events:

Field Read and Field Write Commands
 DDE, OPC, TCP/IP, and Trace Messages
 Recipes/Reports
 Opening and closing the display
 Log-ons and Log-outs
 Serial communication
 Inserting date/time

- * **Log Tags** tab: Use the **Add** button to open the *Object Finder* dialog, which you can use to create a list of tags for IWS to monitor. (Steps for using the *Object Finder* dialog are provided in “Debugging Applications from the Database Spy” on page 12–2.) Use the **Remove** or **Remove All** buttons to delete tag names from this list.

Whenever a specified action occurs or a specified tag changes value, IWS will log the results in the *Output* window—along with the driver, modules, and so forth that initiated the change.

- **Pause:** Select (*enable*) to stop the data collection process temporarily, and re-select (*disable*) to resume data collection.
- **Hide Window:** Select to close (*hide*) the *Output* window.

To reopen the window, you can:

- * Click the **Toggle Output Window** icon on the *Standard* toolbar.
- * Press **Alt+1** from the keyboard.
- * Select **View** → **Toolbars** → **Output** to re-open the window.
- **Cut, Copy, Delete, Select All:** Become active when you select an entry in the *Output* window.

To print the results from the *Output* window, select the text you want to print. Right-click and select **Copy** from the pop-up menu. You can then paste the results into *Notepad* (or another application) and print the report from there.

If you established settings (using the *Log Settings* dialog) in the development environment, those settings will transfer to your CE station, along with the application, so you can create a log from the CE station. You can change the settings from your CE station, or change the settings from *Development* and resend them to your CE station.

- **X-Ref tab:** Use the **Object Finder** field to get a tag, and to find every place in the application 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.

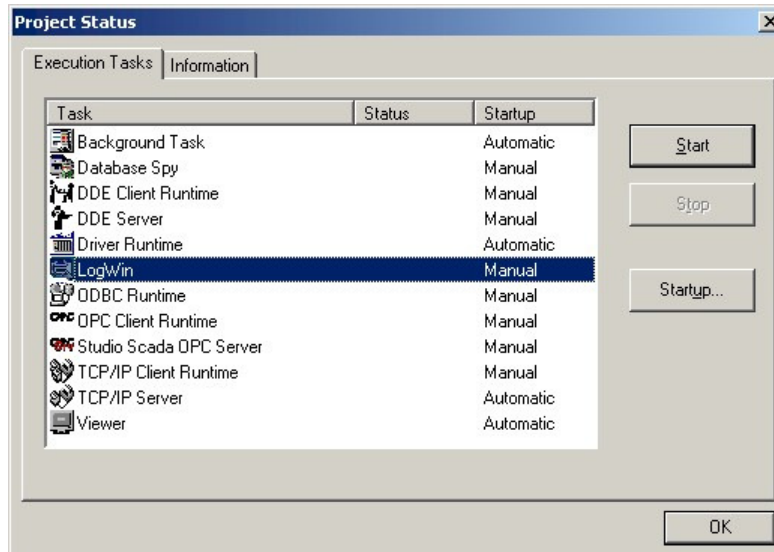
 **Notes:**

- The **X-Ref** tab does not work for functions, only tags, but does allow you to look for indexes.
- See also “Using the Object Finder,” in *Chapter 3: Working with the IWS Development Environment*.

Using the LogWin Module (NT and CE)

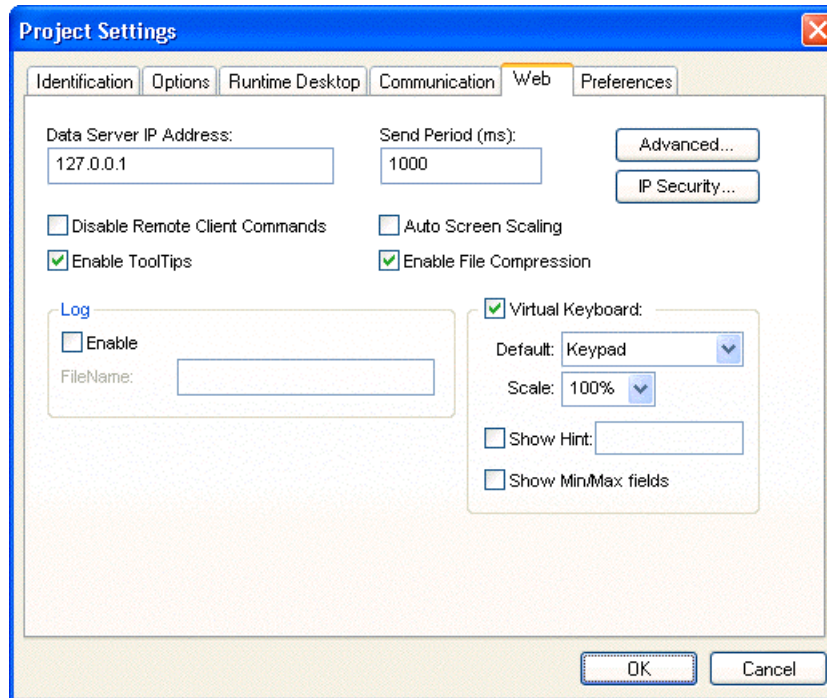
This module provides a continuous record of activities and tags for debugging over long periods of time. It creates a file into which you can dump the data collection results, and this file continues to grow in size until you stop the logging (data collection) process. Use the *LogWin* module (local and remote) to record DDE, OPC, and TCP/IP transactions, activate modules, trace tags, and so forth.

To initiate *LogWin*, go to **Project** → **Status** locally, or select **LogWin** from the **Tools** menu on the CE box.



Initiating LogWin

To debug the Web client, check (*enable*) **Log** by selecting **Project** → **Settings** from the menu bar, and then selecting the **Web** tab. Check the **Enable** check-box and type in the *path+filename* in the **FileName** option box. When finished, select **OK**.



Debugging the Web Client





Using Remote Tools

InduSoft Web Studio offers on-line, remote application management and configuration (download/upload, commands, system and network diagnostics, and debugging). You can configure and debug applications remotely using a TCP/IP link.

You can use the *Execution Control* toolbar to execute and manage an application locally or from a remote location.





Execution Control Toolbar

- **Test Display** button (): Allows you to run in test display mode, which allows you to configure an application while viewing graphical dynamics on-line in the development environment.
- **Stop Test Display** button (): Allows you to stop running in test display mode.
- **Run Application** button (): Allows you to launch any runtime modules specified as Automatic.
- **Stop Application** button (): Allows you to stop all runtime tasks.

⚠ Caution:

Stop Application affects the application from the target station (configured from the *Execution Environment* dialog). Be sure you know which target station is configured (local or remote) before executing **Stop Application**.

- **Send project to target** button (): Allows you to send the application to the remote target station.
- **Execution Environment** button (): Allows you to manage remote workstations.
With this button, you can:
 - Connect in a remote station
 - Send/update system files (CE only)
 - Send/update application files
 - Send files
 - Import application files
 - Import CE logs
 - License remotely (CE only)

📌 Notes:

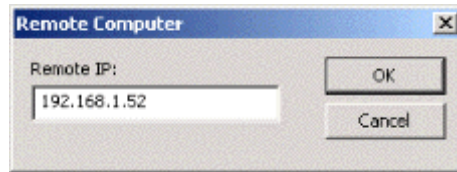
- For a description of the *Execution Control* toolbar, see *Chapter 7: Configuring Screens and Graphics*.
- For a description of configuring the *Execution Environment*, see *Chapter 14: Managing Applications Remotely*.

Using Remote Database Spy

You can use the Remote Database Spy tool (located in the Tools menu) to monitor the Database Spy of an IWS application 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. From the main menu bar, select **Tools > Remote Database Spy**. The *Remote Computer* dialog is displayed.
2. Enter the IP address of the remote computer, as shown below.

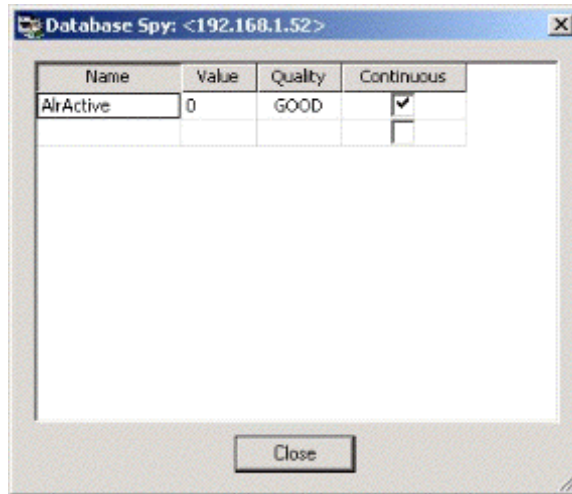


Entering a Remote IP

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.

Using Remote LogWin

You can use the Remote LogWin tool (located in the Tools menu) to monitor the Output log (LogWin) of an IWS application 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 Database Spy tool:

1. From the main menu bar, select **Tools > Remote LogWin**. The *Remote Computer* dialog is displayed.
2. Enter the IP address of the remote computer, as shown below.



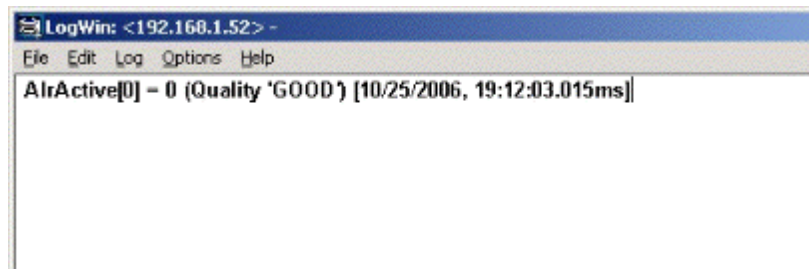
Entering a Remote IP



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.

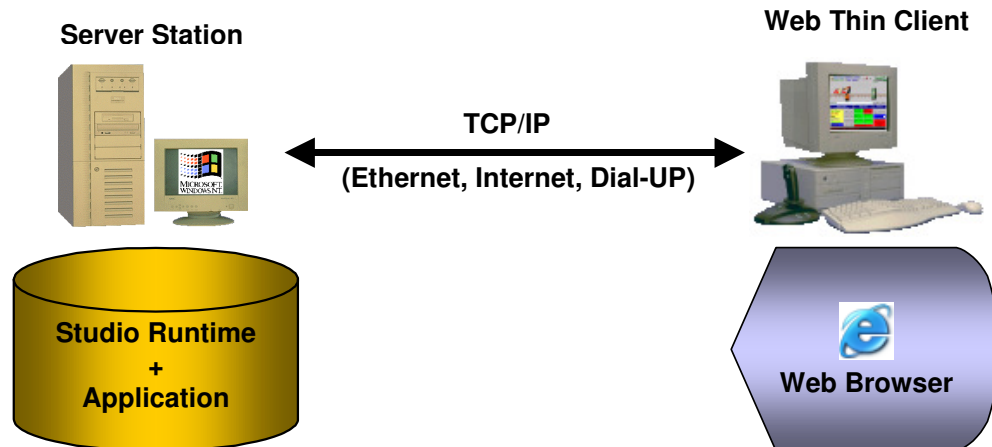


Note:

For a detailed description of managing applications remotely, see *Chapter 14: Managing Applications Remotely*.

Chapter 13: Configuring a Web Solution

IWS allows you to create screens that can be visualized in a remote station using a regular Web browser (e.g. Internet Explorer). The station where the user can visualize the graphical interface (screens) on the Web browser is herein called the Web Thin Client.



IWS is installed on the server station only. Also, the application (screen files, tags database, configuration worksheets, and so forth) is stored on the server only. In other words, you do not need to install IWS or the application on the Web Thin Client station(s). This solution provides a high level of flexibility because any computer physically linked to the server station (TCP/IP link) can access the graphical screen and online/history data from the server, without installing IWS or the application on the Web Thin Client station(s). Any computer or device (e.g. PDAs powered with Windows CE) running Internet Explorer Web browser v6.0 (or higher) can be a Web Thin Client for an IWS application. Moreover, IWS provides a sophisticated *Security System* to prohibit unauthorized access to the application.

Note:

The maximum number of Web 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 Web Thin Client stations.

From the Web Thin Client station, you are able not only to visualize data from the server but also to change set points and/or send commands to the server. When configuring the application, you can (optionally) disable all commands from the Web Thin Client to the server station. In this case, the Web Thin Client stations can read data from but cannot send any data to the server.

- All background tasks (Math, Scheduler and so forth) and communication tasks (Driver, OPC, DDE and so forth) are executed on the server station only. The Web Thin Client is able to load the graphical interface configured on the server (screens with objects and dynamics) and display the online values from the tags configured in the server, as well as history data (Alarm, Events and Trend history data).

Configuring the Application for Different Architectures

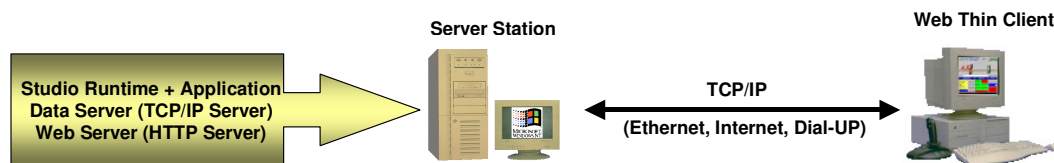
This section describes the typical architectures applied for Web-based solutions and provides examples of how to configure the IWS application for each architecture.

The definitions of some of the terms used in this section are described below:

- **Web server station:** Computer or device running a Web server. The files from the `\Web` sub-folder of the application must be stored in this computer.
- **Data server station:** Computer or device running IWS. The IWS application must be stored in this computer.

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.

Architecture 1: Web server and Web Thin Clients in the same network



This is the most common architecture as well as the simplest to configure. In this architecture, both the Web server (e.g. IIS) and the data server (TCP/IP server module from IWS) are running in the same computer (server station). The Web Thin Client connects to the Web server on the server station to download the HTML screen file. Then it connects to the data server to exchange data with IWS.

Since both the Web Thin Client and the server station are connected to the same network, the Web Thin Client can access the server station directly through its IP address (or host name).

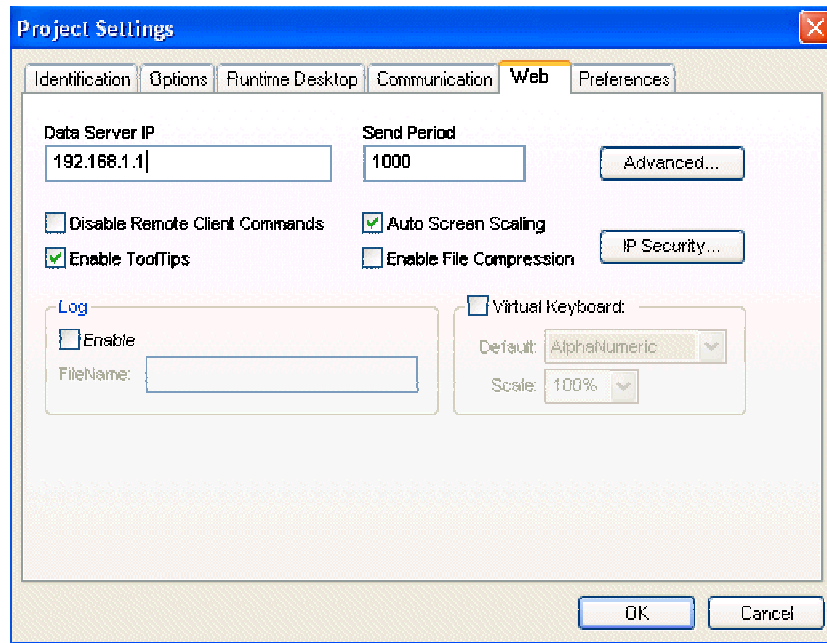
Configuration example:

This example is based on the following premises:

- IP address of the server station on the network: `192.168.1.1`
- Home directory of the Web server (HTTP server) on the server station: `\Web` sub-folder of the application

You must type the following address on the remote Web browser to access a screen (e.g. `myscreen`) from the server: `http://192.168.1.1/myscreen.html`

The **Project Settings > Web** interface must be configured as follows:

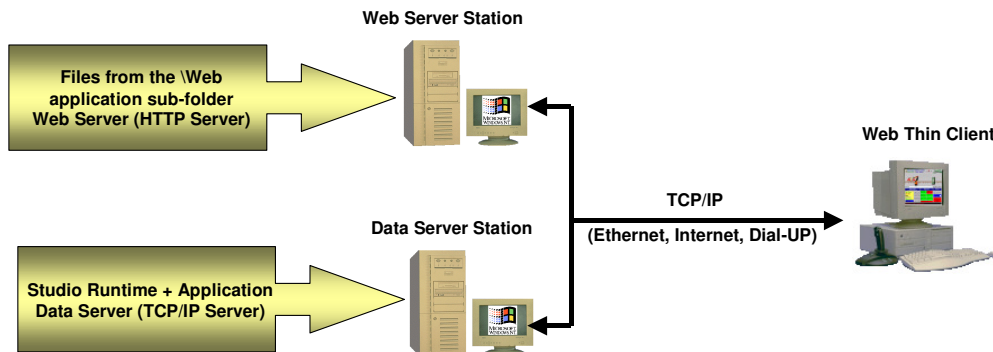


Project Settings > Web Interface

Note:

This architecture is adopted when the server station and the Web Thin Client(s) are directly connected to the same intranet network or via a dial-up connection. If the server is connected to the internet, you must assign a Fix IP address to the server on the internet, and the application must be running in this computer. Consult your ISP provider for further information about how to get a Fix IP address for your server computer on the internet.

Architecture 2: Web server and Web Thin Clients in the same network; Web server and data server in different stations



This architecture is especially useful when you want to isolate the Web server (HTTP server) from the data server (TCP/IP server module from IWS). The common reasons to adopt this architecture are:

- **Allows you to use a standard Web server station shared by several applications in the company.** Some companies use one computer as the standard Web server for all web-based applications. For physical or safety reasons, you may not want to run the actual application in this computer (e.g. It is far in distance from the control room). Therefore, you can run IWS and the application in another computer (data server station) and copy just the web files of the application (files from the **\Web** sub-folder of the application) to the Web server station.
- **Hosts the Web pages on a Web site.** If you want to store the Web pages on a Web site (e.g. **www.mycompany.com**), you can upload just the web files of the application (files from the **\Web** sub-folder of the application) to the Web site and use it as the Web server station. The application (and IWS) keeps running in another computer, physically connected to the internet.
- **Enables you to use a Linux-based Web server (e.g. Apache).** You do not have to install IWS on the Web server station; therefore, if you want to use a Web server for Linux, you can run it on the Web server station and run IWS on the data server station.
- **Hides the IP address (or host name) of the data server station from the users on the Web Thin Client station.** In this architecture, the user has to type the URL of the Web server station on the Web browser (not the IP address of the data server station). This may be desirable for safety reasons.

 **Note:**

It is not necessary for IWS to be installed on the Web server station. The following components must be available on the Web server station:

- Web server (e.g. IIS from Microsoft)
- Files from the **\Web** sub-folder of the application

⇒ **Tip:**

When you have many data server applications in your project, you can use this architecture to share the same Web server for all applications. For example, you can link the Web server to the data servers via a switch. This will keep the traffic in the data servers network from increasing while the Web Thin Clients are downloading files from the Web server station.

In this architecture, both the Web server (e.g. IIS) and the data server (TCP/IP server module from IWS) are running on different computers. The Web Thin Client connects to the Web server station to download the HTML screen file. Then it connects to the data server station to exchange data with IWS.

Since all Web Thin Client, Web server and data server stations are connected to the same network, the Web Thin Client can access the server stations directly through their IP addresses (or host names).

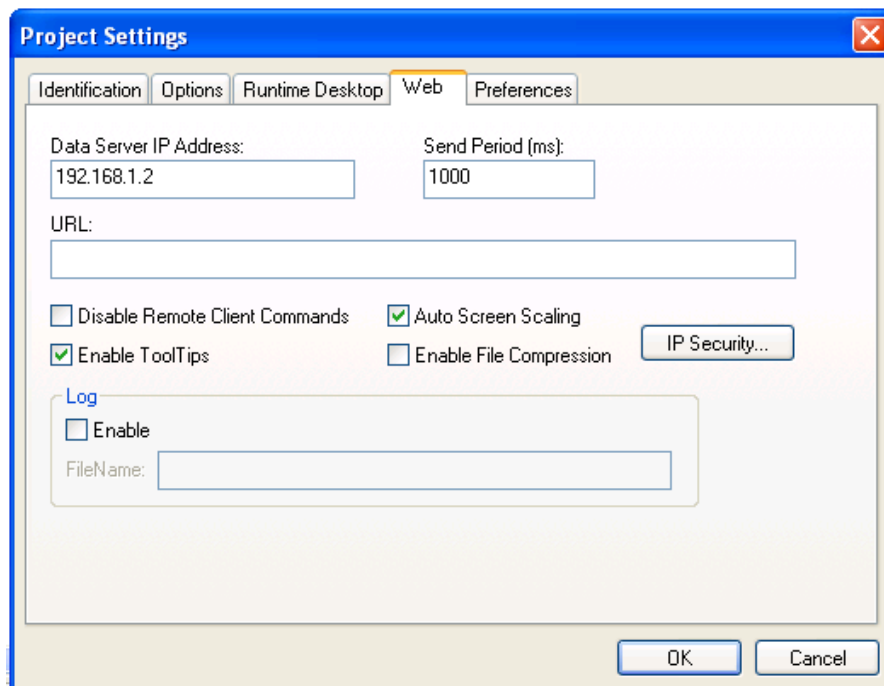
Configuration Example:

This example is based on the following premises:

- IP address of the Web server station on the network: **192.168.1.1**
- IP address of the data server station on the network: **192.168.1.2**
- Home directory of the Web server (HTTP server) on the server station: **\Web** sub-folder of the application

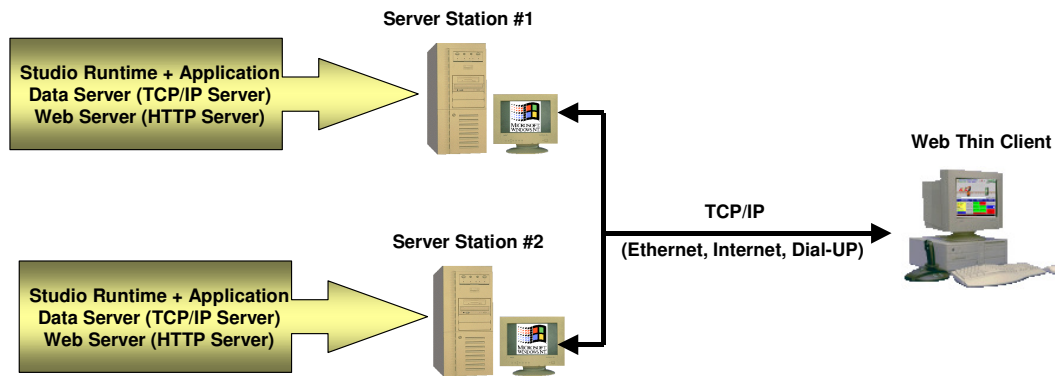
You must type the following address on the remote Web browser to access a screen (e.g. **myscreen**) from the server: **http://192.168.1.1/myscreen.html**

You must configure the **Project Settings > Web** interface as follows:



Project Settings > Web Interface

Architecture 3: Redundant servers and Web Thin Client stations in the same network



This architecture is similar to Architecture 1. But in this architecture, two server stations with the same files run the same application in redundancy. The Web Thin Client connects to the server specified by the user in the address field of the Web browser. If this server goes down for any reason (e.g. power failure), the Web Thin Client switches to the other server station automatically.

This architecture is recommended when it is necessary a high level of availability for the Web Thin Client stations. In other words, even if one Server Station goes down, the Web Thin Client stations are able to get data from the other Server Station.

Configuration Example: This example is based on the following premises:

- IP address of server station #1 on the network: **192.168.1.1**
- IP address of server station #2 on the network: **192.168.1.2**
- Home directory of the Web server (HTTP server) on server station #1: **\Web** sub-folder of the application stored on server station #1.
- Home directory of the Web server (HTTP server) on server station #2: **\Web** sub-folder of the application stored on server station #2.

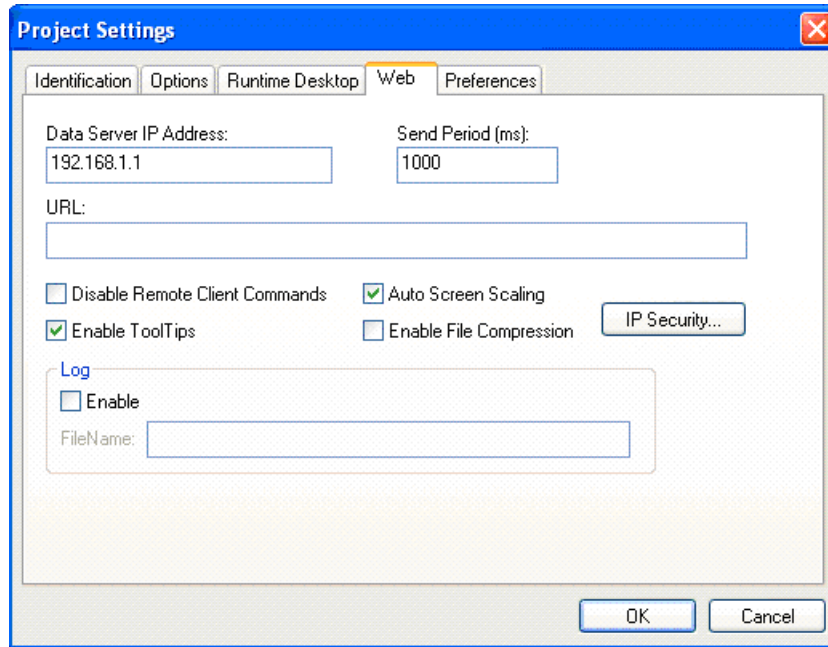
The user must type the following address on the remote Web browser to access a screen (e.g. **myscreen**) from server station #1: **http://192.168.1.1/myscreen.html**

The user must type the following address on the remote Web browser to access a screen (e.g. **myscreen**) from server station #2: **http://192.168.1.2/myscreen.html**

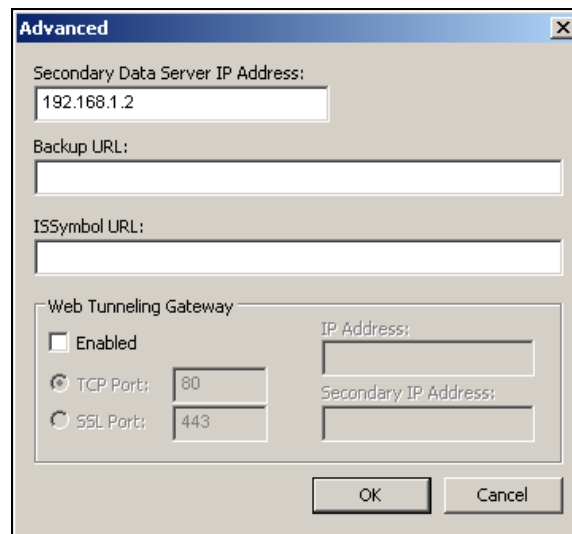
The **Project Settings > Web** interface must be configured as follows:

⇒ **Tip:**

It is possible to configure two data servers that share the same Web server. Just apply the concepts described in both *Architectures 2* and *3*.

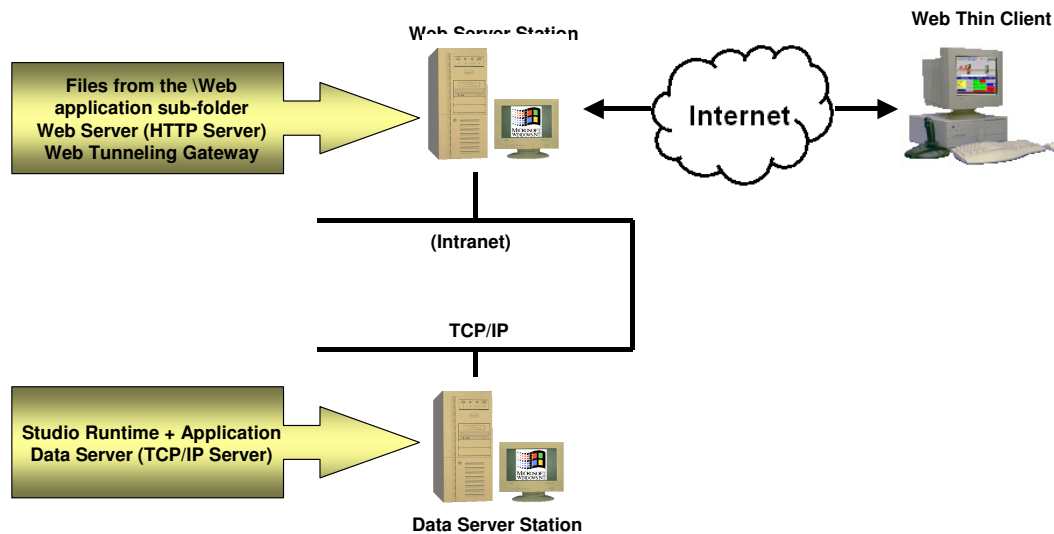


Project Settings > Web Interface



Advanced Dialog Window

Architecture 4: Web server and Web Thin Clients in different networks



This architecture is common when the Web Thin Clients are connected to the server via the internet. Usually, the data server computer (computer where the IWS is running) is not directly connected to the internet. In this case, the data server computer does not have an IP address on the internet. Therefore, it cannot be connected directly through the internet. The Web Tunneling Gateway (WTG), developed by InduSoft, provides the routing capabilities to solve this problem.

The WTG must be installed in the computer with the Fix IP Address on the internet (consult your ISP provider for further information about how to get a Fix IP Address for your computer on the internet). This computer must have the Microsoft IIS Web server installed and running. The WTG is an ISAPI extension for IIS.

Follow the procedure below to install the WTG on the Web server computer:

- Copy the **WebGtw.exe** file from the **\BIN** sub-folder of IWS into any directory of the Web server computer.
- Execute the **WebGtw.exe** file on the Web server computer.

The WTG works as a router between the Web Thin Clients (connected to the internet) and the data server computer (connected to the intranet). The same WTG can route information for more than one data server simultaneously.

Note:

The computer directly connected to the internet (where the WTG is running) is the Web server for the application; therefore, the files from the **\Web** sub-folder of the application must be stored in this computer.

Configuration example:

This example is based on the following premises:

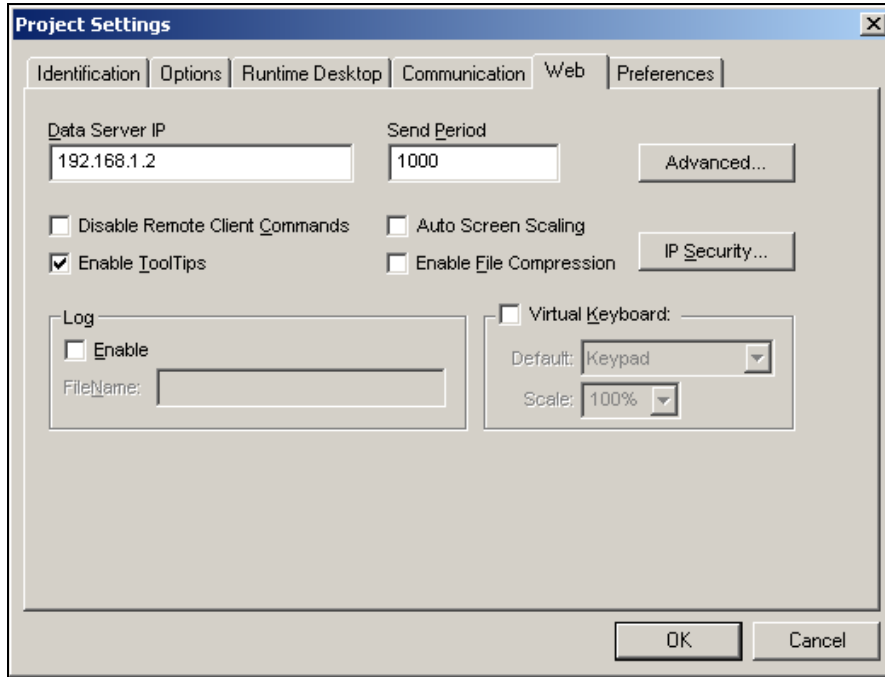
- IP address of the Web server station (internet): **200.0.0.1**
- IP address of the Web server station (intranet): **192.168.1.1**

- IP address of the data server station on the intranet: **192.168.1.2**
- Home directory of the Web server (HTTP server) on the Web server station: **\Web** sub-folder of the application stored on the Web server station.

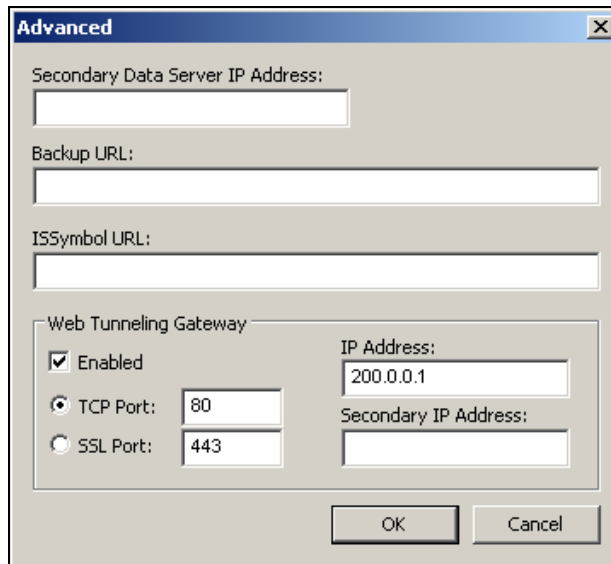
The user must type the following address on the remote Web browser to access a screen (e.g. **myscreen**) from the Web Server Station:

http://200.0.0.1/myscreen.html

The **Project Settings > Web** interface must be configured as follows:



Project Settings > Web Interface



Advanced Dialog Window

 **Note:**

If your Web server is able to provide files via HTTPS (SSL – Secure Socket Layer), you can select this option on the *Advanced* dialog window from the **Project Settings > Web** interface.

 **Tip:**

The WTG encapsulates the protocol implemented by the TCP/IP server module of IWS into HTTP (or HTTPS when the SSL option is selected). By this way, it is not necessary to open an additional TCP Port on the firewall between the Web server and the Web Thin Clients. The same port used by the Web server (HTTP or HTTPS) is used by IWS data protocol.

Installing and Registering the ISSymbol Control Layer

ISSymbol is a component designed by InduSoft that is able to display the screens created with IWS on the Web browser and exchange data (tag values and history data) with the TCP/IP server module of IWS. On the Web 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 IWS application 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 application 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 > Web > Advanced** dialog. The Web browser is not able to display the screens from the IWS application if the ISSymbol control is not properly registered in the Web 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 Web Thin Client station. Check your Web browser's documentation about security settings if you have questions about how to configure these settings.

INSTALLING THE ISSYMBOL CONTROL MANUALLY

You can also install the ISSymbol control manually in the Web Thin Client station. The procedure to install ISSymbol in each operating system is described below.

Windows NT/2000/XP/Vista:

1. Copy the following files...
 - \InduSoft Web Studio v6.1\Bin\ISSymbolReg.exe
 - \InduSoft Web Studio v6.1\Bin\ISSymbolVM.cab...and paste them into any directory of the Web Thin Client station. Make sure that both files are stored in the same directory.
2. Run ISSymbolReg.exe to register the ISSymbol control in the Web Thin Client station.

Windows CE:

1. Copy the following files...
 - \InduSoft Web Studio v6.1\Redist\<OS Version>\<Processor Type>\Bin\ISSymbolCE.ocx
 - \InduSoft Web Studio v6.1\Redist\<OS Version>\<Processor Type>\Bin\IndHTTP.dll...and paste them into any directory of the Windows CE device. Make sure that both files are stored in the same directory.
2. Execute the following command from the Windows CE command prompt:

```
regsvrce.exe "\<ISSymbolPath>\ISSymbolCE.ocx"
```

For example:

```
regsvrce.exe "\Storage Card\ISSymbolCE.ocx"
```

3. Save the registry settings to keep ISSymbolCE.ocx registered when you reboot the Windows CE device.

Windows CE PocketPC:

2. Copy the following files...

- \InduSoft Web Studio v6.1\Redist*<OS Version>*\<Processor Type>\Bin\IndHTTP.dll
- \InduSoft Web Studio v6.1\Redist*<OS Version>*\<Processor Type>\Bin\ISSymbolCE.ocx
- \InduSoft Web Studio v6.1\Redist*<OS Version>*\<Processor Type>\Bin\RegSvrCE.exe

...and paste them into any directory of the PocketPC device. Make sure that both files are stored in the same directory.

3. Execute the RegSvrCE.exe program on the PocketPC device. To register ISSymbolCE.ocx, do the following:
 - Select the \<ISSymbol Path>\ISSymbolCE.ocx file.
 - Select the **Register** option.
 - Click **OK**.

 **Note:**

Internet Explorer is not able to download ActiveX controls automatically from Windows CE and Windows CE PocketPC. Therefore, before using Windows CE devices as Web Thin Clients, you must register the ISSymbolCE.ocx control manually.

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 Web 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 application (settings saved in the HTML page). Since the ISSymbol control is properly registered in the Web 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 with the **Project Settings > Web** dialog window. This setting is saved in the HTML page.
4. ISSymbol prompts a window on the Web 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/or *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 application.

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 Web Thin Client (and vice-versa). Therefore, there is no pooling between the Web 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 Web 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 application running on the server to the Web Thin Client computer(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.

Testing the Application

Before you begin, use this short checklist to verify that you are ready to test your application in a Web Browser:

- Verify that the ActiveX component **ISSymbol.ocx** is installed and registered.
- Verify that the IIS Web Server is pointing to the *Home* directory folder and running **or** that a copy of the IWS **NTWebServer.exe** is running in the *Home* directory folder.
- Verify that the TCP/IP Server is running.

VERIFY THAT APPLICATION TAGS ARE SET TO SERVER IF YOU WANT TO VIEW THE RESULTS OF THOSE TAGS IN THE BROWSER. (REVIEW “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 **<Application Name>.app** file to change the following setting:

```
[Options]
BooleanTrueAboveZero=<value>
```

If **BooleanTrueAboveZero** is set to the default 0, then the application will behave as described above. If **BooleanTrueAboveZero** is set to 1, then the application 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:

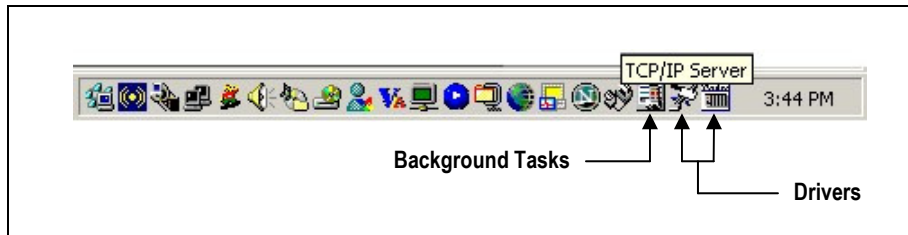
This is a global runtime setting. If you only want to change how certain tags are handled, then you should not change this setting.

Sharing Tags with a Web Thin Client” in *Chapter 5: Working with Tags.*)

To test your Web-based application, use the following steps:

- Click on **Run application** button  (on the IWS main menu bar) to execute the application locally on your Server station.

Check your Windows *Task Bar* and verify that the *TCP/IP Server* and *Background Tasks* are running. You may also note that one or more drivers are running.



Verify that TCP/IP and Background Tasks are Running

- ✓ After the application screen opens, open your Internet Browser (such as Internet Explorer or Netscape) and type the URL address (being sure to include the correct *Home* directory) to open the `<screen name>.html` screen from the Server station (for example, `http://192.168.1.1/myscreen.html`).
- ✓ When the *Log On* dialog displays in the Browser, type your log-on name into the **User Name** text box, your password into the **Password** text box, and then click **OK** to open the `<screen name>.html` screen.

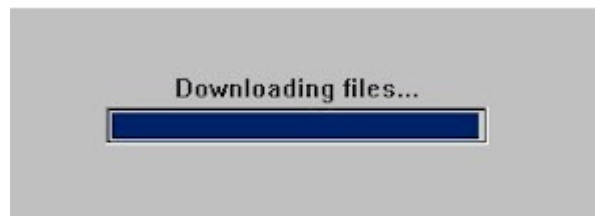


Logging On

Notes:

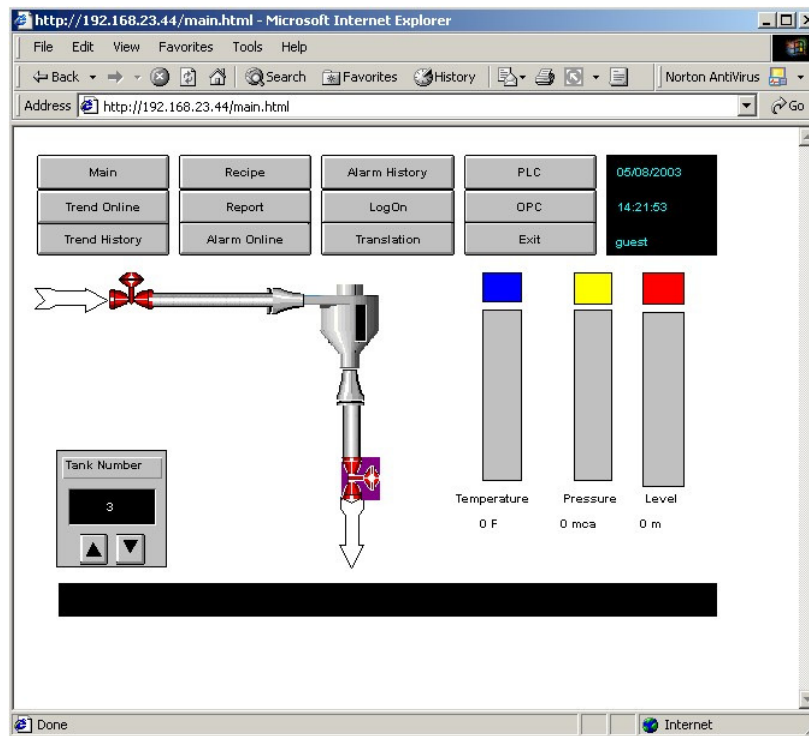
- If you have not defined users in the application's Security folder, the IWS default User Name is **Guest** with no password.
- If you disabled any security requirements for your application, the *Log On* dialog will not display.
- When the *Log On* dialog displays, it is an indication that your TCP/IP Server is running successfully. If a problem exists with the TCP/IP Server, an error message will display instead.

After you close the *Log On* dialog, the following screen displays in the Browser window to indicate the download status.



Reporting Download Status

When the download is complete, your application screen will display. At this point, the Web Thin Client should be reading information from the screen tags (for example, the clock should indicate the passage of seconds).



IWS Application Running in Browser

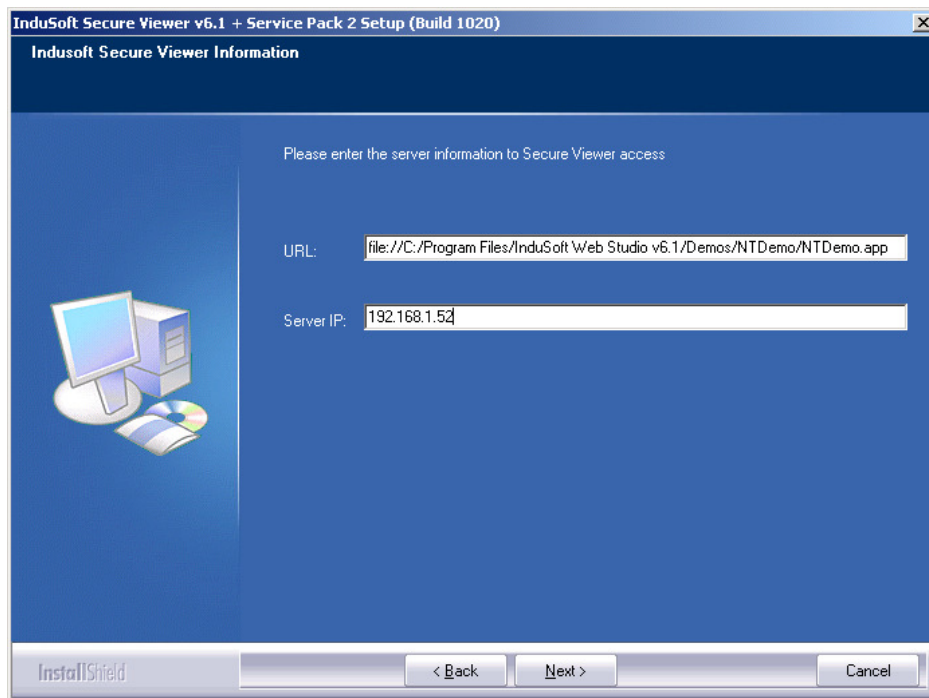
Installing Secure Viewer as Alternative to Web Browser

In some situations, you may not want to have a full-featured Web browser installed and running on a plant floor station. You can install the InduSoft Secure Viewer (**Viewer.exe**) as an alternative to the browser. This program provides the same functionality as a Web Thin Client without the unwanted features of a full browser. When you start the Secure Viewer, it immediately connects to the specified Web and Data Servers and loads the IWS application. No user direction is required.

INSTALLING THE SECURE VIEWER ON A WINDOWS PC

To install the Secure Viewer on a Windows 2000/XP/Vista computer:

- ☑ Request the Secure Viewer installer from your software vendor, save it to the designated computer, and start it.
- ☑ Follow the instructions of the installation wizard. There are only two settings that *must* be configured during installation:



Installing the InduSoft Secure Viewer

- **URL:** Enter the URL or filepath of the application file (***.app**) on the Web Server.
- **Server IP:** Enter the IP address or hostname of the TCP/IP Server.
- ☑ Finish the installation and click **Finish** to close the installer.

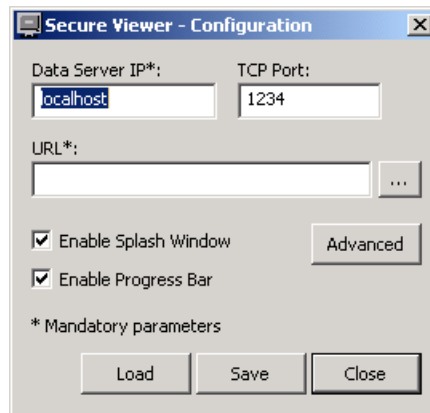
The installation wizard automatically installs and registers the Web Thin Client ActiveX component (ISSymbol.ocx), so if you have correctly configured the **URL** and **Server IP** settings, then the Secure Viewer should be ready to go.

⇒ **Tip:** For additional security, add the Secure Viewer to the Startup directory in Windows.

CHANGING THE SECURE VIEWER CONFIGURATION

The configuration of the Secure Viewer is saved in the **viewer.ini** file, which should be in the same directory as **Viewer.exe**.

There are two ways to change the configuration of the Secure Viewer after it has been installed. First, you can run the Viewer configuration utility (**ViewerCfg.exe**) that was also installed.

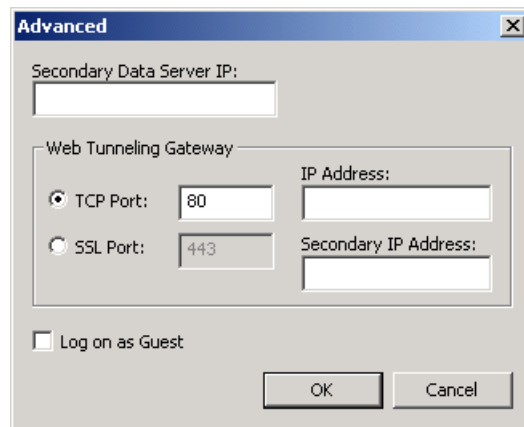


Secure Viewer Configuration Utility

The configuration utility provides the following options:

- **Load** button: Click to load a **viewer.ini** file into the utility for editing.
- **Save** button: Click to save your changes to the **viewer.ini** file.
- **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 application 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 application file.

- **Advanced** button: Click to access additional configuration options:



Secure Viewer Configuration Utility — Advanced Setting

- **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.
- **Web Tunneling Gateway**: If you have configured a Web Tunneling Gateway to bridge your intranet to the Internet (see “Architecture 4: Web server and Web Thin Clients in different networks” on page 13–8), then enter the address(es) for the gateway here.
- **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.

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 application file (*.app)
proxyip = // Proxy Address
proxyp = // Proxy Port
ceemul = // Enable (1) or disable (0) CEView emulation
```

⇒ **Tip:** Remember that the **viewer.ini** file must be saved in the same directory as the Secure Viewer program (**Viewer.exe**).

INSTALLING THE SECURE VIEWER ON A WINDOWS CE DEVICE

You can also install the Secure Viewer on a Windows CE device, by copying the necessary files to the device's non-volatile memory. To install Secure Viewer:

- ✓ Create and configure a **viewer.ini** file with the appropriate settings for the Windows CE device. (For more information about creating this file, see the previous section.)
- ✓ Determine the OS version and processor type of the Windows CE device, and then find the corresponding sub-folder in the InduSoft Web Studio program directory. For example, for Windows CE v4.0 running on a MIPS processor, find the **\Indusoft Web Studio v6.1\Redist\WinCEv4.0\MIPS** sub-folder.
- ✓ Select the Secure Viewer program (**Viewer.exe**) and ISSymbol control (**ISSymbolCE.ocx**) from this sub-folder, as well as the **viewer.ini** file you've already created, and copy all three to non-volatile memory on the Windows CE device.
- ✓ Register the ISSymbol control on the Windows CE device by executing the following command from the **Prompt** window:

```
regsvrce.exe "\<ISSymbolPath>\ISSymbolCE.ocx"
```

Example — `regsvrce.exe "\Storage Card\ISSymbolCE.ocx"`
- ✓ Save the registry settings to keep **ISSymbolCE.ocx** registered when you reboot the Windows CE device.

⇒ **Tip:** Check the device manufacturer's documentation for how to save the registry settings.

- ✓ Start the Secure Viewer by running **Viewer.exe**.

Chapter 14: Managing Applications Remotely

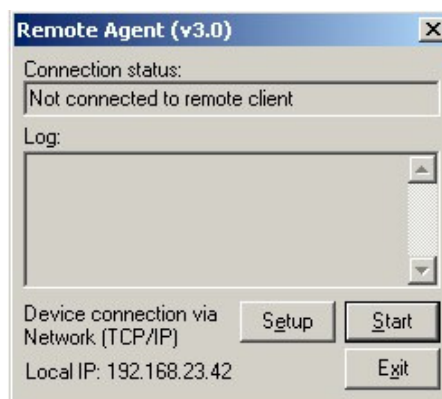
This chapter explains how to download an InduSoft Web Studio application to your runtime workstation and then monitor/manage that application remotely from a variety of devices.

Downloading the Application

After configuring an application and testing it locally (on your development workstation), you can download the application to a remote runtime workstation that is running IWS on Windows 2K/XP/Vista or running CEView on a Windows CE device.

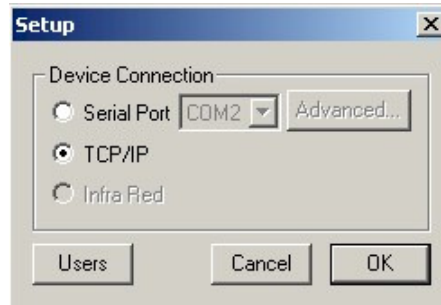
Configuring the Target Station

- ☑ 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 v6.1\Redist\<CEVersion>\<Processor Type>\BIN** folder (for NT, the location is the **\BIN** folder).
 - On a Windows CE device, the file is located in the **\<non-volatile>** folder.
- ☑ Run the **CEServer.exe** on the target workstation, and when the *Remote Agent* dialog displays, click the **Setup** button.



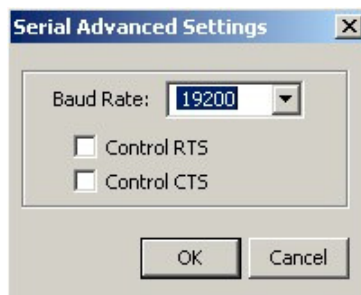
Remote Agent Dialog

The *Setup* dialog opens:



Setup Dialog

- ✓ Use the parameters on this dialog to configure communication between the development and target stations:
 - **Serial Port:** Enable this button to establish a connection to the development station through a serial port. Click the arrow button ▾ and select a communication port from the combo-box list. If you click the **Advanced** button, the *Serial Advanced Settings* dialog opens.




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:** Click the arrow button ▾ to select a predefined baud rate from the combo-box list
- * **Control RTS:** Enable (☑) this box 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:** Enable (☑) this box 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.

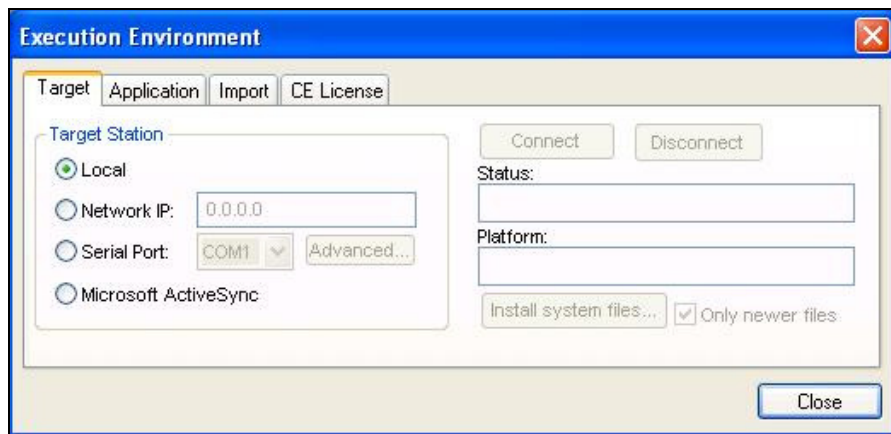
 **Note:**
For performance reasons, we recommend using a TCP/IP connection instead of a Serial Link connection.

- ☑ When you are finished, click **OK** to close the *Setup* dialog, but leave the *Remote Agent* program running in the remote workstation.

Configuring the Development Station

After configuring the target station to receive data, use the following steps to configure the development station to send the application data:

- ☑ Select **Project** → **Execution Environment** from the main menu bar (on the development workstation).

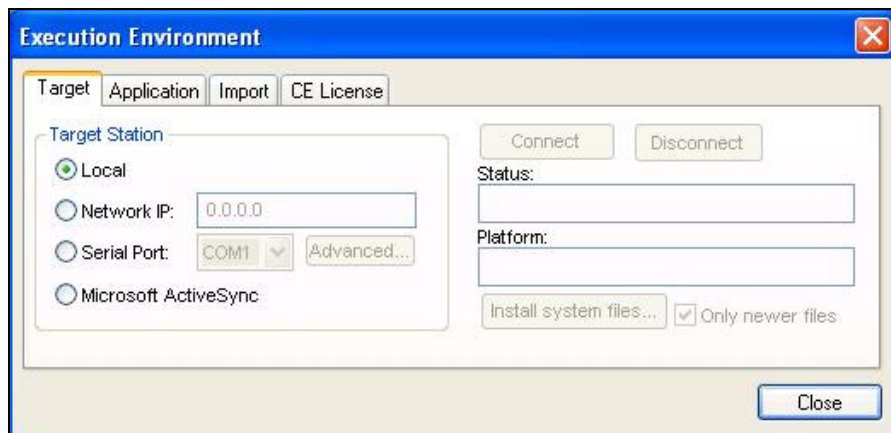


Execution Environment Dialog

This dialog contains the following tabs:

- **Target:** Use this tab to specify the target station, including:
 - * **Local, Network IP address, Serial Port, or Microsoft ActiveSync**
NOTE: Microsoft ActiveSync currently works only with Windows CE 3.0, 4.x and 5.0.
 - * **Connect/Disconnect** the target station
 - * Review station **Status**
 - * Returns the **platform** (operating system + processor type) of the target station after connecting to it.
 - * **Install system files** (or **Only newer files**)
- **Application:** Use this tab to specify the application, including:
 - * **Local and Target** directory paths
 - * Specify whether to **Send** (the application) **to target station**
 - **Only newer files** checkbox: If this checkbox is checked, then only the newer app files will be sent to the device.
 - **Keep user files** checkbox: If this checkbox was checked, then any user files in the application directory will be kept.
 - * **Send file**

- * **Run/Stop** the application
- * Review application **Status**
- **Import:** Use this tab to import an application, including
 - * Specify the **To** and **From** application path
 - * **Get From Target**
 - * **Get Log File**
 - * Review application **Status**
- **CE License:** Use this tab to specify *License Settings* and *License Codes* for a Windows CE license, including
 - * **Product Type**
 - * **Version**
 - * Number of **Web Thin Clients** and **Secure Viewers**
 - * **Site Code**
 - * **Site Key**
- ☑ Select the **Target** tab and use the following options to specify the target station to which you are going to connect.
 - **Local:** Select this button if you are going to run the application on the same station on which you developed the application.
 - **Network IP:** Select this button and type the IP address of the target station into the text box if you specified a TCP/IP connection when you configured the target station.
 - **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 station.
 - **Microsoft ActiveSync:** Select this button to connect to devices on which *MS ActiveSync* is enabled.

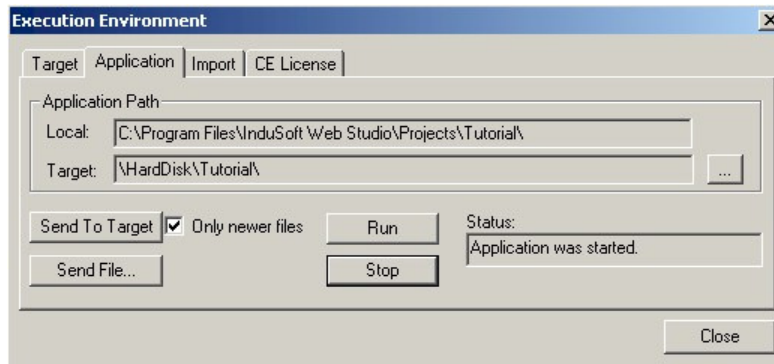


Specifying Link Type and IP Address

- ☑ Click the **Connect** button to connect to the remote workstation.

Note:
If the remote workstation is a WinCE device, you can click on the **Install System Files** button to download the CEView runtime files to the remote workstation.

- ☑ In the *Workspace* window, select the **Application** tab and click the **Send to Target** button to download the application to the remote workstation.



Specifying Link Type and IP Address

⚠ Caution:

When you send an application to the remote target station, the changes take effect online. In other words, after downloading application files to the target station, the new application(s) replace the old ones automatically – even if the application is running.

In addition, if you uncheck (disable) the **Only newer files** check-box on the *Application* tab (*Execution Environment* dialog), IWS will remove all of the files in the target application directory before downloading the new files.

📌 Note:

- If the application download is interrupted, IWS requests confirmation and advises you that the application may not run properly.
- Using the **Send app to target** menu option is the same as using the **Send app to target** button on the *Execution Control* toolbar.

- **Send file:** Click to browse to the directory and select an individual file to send.
- **Stop button:** Click to stop the application.

- ☑ After downloading all application files, click the **Run** button to execute the application in the remote target workstation.

Monitoring/Managing Applications from the Remote Station

After configuring an application and testing it locally (on your development workstation), you can download the application to a remote runtime workstation that is running under Windows 2K/XP/Vista or CEView under Windows CE.

- ☑ Before you begin, verify that the *Remote Agent* (**CEServer.exe**) is running on the remote target workstation.

 **Note:**

The **CEServer.exe** file is located in the following directory on Windows 2000/XP computers (for NT, the location is the **\BIN** folder):

\InduSoft Web Studio v6.1\Redist\CEView\<<Processor Type>\BIN

The file should be located in the **\<non-volatile>** folder on the WinCE device.

- ☑ After downloading all application files, click the **Run** button to execute the application in the remote target workstation.

Configuring Windows CE to Automatically Run an Application

By default, you must manually run your finished IWS application on the Windows CE device, either from your PC by using the *Application* tab of the *Execution Environment* dialog (see page 14–1) or on the device itself by clicking the **Start** button in the *Remote Agent* dialog.

However, you can configure the Windows CE device to automatically run a specified IWS application. To do this, edit the file **CEServer.ini** on the Windows CE device to include the following setting:

```
[Setup]
AppName=<Application Path>
```

Where **<Application Path>** is the location of the IWS application files on the Windows CE device. For example:

```
[Setup]
AppName=\Harddisk\Test\CEserverTest
```

The next time the Windows CE device boots up and opens the Remote Agent dialog (**CEServer.exe**), it will read this setting and automatically run the specified IWS application.

There are three ways to edit the **CEServer.ini** file:

- Edit the file directly on the Windows CE 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 CE device as a shared volume on your PC and edit the file there.
- Edit the file in the **\<IWS Folder>\Redist** directory before you install the system files on the Windows CE 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 CE devices; and
- You already know the location (file path) of the IWS application files on the device (perhaps by using the normal installation method on a test device).

Chapter 15: Scripting Languages: IWS and VBScript

Working with the IWS Scripting Language, Expressions, and Functions

This section explains how to work with the InduSoft Web Studio (IWS) scripting language, expressions, and functions.

Using Tags

Tags are variables that can receive the results of expressions specified in screens and worksheets (such as communication points in field equipment, calculation results, alarm points, and so forth).

 **Note:**

We recommend that you read and understand the concepts discussed in *Chapter 5: Working with Tags* before you read this chapter.

Specifying Data Types

You can use the IWS Scripting Language in many places, for example:

- Dynamic object properties in the Application Builder
- Screen logic in the Application Builder
- Scheduler worksheets
- Math worksheets

A *Math* worksheet has two columns:

- **Tag Name:** Names of tags to receive results from expressions specified in the **Expression** column on the same line.
- **Expression:** Mathematical expressions defined by InduSoft Web Studio.

For example, **Tag Name a**, will receive the result of **Expression (10c) –5**.

	Tag Name	Expression
1	a	10*c-5

Example Math Worksheet

 **IMPORTANT!**

- No attributions are done on the **Expression** column. If you write **A=2** in this column, IWS will compare A with the number 2. The integer result of this expression (Boolean value 0 if false or 1 if true) will be written to the tag in the **Tag Name** column.
- The system is not case-sensitive.
- To add comments to an expression line, use the // characters.

The following data types are acceptable:

- **Integer numbers** (32 bits): 1 23 45 -123
- **Floating point** (8 bytes): 1.234 -775.344
- **Hexadecimal integer numbers** (32 bits): 0x5 0xA0 0xBC4
- **Strings** (1024 characters): “demo” “new demo”

Accessing the Tags Database

To write a value in the database, use the tag name directly, for example:

- In the following script line, the **X** tag will receive the sum of two tags, `level` and `temp`:

	Tag Name	Expression
1	X	Level+temp

Example 1

- IWS allows you to read and write tags using references or pointers. You can declare a tag being used as pointer to another tag in two ways:
 - As a *string* (a pointer to an undefined type)
 - As a pointer of a specific kind (*recommended*)

	Name	Array Size	Type	Description
1	Valve_Fill_State	0	String	
2	@pointer_to_integer	0	Integer	//Pointer to an integer value

Example 2

In the preceding figure **Valve_Fill_State** is a variable of a string type that is a pointer. The **@pointer_to_integer** variable is a pointer to integer values.

Notes:

- The syntax **@tag** allows a tag to access another tag by reference.
- You can use any tag declared as a string as an indirect tag (*pointer*).

Arithmetic Operators

InduSoft Web Studio supports all the following Arithmetic operators:

- + addition
- subtraction
- * multiplication
- / division
- > greater than
- < less than
- = equal
- >= greater than or equal to
- <= less than or equal to

<> different from (unequal to)

 **Notes:**

Math functions are calculated from left to right, according to a specific order for each operator in the formula. To change the order of evaluation, enclose in parentheses the part of the formula to be calculated first. For example, the following formula produces 11 because multiplication is calculated before addition. The formula 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:

$(5+2)*3$

Logic Operators

InduSoft Web Studio supports all the following Logic operators:

- **AND** AND, logic
- **NOT** NOT, logic
- **OR** OR, logic
- **XOR** exclusive or, logic
- **&** AND, bit
- **|** OR, bit
- **~** NOT, bit
- **^** XOR, bit
- **>>n** rotate right – Rotate n bits to right.
- **<<n** rotate left– Rotate n bits to left.

Using Functions

The function tag names used in IWS must conform to the following syntax:

- **num<Name>**: Numerical tag or value
- **str<Name>**: String tag or value
- **tag<Name>**: Tag Name
- **optNum<Name>**: Optional Numerical tag or value
- **optStr<Name>**: Optional String tag or value
- **optTag<Name>**: Optional Tag Name

This syntax identifies the argument types required for each parameter of the IWS function.

InduSoft Web Studio has more than one hundred functions ready for use. For a complete list of available functions, see “Appendix A: Studio Functions.”

Example of functions:

Functions	Execution	2K/XP/Vista	Win CE	Web Client
False()	Synchronous	✓	✓	✓
If()	Synchronous	✓	✓	✓
True()	Synchronous	✓	✓	✓
Format()	Synchronous	✓	✓	✓
GetBit()	Synchronous	✓	✓	✓
Max()	Synchronous	✓	✓	✓
Min()	Synchronous	✓	✓	✓
SendEmail()	Synchronous	✓	✓	✓
Log()	Synchronous	✓	✓	✓
Play()	Synchronous/Asynchronous	✓	✓	✓

IMPORTANT:

You can use the *Database Spy* window to execute any math expression by writing the expression in the *Tag Name* field and clicking the **Toggle** button.

The return value of the expression will display in the *Value* field.

Overview of VBScript

The Microsoft Visual Script Language (VBScript), is a simple, standard and flexible scripting language that allows you to implement logics and algorithms within the IWS application.

IWS implements the Microsoft Visual Basic Scripting Edition 5.5 or higher. Because IWS hosts VBScript, the user can take advantage of all features provided by this language, such as:

- Use syntax, operators and functions available in the language
- Create new variables and Procedures (Functions and/or Sub-routines)
- Access properties, methods and/or events from COM objects, including ActiveX controls
- Execute the logics in any platform that supports VBScript, including Microsoft Windows 2K/XP/Vista (IWS Server station), Microsoft Windows CE (CEView) and Microsoft Internet Explorer (Web Thin Client).

⚠ Caution:

When creating the image for the Microsoft Windows CE device, the hardware manufacturer must enable the support for VBScript on it, so CEView will be able to execute the scripts configured in VBScript language on the device. If you are not sure if the image loaded on your device supports VBScript, please consult the hardware manufacturer.

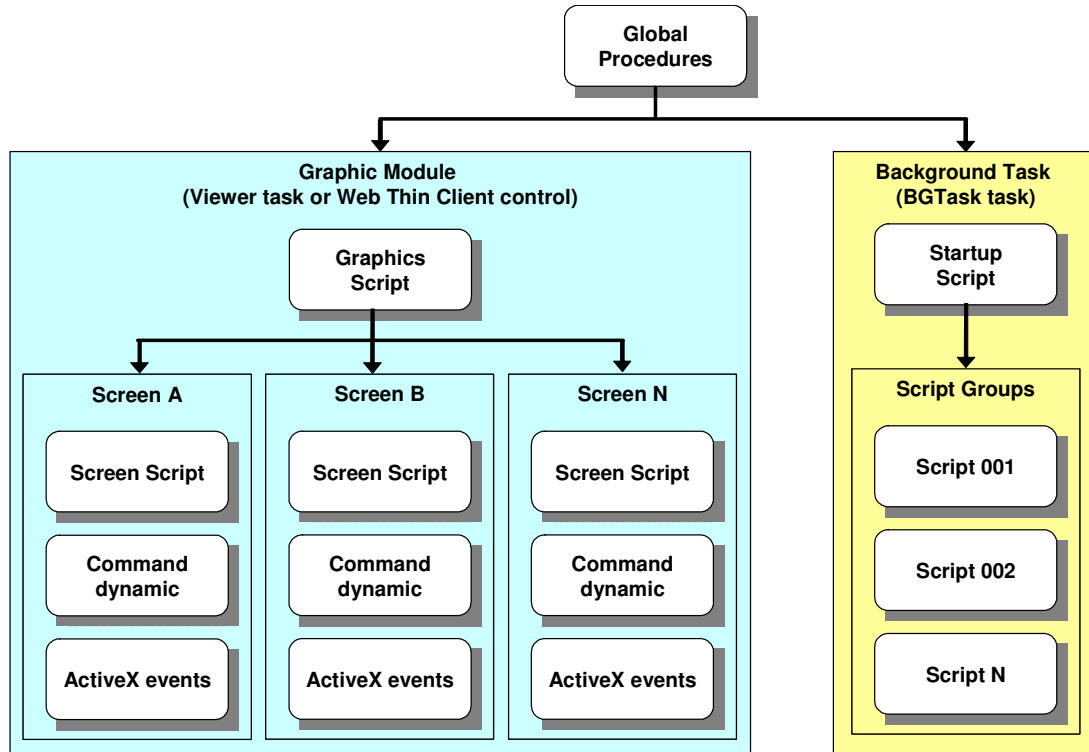
The aim of this documentation is to provide an overview about the integration of VBScript with IWS. Furthermore, it can be used as a quick reference for the most used features of the language. For a full description of the language as well as its interfaces and functions, please consult the Microsoft documentation. At the time that this documentation was written, you could read the VBScript language documentation at Microsoft Developers Network (MSDN).

VBScript in IWS

The following table provides a summary of the VBScript interfaces supported by IWS:

Interface	Scope for Procedures and Variables	Execution	Functionality
Global Procedures	Graphics and Tasks	-	Declaration of Procedures
Graphics Script	Graphics Script interface only	Server (Viewer) + Web Thin Clients	<ul style="list-style-type: none"> ▪ Declaration of Variables ▪ Declaration of Procedures ▪ Execution
Screen Script	Screen where the script is configured	Server (Viewer) + Web Thin Clients	<ul style="list-style-type: none"> ▪ Declaration of Variables ▪ Declaration of Procedures ▪ Execution
Command Dynamic	Object where the script is configured	Server (Viewer) + Web Thin Clients	<ul style="list-style-type: none"> ▪ Declaration of Variables ▪ Execution
ActiveX Events	Object where the script is configured	Server (Viewer) + Web Thin Clients	<ul style="list-style-type: none"> ▪ Declaration of Variables ▪ Execution
Startup Script	All Script Sheets from Tasks	Server (BGTask)	<ul style="list-style-type: none"> ▪ Declaration of Variables ▪ Declaration of Procedures ▪ Execution
Script Groups	Script Group only	Server (BGTask)	<ul style="list-style-type: none"> ▪ Declaration of Variables ▪ 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 IWS built-in 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.

The following VBScript interfaces are available in IWS...

Global Procedures

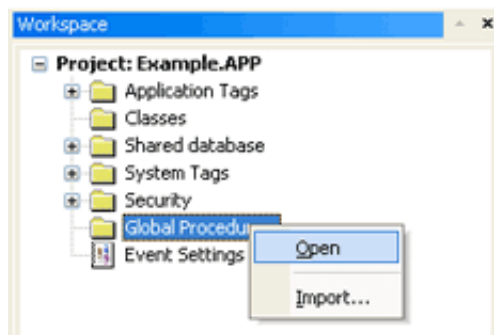
This Global Procedures interface is used to create a library of VBScript functions and sub-routines that can be called by any other scripting interface in IWS. The procedures declared here are never directly executed during runtime; they must be explicitly called by another script.

➔ **Caution:**

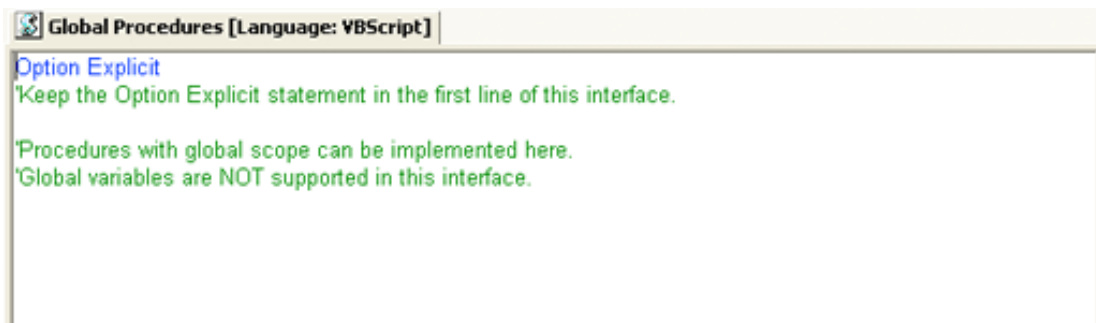
IWS will not prevent you from declaring two or more procedures with the same name. (This includes procedures imported from external files; see “Importing Procedures from an External File” below.) However, if you do, then your application may behave unexpectedly during runtime. Verify that all of your global procedures are named correctly.

To use the Global Procedures interface:

1. In the *Database* tab of the Workspace, right-click on the *Global Procedures* folder and choose **Open** from the pop-up menu.



The Global Procedures interface is displayed.



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

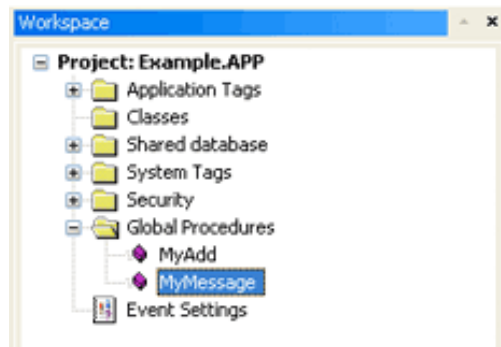
Function MyAdd(number1, number2)
    MyAdd = number1 + number2
End Function

Sub MyMessage(message)
    MsgBox(message)
End Sub
```

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 (**File** → **Save**). The functions and sub-routines are added to the Global Procedures folder in the Workspace.



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
```

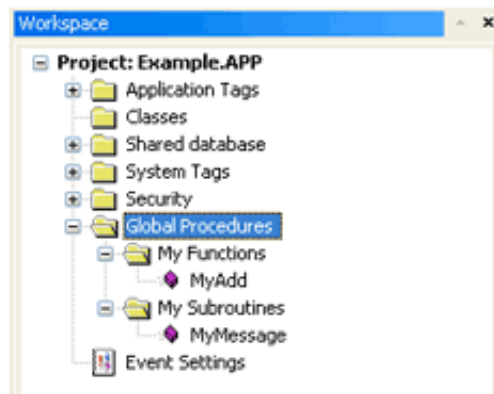
```

'$region:My Functions
Function MyAdd(number1, number2)
    MyAdd = number1 + number2
End Function

'$region:My Subroutines
Sub MyMessage(message)
    MsgBox(message)
End Sub

```

2. Save your changes (**File** → **Save**). The procedures are organized into subfolders in the Workspace.

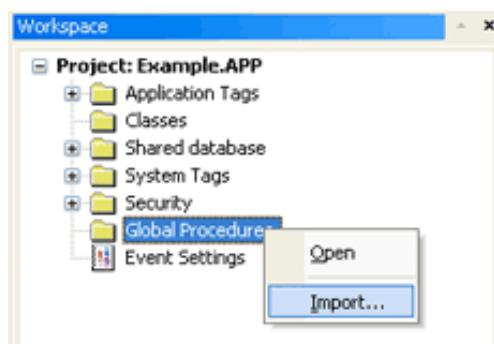


IMPORTING PROCEDURES FROM AN EXTERNAL FILE

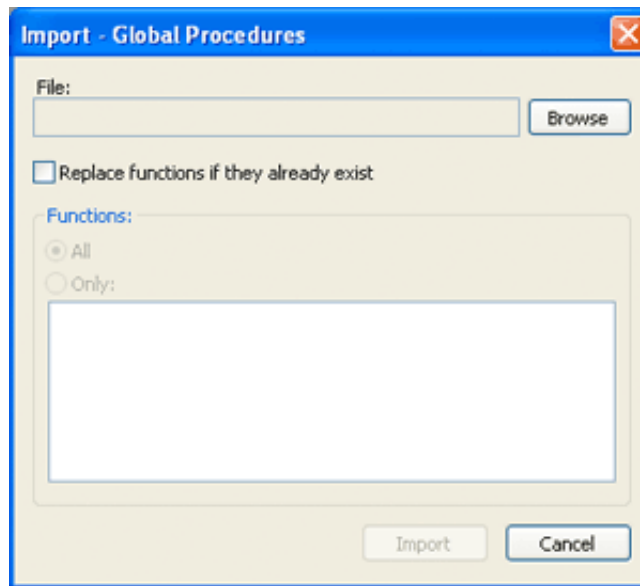
You can also import procedures from an external file and add them to the Global Procedures folder. This is useful if you have a library of existing procedures that you want to reuse in different IWS applications.

To import procedures:

1. Save and close all open screens and worksheets.
2. In the *Database* tab of the Workspace, right-click on the *Global Procedures* folder and choose **Import...** from the pop-up menu.



The *Import - Global Procedures* dialog is displayed.

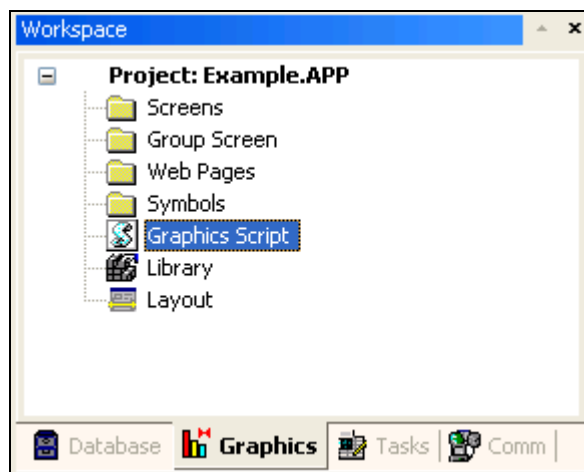


3. In the **File** field, click on the **Browse** button to open a standard Windows file browser and select a global procedures file. (This is a plain text file that is saved with the `.gis` file extension.)
4. Click the **Replace functions if they already exist** checkbox to overwrite functions in your IWS application with functions imported from the global procedures file (`*.gis`), if the functions have the same name.
5. In the *Functions* area, you can import **All** functions from the global procedures file (`*.gis`), or import **Only** selected functions.
6. Click the **Import** button.

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 Workspace:



This interface can be used to execute logics on the following events, based on pre-configured 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 starting the application.
- `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 application is running.
- `Graphics_OnEnd()`: The code configured within this sub-routine is automatically executed just once when the graphic module is closed.

 **Caution:**

Do NOT change the name of the pre-configured sub-routines. If you do, the system will be unable to call them automatically.

On the Server (where IWS 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 Web Thin Client (Web Browser):

- The graphic module is the ISSymbol control.
- The `Graphics_OnStart()` sub-routine is executed once on the Web Thin Client station after logging in successfully.
- The `Graphics_WhileRunning()` sub-routine keeps being executed on the Web Thin Client station while the ISSymbol control is hosted by the Web Browser.
- The `Graphics_OnEnd()` sub-routine is executed once on the Web Thin Client station when the Web Browser is shut down (or when the ISSymbol control is no longer hosted by the Web Browser).

The variables and procedures declared in this interface are NOT available for any other VBScript interface – they have local scope only.

 **Note:**

- The execution of the Graphic Script sub-routines on the server is totally independent of the execution on the Web Thin Client stations.
- The procedures and/or variables created in this interface have local scope: they can be accessed only from the Graphic Script interface of the local station where they are being executed.

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 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

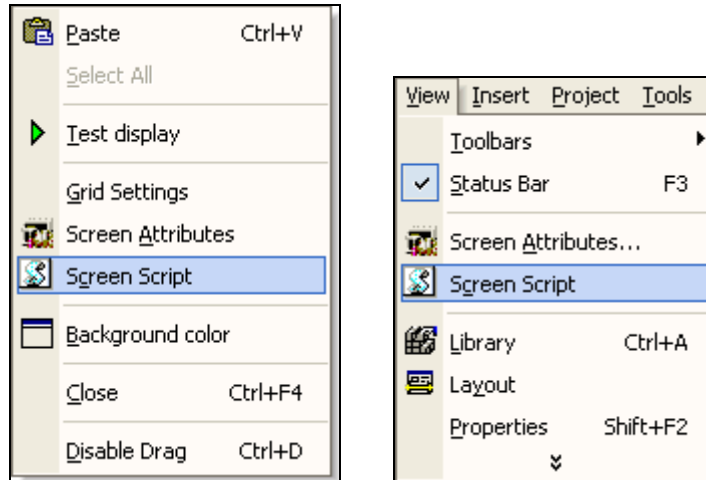
' This procedure is executed just once when the graphic module is started
Sub Graphics_OnStart ()
    MsgBox("Welcome to the system!")
End Sub

' 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
```

Graphic Module - Screen Script

To edit the Graphics Script interface, right-click on the screen, and select the Screen Script option. You can also select View -> Screen Script from the drop-down menu.



This interface can be used to execute logics on the following events, based on pre-configured sub-routines:

- `Screen_OnOpen()`: The code configured within this sub-routine is automatically executed just once when its screen is open.
- `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 application 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 name of the pre-configured sub-routines. If you do, the system will be unable to call them automatically.

📌 Notes:

- The execution of the Screen Script sub-routines on the server is totally independent of the execution on the Web Thin Client stations.
- The procedures and/or variables created in this interface have local scope: they can be accessed only from the specific screen where they are implemented.

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 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
```

Graphic Module - Command Dynamic

To edit the Command dynamic interface, do the following:

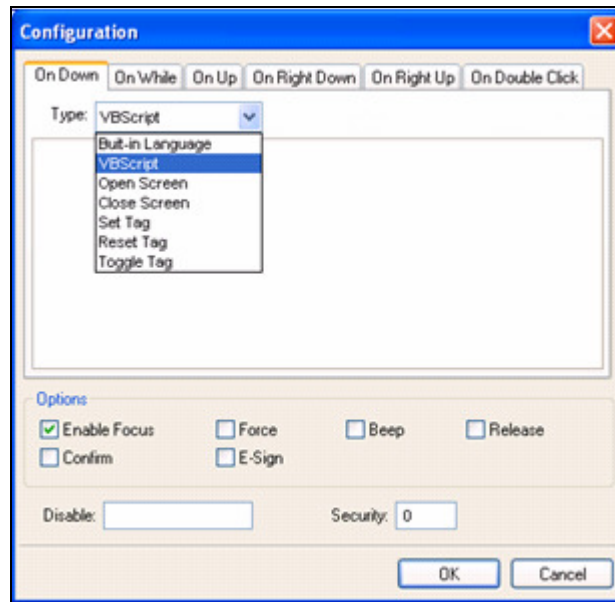
- Select the object.
- Click the Command dynamic icon on the Mode toolbar.



- Right-click on the object.
The Object Properties dialog box for the Command dynamic will open.



- ✓ Click the Config... button.
- ✓ Select VBScript as the Type.



Use this interface to execute logics when the user clicks on the object where the Command dynamic is applied (during the runtime) or presses the shortcut (Key) associated with the Command dynamic.

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 application. 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 dynamic is configured.

Note:

Further information about the Command dynamic is available in chapter 3, which describes the dynamics from the screen editor.

Example:

```
'The script below will be executed when the user clicks on the object
'where this dynamic is configured
$MyValue = InputBox("Please enter the new set-point", "Set-point")
```

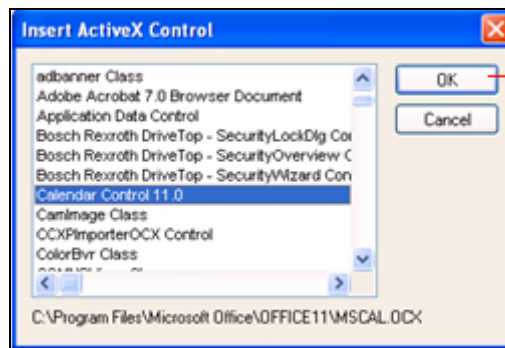

Graphic Module - ActiveX Events

To edit the ActiveX Events interface, select the Script option from the Events tab of the ActiveX object inserted on the screen.

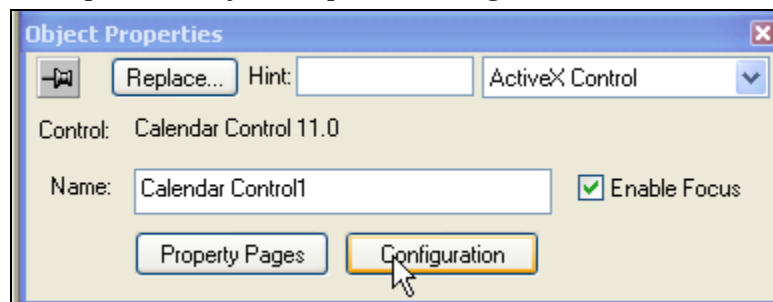
- ☑ Click the ActiveX Control icon in the Mode toolbar.



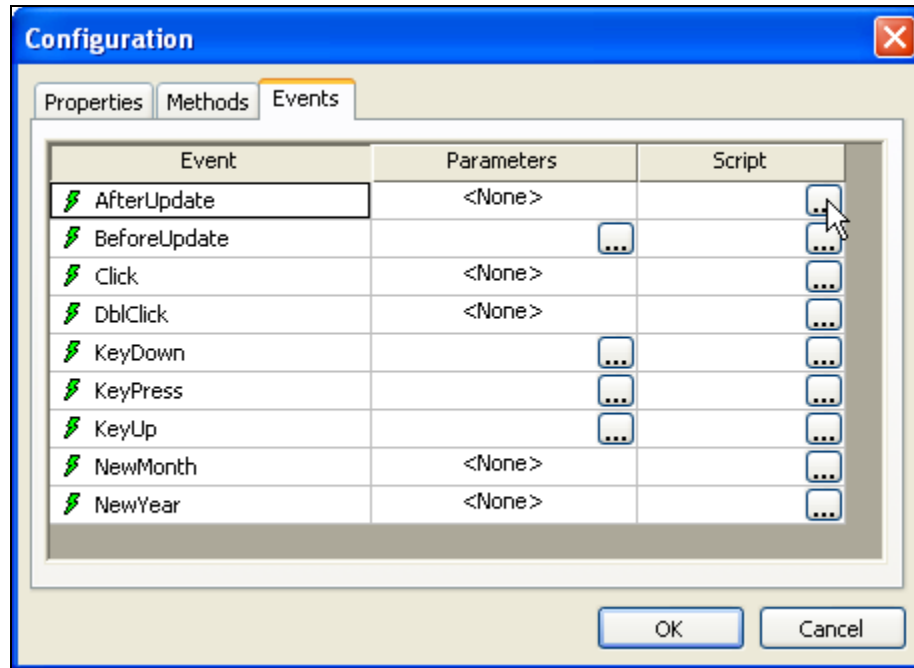
The Insert ActiveX Control dialog box opens.



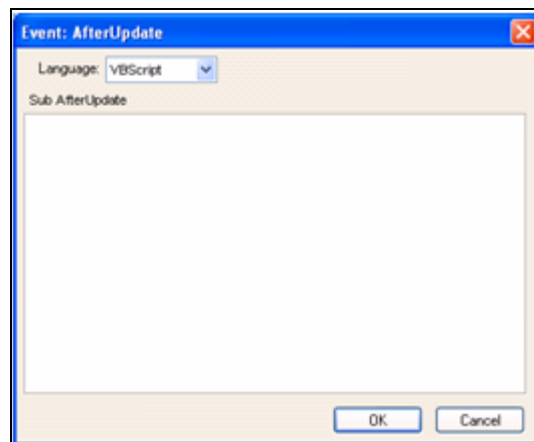
- ☑ Select the ActiveX Control that you wish to use, and then click OK.
- ☑ The object that symbolizes the selected ActiveX Control will display. Right-click on this object to open the Object Properties dialog.



- ☑ Click the Configuration button. The Configuration dialog will open. Click the Events tab.



- ☑ 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 application.

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.

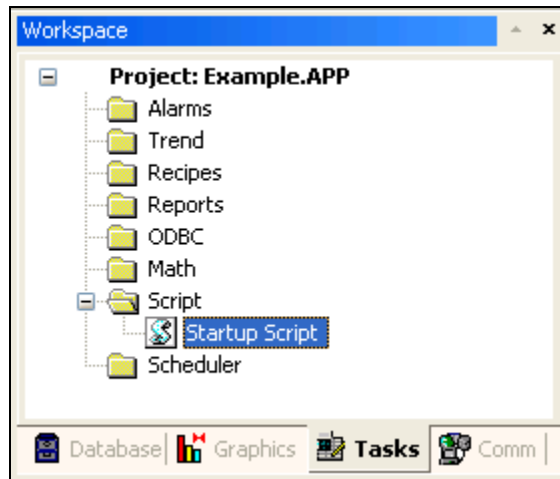
Note: Further information about the ActiveX Events is available in the Using the Active Objects Toolbar section of Chapter 7.

Example:

```
'The script below will be executed when the Calendar Control ActiveX  
'triggers its"AfterUpdate" event  
$MyYear = CalendarControl1.Year  
$MyMonth = CalendarControl1.Month  
$MyDay = CalendarControl1.Day
```

Background Task - Startup Script

To edit the Startup Script interface, click its icon from the Script folder, on the Tasks tab of the Workspace:



The code configured in this interface 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 application is starting.

You can declare and initialize variables and implement procedures. However, variables or procedures declared in this interface will be available ONLY to the script groups from the Script task – 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
```

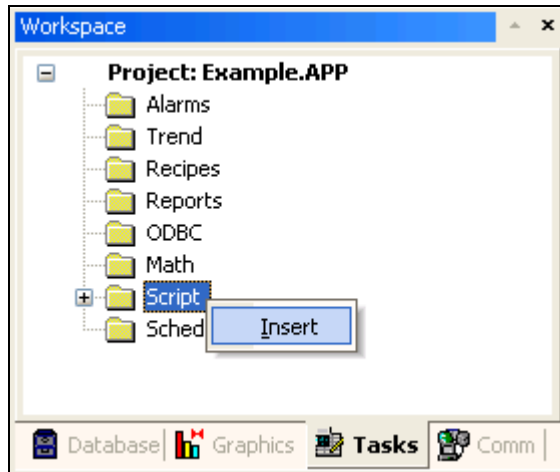
```
Sub CheckLimits (myValue, myHiLimit, myLoLimit)
    If (myValue > myHiLimit Or myValue < myLoLimit) Then
        MsgBox ("Value out of range")
    End If
End Sub
```

' The code configured here is executed just once when the Background task is started

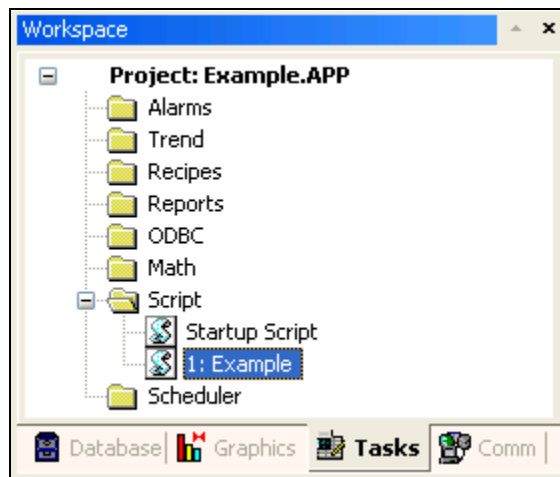
```
If $GetOS()=3 Then
    MsgBox ("Welcome! This application is running under Microsoft
Windows CE operating system.")
Else
    MsgBox ("Welcome! This application Is running under Microsoft
Windows 2K/XP/Vista operating system.")
End If
```

Background Task - Script Groups

To create a new Script Group, right-click the Script folder in the Tasks tab of the Workspace. Select the Insert option from the pop-up menu:



To edit an existing Script Group interface, click its icon in the Script folder on the Tasks tab of the Workspace:



The code configured in each Script Group is executed by the Background Task. IWS scans the Script Groups sequentially (based on the number of the group) and executes only the groups in which the condition configured in the Execution field of the Script Group is TRUE (different from 0).

Note:

You must use the syntax supported by the IWS built-in language in the Execution field of each Script group. Only the body of the Script group supports VBScript language.

Caution:

When any Script Group is saved during the runtime (on-line configuration), the Startup Script interface will be executed again, and the current value of the local variables of any Script Group will be reset.

Variables declared in the group have local scope for that specific group only. They are not available for any other VBScript interface. From the Script Group, you can call

procedures implemented in the Global Procedures interface or in the Startup Script interface; however, you cannot implement (declare) procedures in the Script Groups.

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
```

Language Reference

Constants

Color Constants		
Constant	Value	Description
vbBlack	&h00	Black
vbRed	&hFF	Red
vbGreen	&hFF00	Green
vbYellow	&hFFFF	Yellow
vbBlue	&hFF0000	Blue
vbMagenta	&hFF00FF	Magenta
vbCyan	&hFFFF00	Cyan
vbWhite	&hFFFFFF	White

Comparison Constants		
Constant	Value	Description
vbBinaryCompare	0	Perform a binary comparison
vbTextCompare	1	Perform a textual comparison

Date and Time Constants		
Constant	Value	Description
vbSunday	1	Sunday
vbMonday	2	Monday
vbTuesday	3	Tuesday
vbWednesday	4	Wednesday
vbThursday	5	Thursday

vbFriday	6	Friday
vbSaturday	7	Saturday
vbUseSystemDayOfWeek	0	Use the day of the week specified in your system settings for the first day of the week.
vbFirstJan1	1	Use the week in which January 1 occurs (default).
vbFirstFourDays	2	Use the first week that has at least four days in the new year.
vbFirstFullWeek	3	Use the first full week of the year.

Date Format Constants

Constant	Value	Description
vbGeneralDate	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.
vbLongDate	1	Display a date using the long date format specified in your computer's regional settings.
vbShortDate	2	Display a date using the short date format specified in your computer's regional settings.
vbLongTime	3	Display a time using the long time format specified in your computer's regional settings.
vbShortTime	4	Display a time using the short time format specified in your computer's regional settings.

Miscellaneous Constants

Constant	Value	Description
vbObjectError	-2147221504	User-defined error numbers should be greater than this value.

Box Constants - Buttons and Icons

Constant	Value	Description
vbOKOnly	0	Display OK button only.
vbOKCancel	1	Display OK and Cancel buttons.
vbAbortRetryIgnore	2	Display Abort, Retry, and Ignore buttons.
vbYesNoCancel	3	Display Yes, No, and Cancel buttons.
vbYesNo	4	Display Yes and No buttons.
vbRetryCancel	5	Display Retry and Cancel buttons.

vbCritical	16	Display Critical Message icon.
vbQuestion	32	Display Warning Query icon.
vbExclamation	48	Display Warning Message icon.
vbInformation	64	Display Information Message icon.
vbDefaultButton1	0	First button is the default.
vbDefaultButton2	256	Second button is the default.
vbDefaultButton3	512	Third button is the default.
vbDefaultButton4	768	Fourth button is the default.
vbApplicationModal	0	Application modal. The user must respond to the message box before continuing work in the current application.
vbSystemModal	4096	System modal. On Win16 systems, all applications are suspended until the user responds to the message box. On Win32 systems, this constant provides an application modal message box that always remains on top of any other programs you may have running.

Box Constants – SelectedButton

Constant	Value	Description
vbOK	1	OK button was clicked.
vbCancel	2	Cancel button was clicked.
vbAbort	3	Abort button was clicked.
vbRetry	4	Retry button was clicked.
vbIgnore	5	Ignore button was clicked.
vbYes	6	Yes button was clicked.
vbNo	7	No button was clicked.

String Constants

Constant	Value	Description
vbCr	Chr(13)	Carriage return
VbCrLf	Chr(13) & Chr(10)	Carriage return–linefeed combination
vbFormFeed	Chr(12)	Form feed; not useful in Microsoft Windows
vbLf	Chr(10)	Line feed
vbNewLine	Chr(13) & Chr(10) or Chr(10)	Platform-specific newline character; whatever is appropriate for the platform
vbNullChar	Chr(0)	Character having the value 0
vbNullString	String having value 0	Not the same as a zero-length string (""); used for calling external procedures

vbTab	Chr(9)	Horizontal tab
vbVerticalTab	Chr(11)	Vertical tab; not useful in Microsoft Windows

Tristate Constants		
Constant	Value	Description
vbUseDefault	-2	Use default from computer's regional settings.
vbTrue	-1	TRUE
vbFalse	0	FALSE

VarType Constants		
Constant	Value	Description
vbEmpty	0	Uninitialized (default)
vbNull	1	Contains no valid data
vbInteger	2	Integer subtype
vbLong	3	Long subtype
vbSingle	4	Single subtype
vbDouble	5	Double subtype
vbCurrency	6	Currency subtype
vbDate	7	Date subtype
vbString	8	String subtype
vbObject	9	Object
vbError	10	Error subtype
vbBoolean	11	Boolean subtype
vbVariant	12	Variant (used only for arrays of variants)
vbDataObject	13	Data access object
vbDecimal	14	Decimal subtype
vbByte	17	Byte subtype
vbArray	8192	Array

Errors

VBScript Run-time Errors	
Error Number	Description
429	ActiveX component can't create object
507	An exception occurred
449	Argument not optional
17	Can't perform requested operation
430	Class doesn't support Automation
506	Class not defined
11	Division by zero
48	Error in loading DLL

5020	Expected ')' in regular expression
5019	Expected ']' in regular expression
432	File name or class name not found during Automation operation
92	For loop not initialized
5008	Illegal assignment
51	Internal error
505	Invalid or unqualified reference
481	Invalid picture
5	Invalid procedure call or argument
5021	Invalid range in character set
94	Invalid use of Null
448	Named argument not found
447	Object doesn't support current locale setting
445	Object doesn't support this action
438	Object doesn't support this property or method
451	Object not a collection
504	Object not safe for creating
503	Object not safe for initializing
502	Object not safe for scripting
424	Object required
91	Object variable not set
7	Out of memory
28	Out of stack space
14	Out of string space
6	Overflow
35	Sub or function not defined
9	Subscript out of range
5017	Syntax error in regular expression
462	The remote server machine does not exist or is unavailable
10	This array is fixed or temporarily locked
13	Type mismatch
5018	Unexpected quantifier
500	Variable is undefined
458	Variable uses an Automation type not supported in VBScript
450	Wrong number of arguments or invalid property assignment

VBScript Syntax Errors

Error Number	Description
1052	Cannot have multiple default property/method in a Class
1044	Cannot use parentheses when calling a Sub
1053	Class initialize or terminate do not have arguments
1058	'Default' specification can only be on Property Get
1057	'Default' specification must also specify 'Public'
1005	Expected '('
1006	Expected ')'
1011	Expected '='
1021	Expected 'Case'
1047	Expected 'Class'
1025	Expected end of statement
1014	Expected 'End'
1023	Expected expression
1015	Expected 'Function'
1010	Expected identifier
1012	Expected 'If'
1046	Expected 'In'
1026	Expected integer constant
1049	Expected Let or Set or Get in property declaration
1045	Expected literal constant
1019	Expected 'Loop'
1020	Expected 'Next'
1050	Expected 'Property'
1022	Expected 'Select'
1024	Expected statement
1016	Expected 'Sub'
1017	Expected 'Then'
1013	Expected 'To'
1018	Expected 'Wend'
1027	Expected 'While' or 'Until'
1028	Expected 'While,' 'Until,' or end of statement
1029	Expected 'With'
1030	Identifier too long
1014	Invalid character
1039	Invalid 'exit' statement
1040	Invalid 'for' loop control variable
1013	Invalid number
1037	Invalid use of 'Me' keyword
1038	'loop' without 'do'
1048	Must be defined inside a Class

1042	Must be first statement on the line
1041	Name redefined
1051	Number of arguments must be consistent across properties specification
1001	Out of memory
1054	Property Set or Let must have at least one argument
1002	Syntax error
1055	Unexpected 'Next'
1015	Unterminated string constant

Functions

Function Names			
Abs	Array	Asc	Atn
CBool	CByte	CCur	CDate
CDbl	Chr	CInt	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

Keywords	
Keyword	Description
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.

Methods

Methods	
Method	Description
Clear	Clears all property settings of the Err object.
Execute	Executes a regular expression search against a specified string.
Raise	Generates a run-time 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.

Objects and Collections

Objects and Collections	
Name	Description
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 run-time errors. Accepts the Raise and Clear methods for generating and clearing run-time 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.

Operators

Arithmetic		
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		
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		
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		
Symbol	Name	Description
=	Assignment	Assigns a value to a variable or property.

Properties

Properties	
Property Name	Description
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

Statements	
Statement Name	Description
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.

Tips and Tricks

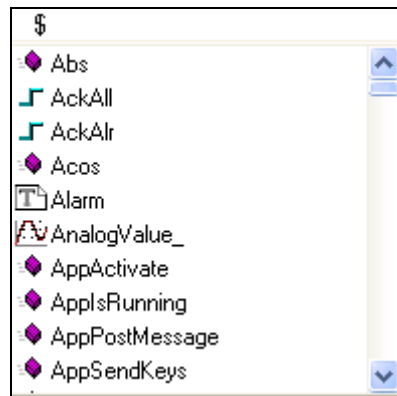
VBScript Editor - IntelliSense

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.

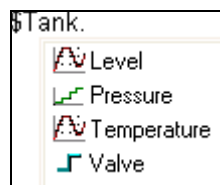
IntelliSense comprises the following options:

- **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 IWS built-in language. Selecting from the list inserts the member into your code.

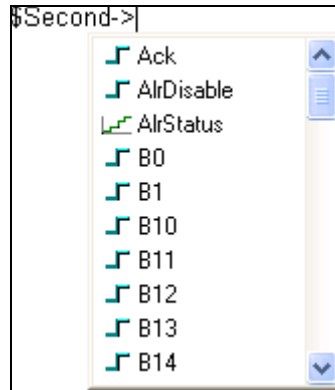
When you type the \$ character on any VBScript interface, a list box will automatically open with the list of all tags available for the current application as well as all functions from the IWS built-in language.



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:



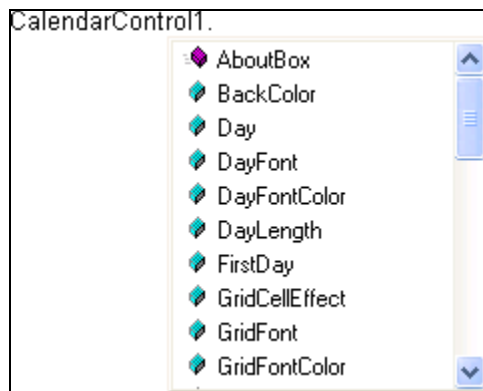
When you type the name of a tag followed by the hyphen and greater than characters (->) on any VBScript interface, a list box will automatically open with the list of all fields available for this tag:



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
	Function from the IWS built-in 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 IWS built-in 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+SPACEBAR 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 CE 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 Web 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
	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, CVDate
	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 Is keyword or any comparison operators
	Expressions containing a range of values using the To keyword
Strings	Fixed-length strings
	LSet, RSet
	Mid Statement
	StrConv
Using Objects	Collection access using !

Screen Events


In addition to the Screen Script, you can configure logics using the IWS built-in 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 (IWS Built-in language), IWS will respect the following execution order:

Event	Order of execution
When opening the screen	<ul style="list-style-type: none"> ▪ Screen_OnOpen() sub-routine from the Screen Script interface (VBScript language) ▪ On Open from the Screen Logic interface (IWS Built-in language)
When closing the screen	<ul style="list-style-type: none"> ▪ On Close from the Screen Logic interface (IWS Built-in language) ▪ Screen_OnClose() sub-routine from the Screen Script interface (VBScript language)

MsgBox() and InputBox() functions

The MsgBox() and InputBox() functions from the VBScript language allow you to display pop-up messages during the 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 IWS built-in language.

Support for ActiveX objects

Using the VBScript interfaces for the Graphic module (Graphics Script, Screen Script, Command dynamic and ActiveX events), you can use this syntax to access properties and methods directly from any ActiveX 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 (Object Properties dialog window) to modify this name.

After inserting an ActiveX 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 CalendarControll1 ActiveX object
```

```
CalendarControll1.Day
```

```
//Triggers the method AboutBox from the CalendarControll1 ActiveX object
```

```
CalendarControll1.AboutBox
```

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” below.

NOT OPERATOR IN BUILT-IN LANGUAGE

By contrast, the NOT operator in the built-in scripting language toggles the given numeric value as if it was 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

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...	
	IWS Tag	VBScript
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 application. Logical and arithmetic operations — especially the logical operator NOT — could change tag values in unexpected ways.

You can change this behavior, if necessary, by changing your application's runtime settings.

CHANGING HOW VBSCRIPT HANDLES BOOLEAN TAGS

As stated above, the numeric value of TRUE in a Boolean tag is different than the numeric value of TRUE in VBScript. You can change this behavior by editing the **<Application Name>.app** file to change the following setting:

```
[Scripts]
VBBoolean=<value>
```

If **VBBoolean** is set to 0, then the application will behave as described above. If **VBBoolean** is set to 1, then VBScript — as it is implemented within IWS — will read/write a value of 1 for TRUE on Boolean tags. (This setting does not apply to Integer or Real tags.)

For applications created with IWS v6.1 SP4 or later, the default setting for **VBBoolean** is 1.

For applications created with IWS v6.1 SP3 or earlier and then updated to v6.1 SP4 or later, the default setting for **VBBoolean** is 0. This is to maintain backward compatibility.

Note:

Be careful when defining a custom property on a generic object or 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
```

Windows CE Support

CEView also support VBScript. The hardware manufacturer of the Microsoft Windows CE 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 Microsoft WindowsCE device is created.

If you are not sure if the image loaded on your device supports VBScript, please consult the hardware manufacturer.

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.

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 the 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
```

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.

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
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.

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
```

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.

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 can't.

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.

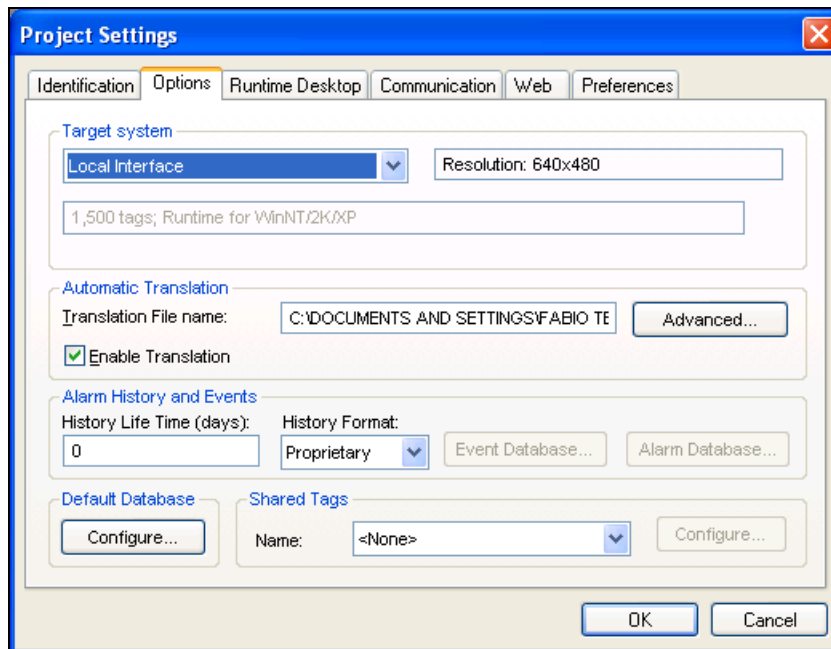
Chapter 16: Using the Translation Tool/Editor

Using the Translation Tool, you can translate any text from the application that is visible during the runtime into different languages (idioms). The main advantages of this feature are:

- You do not need to re-create the screens, alarms or any other interface of your application to translate it into another language. All application files remain in the original language (e.g., English), and the translations to other languages are contained in external files.
- InduSoft Web Studio is UNICODE-compliant (2 bytes for each character). Therefore, you can translate your application into any language that requires UNICODE characters (Japanese, Chinese, Korean, etc.) as long as you select a UNICODE font for your project.
- The translations can be configured in a standard, comma-delimited text file (*.CSV). Therefore, you can either use the *Translation Editor* in IWS (**Tools > Translation Editor**) to write the translated text, or you can use any third-party editor that can save CSV files, such as Microsoft Notepad or Microsoft Excel.
- The *Translation Tool* translates entire phrases or paragraphs instead of translating word-by-word, allowing you to adjust the translated text according to the grammar rules of each language.
- You can configure the default translation language when running the application by using the *Automatic Translation* interface. This is available from the **Options** tab of the *Project Settings* dialog (**Project > Settings**). In addition to that, you can change the language during the runtime (on-the-fly) by using the **SetTranslationFile()** built-in function of IWS.
- When you execute the **SetTranslationFile()** function on a Web Thin Client station, you apply it to that particular Web Thin Client instance only. Therefore, you can have several Web Thin Clients running simultaneously, and each one can display the screens in a different language.

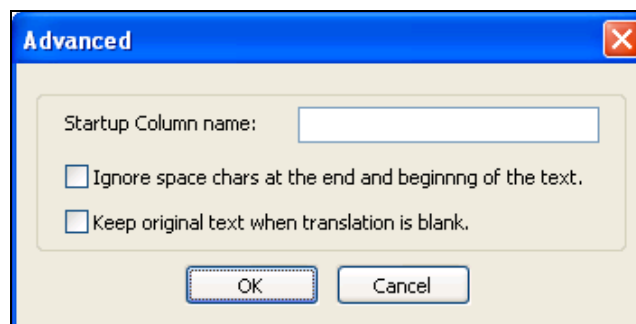
Project Settings for Automatic Translation

When configuring an application to support more than one language, you can configure the following parameters in the **Options** tab of the *Project Settings* dialog (menu **Project > Settings**):



Project Settings Dialog – Options Tab

- **Translation File Name:** The translation file configured in this field will be loaded by default when launching the application. This option is useful to configure the default translation file name when launching the application. The user can set a different language during the runtime by executing the **SetTranslationFile()** function.
- **Enable Translation:** When this option is unchecked (clear), the Translation Tool is disabled and the text elements are displayed only as configured in the original application files.
- When you press the **Advanced** button, you can configure the following settings for the Translation Tool:



Advanced Dialog

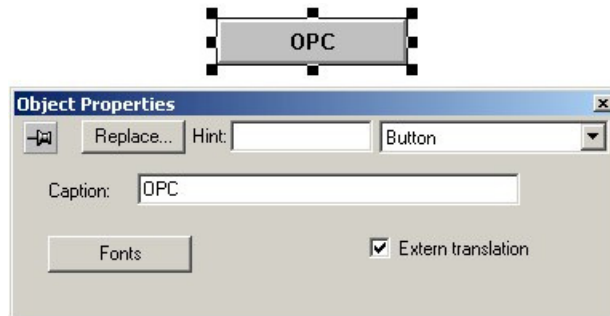
- * **Startup Column Name:** You can create a CSV file with translation for more than one language. The original text that was used when the application was created is in the first column of the CSV file. Each different language is added in additional columns. The first row of the CSV file indicates the name of each column. You can specify in this field the name of the column that must be used to translate the application by default, when launching the application. If this field is left in blank, the application is launched with its original language. The user can set a different language during the runtime by executing the `SetTranslationFile()` function.
- * **Ignore space chars at the end and beginning of the text:** When this option is checked, the space character at the end or at the beginning of each text is ignored for translation purposes. This option is useful to avoid duplicated entries in the translation table due to space chars configured by mistake when creating the objects or just for alignment.
- * **Keep original text when translation is blank:** When this option is checked, the original text is kept when there is not any text configured as a translation for it in the column currently set for translation of the application. If this option is unchecked, the original text is omitted from the application during the runtime, when there is not any translated text for it in the currently set for translation of the application.

Configuring Object Properties for Screen Objects

To enable translation for individual screen objects, use the following procedure:

- ✓ Create the text and screen objects for your application using the toolbars described in “Using Objects and Dynamics” beginning on page 7–8.
- ✓ When you open the *Object Properties* dialogs to specify the parameters for each object, verify that the **Extern translation** check-box is enabled (☑).

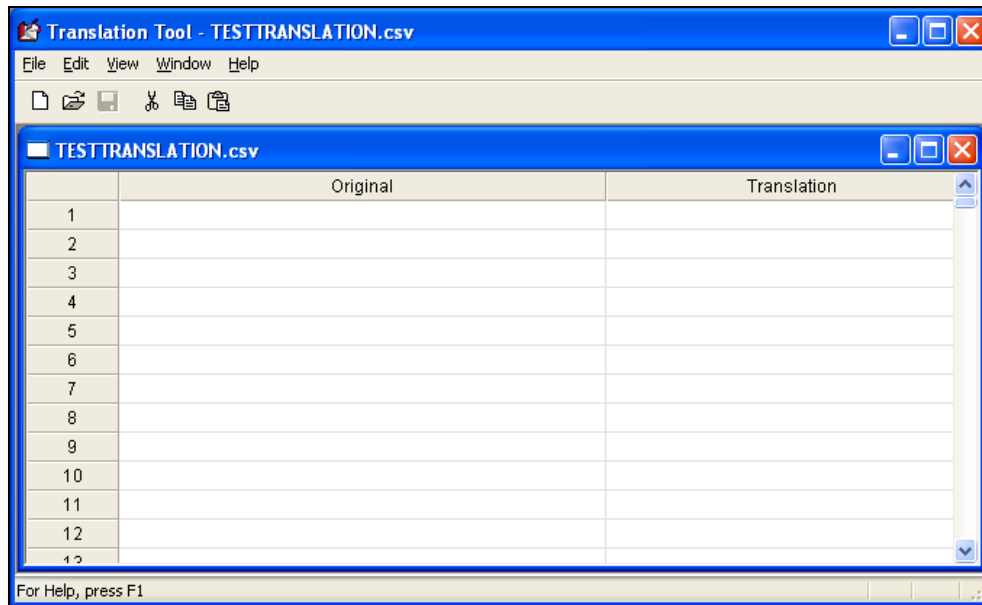
For example, in this figure **Extern translation** is enabled for the **OPC** button object:



Translation Enabled for the OPC Button

Translation Editor

The *Translation Editor* can be launched by the **Tools > Translation Editor** menu option:



Translation Tool

The file specified in the **Translation File Name** field from the **Options** tab of the *Project Settings* dialog (**Project > Settings**) is open by default. You can open a different translation file by the **File > Open** option from the menu of the *Translation Editor*. Moreover, you can create a new translation file with the **File > New** option from the menu of the *Translation Editor*.

The first column is reserved for the text in the original language used when creating the application. Although you can add text in this column manually, you can also import the text from your application automatically into this column by executing the command **Import String Application** available in the **File** menu of the *Translation Editor*.

⇒ **Tip:**

You can execute the command **File > Import String Application** to update the current worksheet on the *Translation Editor* as many times as necessary. Whenever this command is executed, the text available in the application which are not already included in the worksheet are inserted on it. However, this command does not remove any text from the worksheet, regardless of how it had been inserted on it (manually or automatically).

You can write the translation for each different language in additional columns (not in the first column). You can use the following options from the **Edit** menu of the *Translation Editor* to configure the columns of the translation file:

- **Insert Column** (shortcut F9): Insert a new column on the translation file.
- **Rename Column** (shortcut F10): Allows you to rename the column currently selected.
- **Delete Column** (shortcut F11): Delete the column currently selected.

After editing the translation file, you can use the **File > Save** or **File > Save As** options from the menu of the *Translation Editor* to save the settings into the translation file (CSV format).

Since InduSoft Web Studio supports translation files in the standard CSV format, you can either use the *Translation Editor* or any other editor for CSV files, such as Microsoft Notepad or Microsoft Excel to create the translation file. The first row of the CSV file indicates the name of each column. The columns are separated using the comma character (“,”).

Although you can create more than one translation file, it is recommended keeping the translation for all languages in only one file because it will be easier to keep it updated when modifying the application – using the **File > Import String Application** just once, you can update the translation file.

 **Notes:**

For legacy reasons, the *Translation Tool* supports files with the extension **.TRA**. The translation files saved with this extension use the pipe character (“|”) instead of the comma character (“,”) to separate the text between columns.

Editing Worksheets

To open a worksheet for editing:

- Select **File → Open**.
- When the *Open* dialog displays, locate and select the worksheet you want to edit.
- Click **OK** to close the dialog.

 **Save Typing Time:**

After typing terms into a worksheet, you can select **File → Save As** to save a copy of the worksheet under a new name, or select **Window → New Window** to open a new window containing a duplicate worksheet, which you can then save under a new name. Then, in the new worksheet, you can keep the terms in one column and replace the terms in the other column.

There are several ways to edit a *Translation* worksheet:

- You can change or delete individual terms
- You can insert lines
- You can copy from one worksheet to another

CHANGING OR DELETING TERMS

The procedure for changing or deleting terms in a worksheet is as follows:

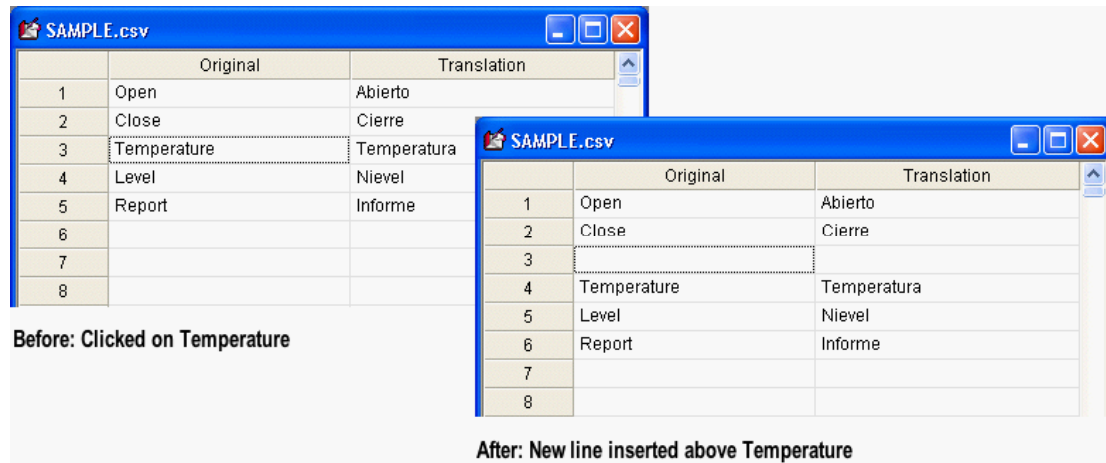
- When the selected worksheet displays in the *Translation Editor*, double-click on the term you want to change and type a new term in its place. Press **Enter**.
- Save your changes to the worksheet by selecting **File → Save** or **File → Save As**.

 **Note:**

Although you can save translation worksheet files to any directory, we strongly recommend saving your files in your project’s *Web* folder to make them available for use on a Web Thin Client.

INSERTING LINES

If you wish to insert one or more terms between existing terms in a worksheet, place your cursor in a cell, select **Edit** → **Insert Line**, and IWS will insert a blank line *above* the existing cell. For example:



Inserting a Line

INSERTING COLUMNS

You can insert a column in basically the same way that you inserted a line, although it does not matter where you place your cursor because the new column will always be appended to the right of the existing columns. Just select **Edit** → **Insert Column**, and IWS will insert the new column to the right. For example:

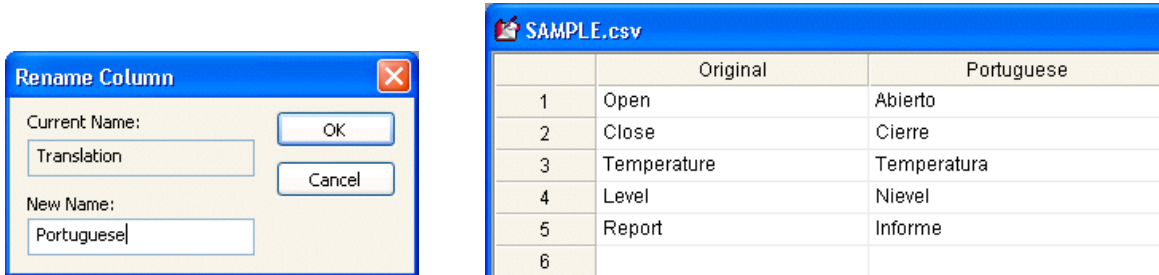
The screenshot shows the IWS application window titled 'SAMPLE.csv' with a table that now has three columns: 'Original', 'Translation', and 'Translation 3'. The rows are as follows:

	Original	Translation	Translation 3
1	Open	Abierto	
2	Close	Cierre	
3	Temperature	Temperatura	
4	Level	Nivel	
5	Report	Informe	
6			
7			
8			

Inserting a Column

RENAMING COLUMNS

You can rename your columns in order to make it easier to identify the translations. Place your cursor anywhere in the column you want to rename, and then select **Edit** → **Rename Column**. The *Rename Column* dialog is displayed; enter the new name in the **New Name** field and click **OK**. For example:





Renaming a Column

COPYING TERMS BETWEEN WORKSHEETS

The *Translation Editor* allows you to cut or copy terms from one *Translation* worksheet and paste them into another. You also can copy terms from an Excel file and paste them into a *Translation* worksheet.

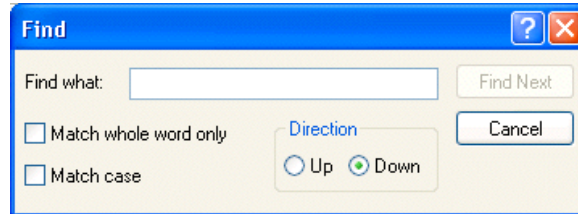
To copy terms from one worksheet (or file) to another, use the following steps:

- ✓ Open both worksheets.
- ✓ Locate the term(s) you want to copy and select them with your cursor.
- ✓ You can select multiple entries within a column and copy those entries into another column.
- ✓ Copy the term(s) by selecting **Edit** → **Copy**, pressing **Ctrl+C**, or clicking .
- ✓ In the second worksheet, select the cell(s) where you want to put the new terms, and paste the term(s) by selecting **Edit** → **Paste**, pressing **Ctrl+V**, or clicking .
- ✓ Repeat the previous steps until you have pasted all the terms you want to share.
- ✓ Save your changes.

SEARCHING THE WORKSHEET

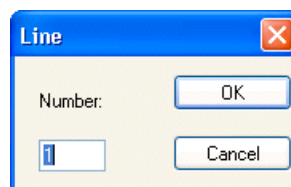
If your worksheet contains a lot of terms (you can enter up to 8,193 terms), you can use the **Edit** → **Find** or **View** → **View Line** options to search for terms.

- Selecting **Edit** → **Find** opens the *Find* dialog:



Find Dialog

- Type a term into the **Find What** text box, and specify one or more of the following *optional* search parameters:
 - * **Match whole word only:** Enable () this box to prevent IWS from finding a term embedded within another term. For example, if you did not enable this option and tried to search for “and,” IWS might find *operand*, *expand*, or *standard*.
 - * **Match case:** Enable () this box to search for a term using the same capitalization you typed into the **Find What** text box. For example, if you did not enable this option and tried to search for “TankLevel,” IWS might find *TANKLEVEL* or *tanklevel*.
 - * **Up and Down:** Enable this button to control the direction in which the program searches for your term. The search commences from the active line. For example, if there are 345 lines in the entire worksheet and you currently on line 325, IWS will search lines 325 through 345 only.
- Click **Find Next** to start (and continue) searching for a term.
- Use the **View** → **View Line** option to jump to a particular line in the worksheet. When the *Line* dialog displays, type the line number into the **Number** field and click **OK**.



Line Dialog

RESTORING DEFAULTS

After resizing a *Translation* worksheet or columns in a worksheet, you can select **Window** → **Restore Defaults** from the *Translation Editor* menu bar and the window (or column) will revert to its original display size.

Saving Your Worksheets

To save your work after creating or editing a *Translation* worksheet, select **File** → **Save** or **File** → **Save As** from the *Translation Editor* menu bar.

Executing the Translation Functions

After enabling translation for the application and the screen objects, and creating/editing *Translation* worksheets, you can execute the following IWS functions:

- **SetTranslationFile**(*strFileName*, *optStrColumnName*): Specifies the *Translation* worksheet file and translates all enabled text within the application during runtime.
- **Ext**(*strText*): Specifies a text string (not related to a screen object) to be translated. For example, the result of an expression to be viewed by an end user.

You can execute both functions synchronously, and they will search the application and change the specified text/objects into the new language. Both functions are supported on Windows 2K/XP/Vista/CE and Web Thin Clients platforms.

Note:

You can save translation files to any directory; however, we strongly recommend saving these files in your project's *Web* folder so they can be used by a Web Thin Client.

For example, if your default development language is English and you want to make the file available to Web Thin Clients, you might change the default worksheet name to **English.csv** and save the file in the following folder:

```
C:\Program Files\InduSoft Web Studio v6.1\Projects\<Project Name>\Web
```

Using the SetTranslationFile() Function


You can use the **SetTranslationFile()** function to translate all enabled screen objects within an application during runtime. (This assumes that you've already created a translation file with the appropriate columns/languages and saved the file in the project directory.)

To execute this function,

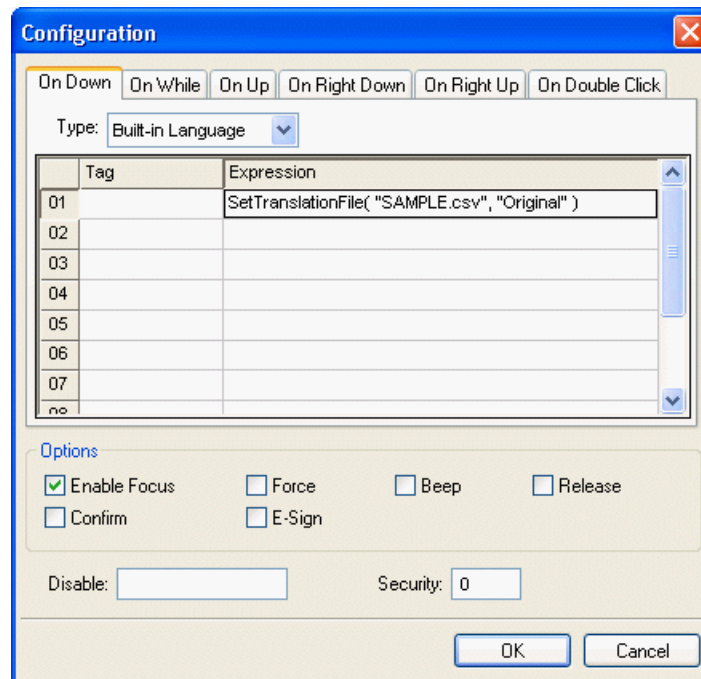
- ✓ Create two buttons (or some other screen objects).
- ✓ Double-click on each button to open the *Object Properties* dialog and specify a meaningful name for each (for example, *English* and *Spanish*).



Create Two Buttons

- ✓ Select the first button and apply the **Command** () property.
- ✓ Double-click on the button to open the Object Properties dialog.
- ✓ Click **Config** to open the expanded *Configuration* dialog.
- ✓ Select the **On Down** command tab.
- ✓ Choose "Built-in Language" from the **Type** menu.
- ✓ Type **SetTranslationFile**(*strFileName*, *optStrColumnName*) into the **Expression** field — substituting the name of the translation file for *strFileName* and the name of the column for *optStrColumnName*. Because these values are Strings, remember to put them in double-quotes.

For example, `SetTranslationFile("SAMPLE.csv", "English")`




Setting the Translation File

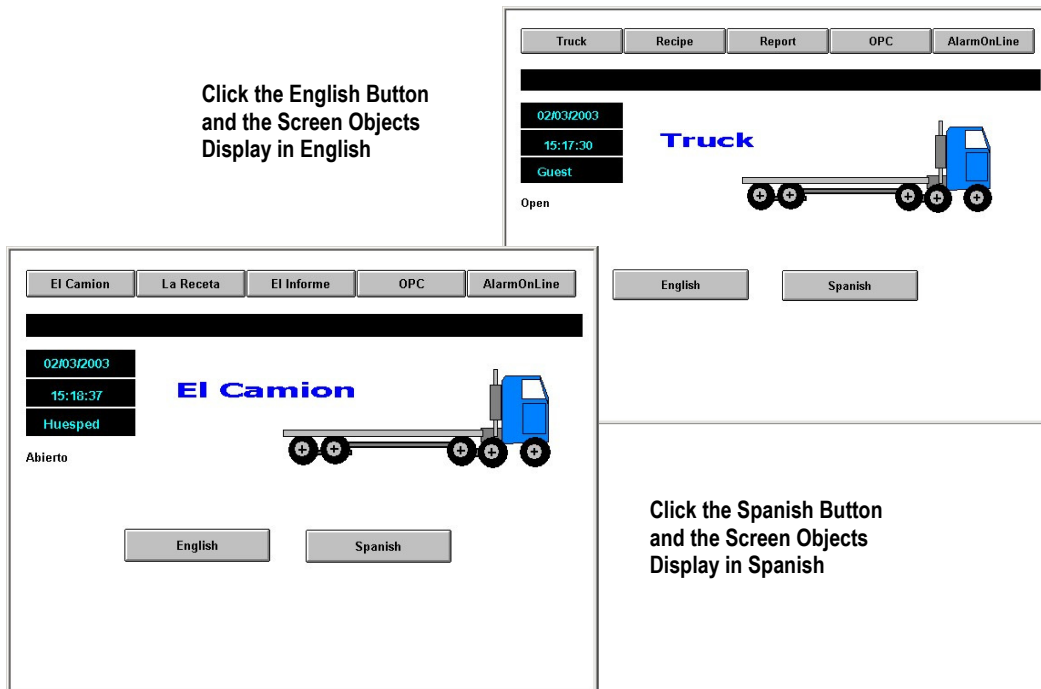
Note:

If you saved these *Translation* worksheets in your project's *Web* folder to make the files available for Web Thin Clients), you also must type the directory path to that file. For example,

`C:\Program Files\InduSoft Web Studio v6.1\<Project Name>\Web\<filename.csv>`
 or type (`"\Web\<filename>.csv") .`

- Apply the **Command** property to the second button and configure the function as described above, replacing the column name as needed
 For example, `SetTranslationFile("SAMPLE.csv", "Spanish")`

Now, when you run the application () you can click the buttons to toggle between the two languages, as shown in the following screens:



Translating between English and Spanish

If you want to verify that the translation was successful, you can create a tag (for example, `STATUS`) and add it to your application screen. When you run the application, the tag will return one of the values listed in the following table.

Returned Value	Description
0	Success
1	Invalid number of parameters
2	Wrong parameter type
3	Translation file could not be found or opened

Using the `Ext ()` Function

You can use the `Ext ()` function to translate text strings within an application, such as the results of an operation that you want displayed to an end-user.

To execute this function,

- ✓ Open an IWS worksheet (for example a *Math* worksheet), and type a string-type tag in the **Tag Name** column.
- ✓ Double-click on each button in the display to open the *Object Properties* dialog and specify a meaningful name for each (for example, *English* and *Spanish*).

The `Ext (strText)` function returns the text translation using the active *Translation* worksheet.

For example,

```
Ext("Start") // Returned value in Spanish = "Comience"  
Ext("Stop") // Returned value in Spanish = "Pare"  
Ext("StrTag")
```

Closing the Translation Editor

To close the Translation Editor, select **File** → **Exit** from the *Translation Editor* menu bar.

- If you have already saved the open *Translation* worksheet(s), the *Translation Editor* will close immediately.
- If you have unsaved changes in an open *Translation* worksheet(s), IWS will prompt you to save your changes.
 - Click **Yes** to save the changes and close the window.
 - Click **No** to close the window without saving your changes.
 - Click **Cancel** to keep the window open and continue working on the open worksheet.

Chapter 17: IWS 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 Studio or to external SQL Relational Databases.

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 Test Table** for the list of databases already tested).

The following tasks support the database interface:


- **Alarms:** The application can save and/or retrieve the alarm history messages in a relational database.
- **Events:** The application can save and/or retrieve the event messages in a relational database.
- **Trends:** The application can save and/or retrieve the Trend history values in a relational database.
- **Viewer:** Database information can be displayed both in table format (*Alarm* and *Grid* objects) or in a graphical format (*Trend* object).
- **Web:** Because the items listed below are already available in IWS Web interface, you can deploy an application 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:

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

Conformance Test Table

 **Note:** For information about how to configure a specific database, please refer to the

Appendixes.

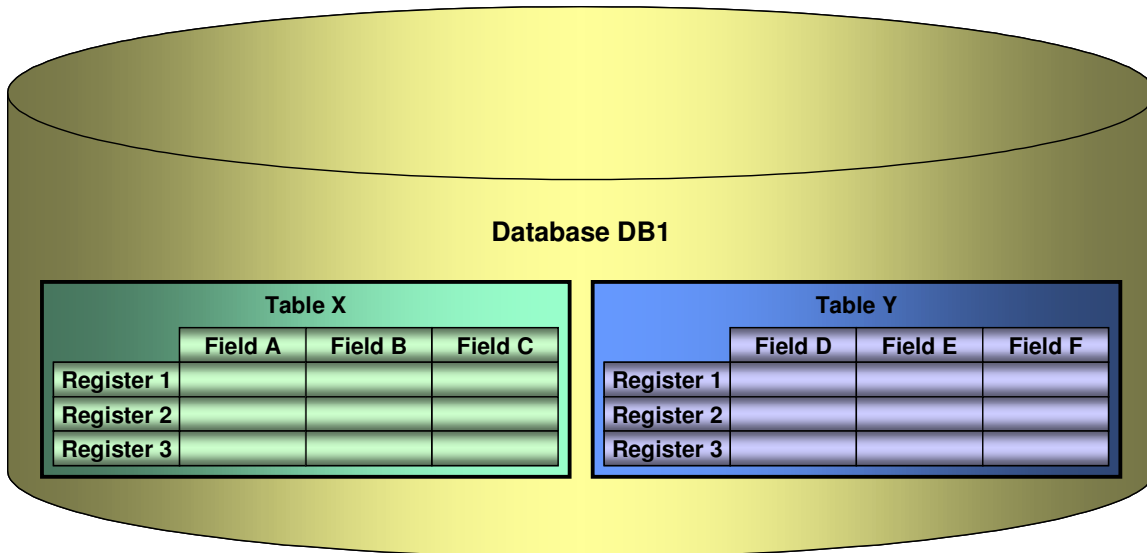
General Concepts

This section describes databases, database providers and the way IWS interfaces with different databases.

SQL Relational Databases

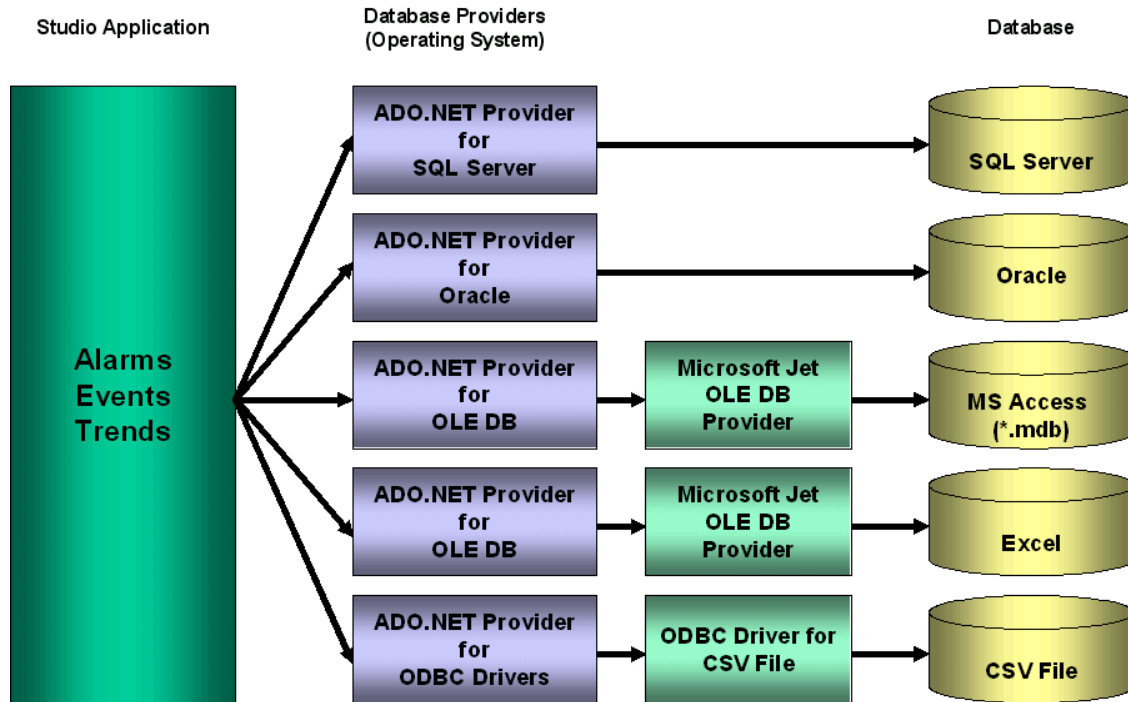
A SQL Relational Database is a set of information stored in tables with fields and registers, which support SQL commands.

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 application 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 applications 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 application settings are mostly uniform, regardless of the specific database chosen by you.

History Format

The IWS tasks that can generate history data (Alarms, Events and Trend) can be configured to save data either in the Proprietary history file format from IWS or to an external SQL Relational database. You can choose the history file format by the **History Format** combo-box available for each task. The following table shows the options available for each task:

Task	History Format	Settings
Alarms	Proprietary	<p>File Format: Text (UNICODE). IWS uses the vertical bar character () to separate the fields.</p> <p>Default Path: ...\<application ,="" path>\alarm\alyymmdd.alh="" where:<br=""></application>YY = Two last digits of the year MM = Month DD = Day.</p>
	Database	<p>Database Type: Chosen by the user</p> <p>Default Table Name: AlarmHistory</p>
Events	Proprietary	<p>File Format: Text (UNICODE). IWS uses the vertical bar character () to separate the fields.</p> <p>Default Path: ...\<application ,="" path>\alarm\evyyddmm.evt="" where:<br=""></application>YY = Two last digits of the year MM = Month DD = Day.</p>
	Database	<p>Database Type: Chosen by the user</p> <p>Default Table Name: EventHistory</p>
Trend	Proprietary	<p>File Format: Binary</p> <p>Default Path: ...\<application ,="" path>\hst\ggyyddmm.hst="" where:<br=""></application>GG = Trend group number (in hexadecimal format) YY = Two last digits of the year MM = Month DD = Day.</p>
	Database	<p>Database Type: Chosen by the user</p> <p>Default Table Name: TRENDGGG (GGG = Trend Worksheet Number – e.g. TREND001 for the Trend Worksheet 001)</p>

Primary and Secondary Databases

IWS supports redundant systems. Therefore, when configuring the database interface, you can configure the Primary Database and, optionally, the Secondary Database. These can be configured 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:**

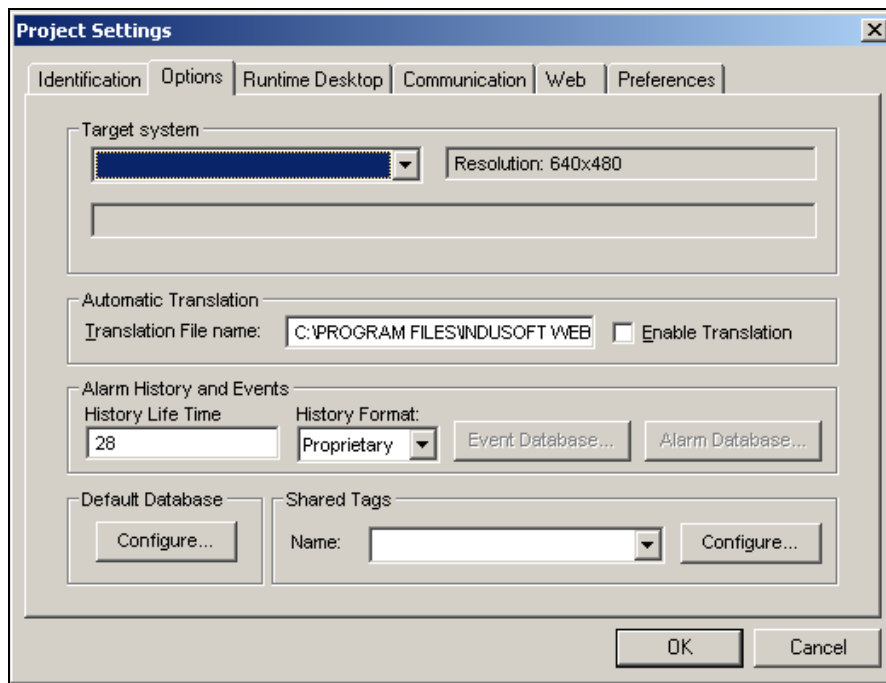
The Primary and Secondary can be different types of databases. However, they must have the same fields.

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).

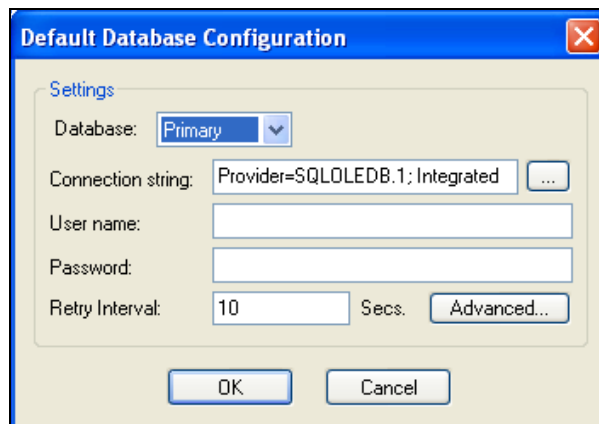
Default Database

Although IWS allows you to configure a different database for each task, typically the same database type (e.g. SQL Server, MS Access, Oracle, and so forth) is used by all tasks of the same project. Therefore, in order to save time when configuring the application, IWS allows you to configure the *Default Database*. When configuring each task, you can choose to use the settings configured for the Default Database. If you choose this method, it will not be necessary to re-configure the same settings for each task, since they share the same database.

The settings for the Default Database can be configured by pressing the **Configure** button from the **Default Database** box on the **Options** tab of the **Project Settings** dialog.



By clicking on this button the following Window will display:



0

Please refer to the section *Configuring Database Settings* for information about the fields on this Window.

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 application is run on the Windows CE operating system (currently, most of the Providers are not supported for the Windows CE operating system). In order to solve this problem, InduSoft designed a flexible solution that allows you to configure distributed systems, as illustrated in the picture below:



The application is running in the Studio Application station (where IWS and/or CEView are/is installed). The application 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 IWS 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 Studio Database Gateway, but they should be changed only under special circumstances. See the “Studio Database Gateway” for information on how to configure the Studio Database Gateway advanced settings.

⇒ **Tip:**


The IWS Database Gateway is a TCP/IP Server for the IWS application 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`

Configuring Database Settings

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 Studio or to external SQL Relational Databases. When selecting Database as the History Format, the database interface settings can be configured through the following interfaces:

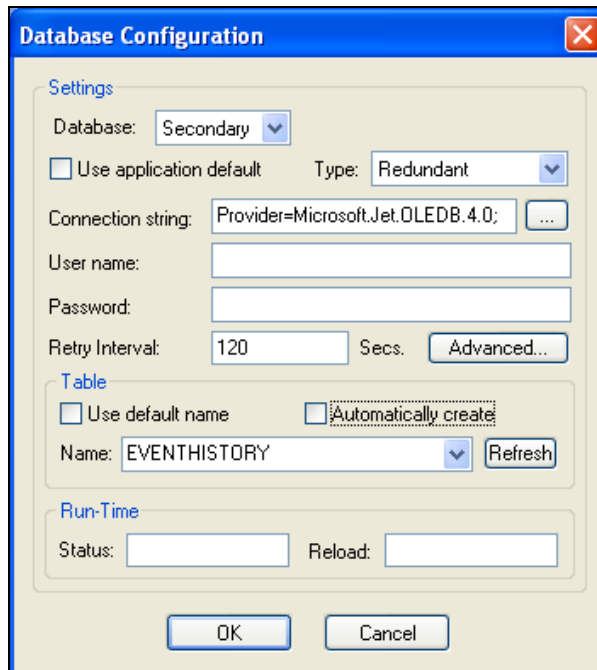
Task	Interface
Alarms	<ul style="list-style-type: none"> ▪ Select the Project > Settings menu. ▪ Select the Options tab on the Project Settings dialog window. ▪ Choose Database in the History Format combo-box. ▪ Click on the Alarm Database button. ▪ Configure the database settings on the Database Configuration dialog.
Events	<ul style="list-style-type: none"> ▪ Select the Project > Settings menu. ▪ Select the Options tab on the Project Settings dialog window. ▪ Choose Database in the History Format combo-box. ▪ Click on the Event Database button. ▪ Configure the database settings on the Database Configuration dialog.
Trend	<ul style="list-style-type: none"> ▪ Create or open a Trend worksheet. ▪ Choose Database in the History Format combo-box. ▪ Click on the Database Configuration button. ▪ Configure the database settings on the Database Configuration dialog.

-  **Note:**
- Both **Alarms** and **Events** are saved in the IWS proprietary format, or both **Alarms** and **Events** are saved in external Relational Databases; however, they can be saved on different databases.
 - Each **Trend** worksheet can be configured to save data either in the IWS proprietary format or in an external SQL Relational Database.

Database Configuration Dialog

The Database Configuration dialog allows you to configure the necessary settings to link IWS to an external SQL Relational Database.

The following picture illustrates the Database Configuration dialog:



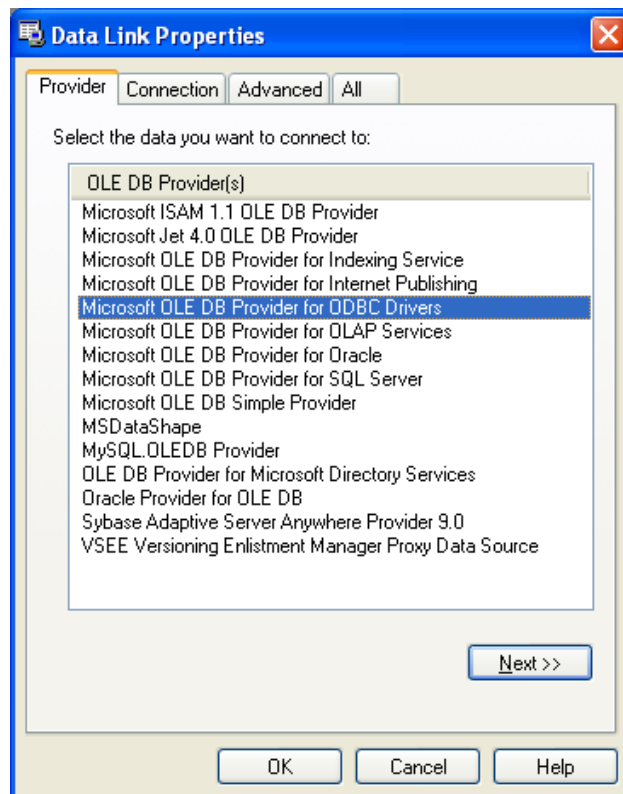
Database Configuration Dialog

- **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.

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 application default** check-box: 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.

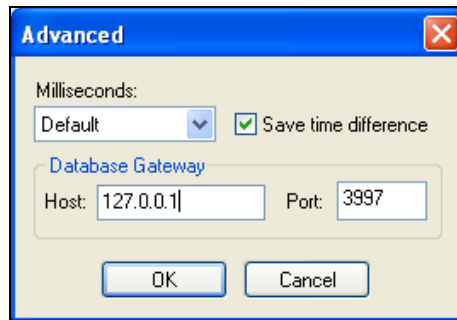


Note:

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 applications, the default value of these settings do not need to be modified and should be kept.



- **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 by InduSoft, the **Default** option attempts to save the milliseconds in a separate field.

⇒ **Tip:**

The default option for each database is configured in the StudioADO.ini file, stored in the \BIN sub-folder of IWS. See “Studio Database Gateway” for information about how to configure the StudioADO.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** check-box: When this option is checked (default), IWS saves the Time Zone configured in the computer where the application 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 Studio 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 Studio database gateway with the same TCP Port. See “Studio Database Gateway” for information about how to configure the advanced settings for the Studio ADO Gateway.

- **Disable Primary Key:** 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), you should check this field.
- **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. InduSoft 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** check-box: 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** check-box: 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 datas will be saved.
- **Refresh** button: If the database configured is currently available, you can press the **Refresh** button to populate the **Name** combo-box with the name of the tables currently available in the database. In this way, you can select the table where the history data should be saved instead of writing the Table name manually in the **Name** field.
- **Run-Time** pane: This area allows you to enter with IWS tags. The following fields are available:
 - **Status** (output) check-box: The tag in this field will receive one of the following values:

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

- **Reload** (output): If you are using in curly brackets in any of the configuration fields, you have to specify the reload tag. 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 and will set the tag back to 0 when it is finished.

Studio Database Gateway

The Studio Database Gateway is a TCP/IP server that interacts with databases using the Microsoft *.NET Framework 1.1*. It can run on the same computer that is running the IWS application, or on a different computer. The Database Gateway Host in the Advanced Settings (see “Database Configuration Dialog” 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 **StudioADO.ini** from the IWS BIN folder to the remote computer, then execute the **StADOSvr.exe**.

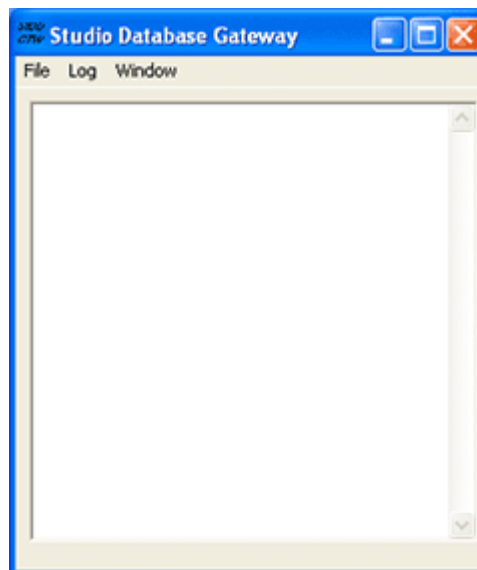
The gateway can be started multiple times for different TCP/IP port numbers. The default port number is 3997, and it is changed by specifying the desired port number in the command prompt (e.g. **StADOSvr 1111**). When running the StADOSvr, it will add the following icon to the tray bar:



When you right-click on the tray bar icon, the following options will display:



The Hide option defines whether the debug window will be displayed or not. If you de-select it, the following window will display:

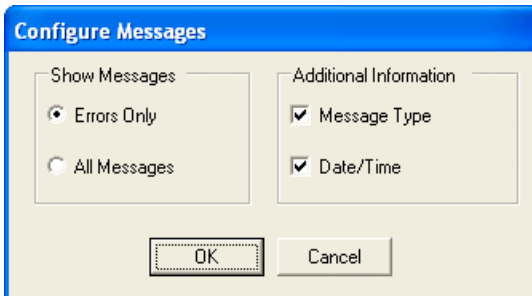


Any failure that occurs during operations with databases will be displayed both in this window and also in the *LogWin* window. The messages are reported by exceptions

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 Studio 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.
 - **Remote IP Address** checkbox: Click (check) this option to show the database gateway’s IP address.

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 parameters or add new providers to the file. The following parameters are available:

Section of .INI File	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 <i>Database Configuration Dialogs</i> .
	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.
	DataAdapterClass	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
Providers	TableDelimiterPrefix	Any character or group of characters	Specify a character that will be placed before table names on SQL statements

Section of .INI File	Parameter	Accepted Values	Description
	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.
	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.

Section of .INI File	Parameter	Accepted Values	Description
Providers	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 application.
	TimeStampType	Any string representing a valid datatype for the database	When trying to create columns to store TimeStamp values, the datatype 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 requested.
	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
	TimeOut	5	Time out to perform connection and query updates
	TimeOut	60	Time out to perform synchronization
Connection	RegBufSize	128	Size of the internal buffer used by the database API.

The parameters are grouped into three sections — Providers, Communication, and Connection — 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 parameter on the file lists the number of providers:

```
Count=5
```

The providers are identified by the “Provider” parameter followed by a number. When connecting to a database, the Provider parameter in the connection string is compared to the provider’s identification, in order to determine which provider will be used. If there is no provider with the value on the connection string, all the default values are assumed. Besides its identification, each provider can have its own value per each parameter. Again, if no value is specified, the default is used. Below is an example with seven providers:

```

Count=7

Provider1=MICROSOFT.JET.OLEDB
SaveMSec1=3
ColumnDelimiterPrefix1=[
ColumnDelimiterSuffix1=]
SingleConnection1=1

Provider2=SQLOLEDB
ConnectionString2=System.Data.SqlClient.SqlConnection
DataAdapterClass2=System.Data.SqlClient.SqlDataAdapter
CommandBuilderClass2=System.Data.SqlClient.SqlCommandBuilder
ColumnDelimiterPrefix2=[
ColumnDelimiterSuffix2=]
TableDelimiterPrefix2=[
TableDelimiterSuffix2=]
RealType2=FLOAT

Provider3=MSDASQL
ConnectionString3=System.Data.Odbc.OdbcConnection
DataAdapterClass3=System.Data.Odbc.OdbcDataAdapter
CommandBuilderClass3=System.Data.Odbc.OdbcCommandBuilder
ValueString3=?
ValueAddNumber3=0
StringType3=VARCHAR(128)
EnableTop3=0

Provider4=ORACLEDB
Assembly4=System.Data.OracleClient
ConnectionString4=System.Data.OracleClient.OracleConnection
DataAdapterClass4=System.Data.OracleClient.OracleDataAdapter
CommandBuilderClass4=System.Data.OracleClient.OracleCommandBu
ilder
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.AsaCommandBuild
er
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.MySqlCommandBuil
der
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.OracleCommandBu
ilder
ValueString7=Value
ValueAddNumber7=1
ValueStringPrefix7=:
BoolType7=Number(1)
IntegerType7=Number(10)
RealType7=Number
StringType7=VARCHAR(255)
TimeStampType7=TIMESTAMP(0)
EnableTop7=0
```

IWS DEVELOPMENT ENVIRONMENT > THE WORKSPACE > TASKS TAB > ALARMS FOLDER

Alarm summary: When you enable the alarm history file for a group, IWS saves the alarm events to the history database, according to the File Format configured for the Alarm History and Events. The information saved in the history file is described in the following table.

Field Name	Data Type	Remarks
Version	Integer	This field is created only when the File Format is Proprietary. Current version: 003
AI_Start_Time	TimeStamp	Timestamp indicating when the alarm started. When the File Format is Proprietary, IWS saves the Date (MM/DD/YYYY) in one field and the Time (HH:MM:SS.MSS) in the next field.
AI_Tag	String	Tag Name
AI_Message	String	Alarm message
AI_Ack	Boolean	0: Indicates the alarm was acknowledged or does not require acknowledgment 1: Indicates the alarm was not acknowledged
AI_Active	Boolean	0: Indicates the alarm is not active 1: Indicates the alarm is active
AI_Value	Real	Tag value when the alarm event occurred
AI_Group	Integer	Alarm Group Number
AI_Priority	Integer	Alarm Priority Number
AI_Selection	String	Alarm Selection value
AI_Type	Integer	1: HiHi 2: Hi(On) 4: Lo(Off) 8: LoLo 16: Rate(Change) 32: DevP 64: DevM
AI_Ack_Req	Boolean	0: Requires acknowledgement (Ack) 1: Does not require acknowledgement
AI_Norm_Time	TimeStamp	Timestamp indicating when the alarm was normalized. When the File Format is Proprietary, IWS saves the Date (MM/DD/YYYY) in one field and the Time (HH:MM:SS.MSS) in the next field.
AI_Ack_Time	TimeStamp	Timestamp indicating when the alarm was acknowledged. When the File Format is Proprietary, IWS saves the Date (MM/DD/YYYY) in one field and the Time (HH:MM:SS.MSS) in the next field.
AI_User	String	User logged when the alarm event occurred. This field only exists for Version >=1

AI_Comment	String	Comment (optional) typed by the operator when the alarm was acknowledged. This field only exists for Version >=1
AI_User_Full	String	Full name of the user logged when the alarm event occurred. This field only exists for Version >=2
AI_Station	String	Name of the station (computer) where the alarm event occurred. This field only exists for Version >=2
AI_Previous_Tag_Value	Real	Tag value that occurred before the alarm event. This field only exists for Version >=3
AI_Deleted	Boolean	0: Alarm message was not deleted 1: Alarm message was deleted This field is created only when the File Format is Database.
AI_Bias	Integer	Difference (in minutes) from the Time Stamp columns and the GMT time. This field only exists for Version >=3
AI_Last_Update	TimeStamp	Time Stamp 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 File Format is <i>Database</i> .

⇒ **Tip:**

When saving the History Alarms in a SQL Relational Database (File Format = Database), you can customize the name of the columns created in the database by editing the <ApplicationName>.APP file, as follows:

```
[Alarm]
<DefaultName>=<NewName>
```

For example:

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

IWS DEVELOPMENT ENVIRONMENT > THE WORKSPACE > DATABASE TAB > EVENT SETTINGS

Event log files are saved to the history database, according to the File Format configured for the Alarm History and Events. The information saved in the history file is described in the following table.

Field Name	Data Type	Remarks
Version	Integer	This field is created only when the File Format is Proprietary. Current version: 002
Event_Type	Integer	1: SECURITY SYSTEM 2: DISPLAY 3: RECIPE 4: REPORT 5: CUSTOM MESSAGES 6: SYSTEM WARNING 7: LOG TAGS
Event_Time	TimeStamp	Timestamp indicating when the event occurred. When the File Format is Proprietary, IWS saves the Event Time in the following format: MM/DD/YYYY HH:MM:SS.MSS.
Event_Info	String	Tag Name
Value	Real	Tag value when the event occurred
Source	String	Name of the task that generated the event
User	String	User logged when the event occurred.
User_Full	String	Full name of the user logged when the event occurred.
Message	String	Event message
Station	String	Name of the station (computer) where the event occurred.
Comment	String	Comment (optional) typed by the operator when the event occurred. This field only exists for Version >=2
Previous_Value	Real	Tag value that occurred before the event. This field only exists for Version >=2
Deleted	Boolean	0: Event message was not deleted 1: Event message was deleted This field is created only when the File Format is Database.
Bias	Integer	Difference (in minutes) from the Time Stamp columns and the GMT time. This field only exists for Version >=2
Last_Update	TimeStamp	Time Stamp 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 File Format is Database.

⇒ Tip:

When saving the Events in a SQL Relational Database (File Format = Database) you can customize the name of the columns created in the database by editing the <ApplicationName>.APP file as follows:

```
[EventLogger]  
<DefaultName>=<NewName>
```

For example:

```
[EventLogger]  
Event_Info=Information  
Message=Event_Message
```

IWS DEVELOPMENT ENVIRONMENT > THE WORKSPACE > TASKS TAB > TREND FOLDER

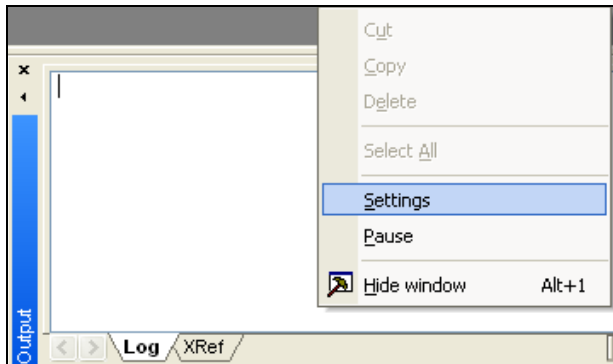
- **Name of history files** pane: Specify the following parameters to define the history file name. You can generate trend historical files in two forms: by date or batch (by events).
 - **Date** (default) check box: Click (check) to generate history files based on the date. Use this option if you have a continuous process. Depending on the options selected in the History Format combo-box, IWS saves the Trend history data either to proprietary binary files or to a SQL 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.
<Tag Name>	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.

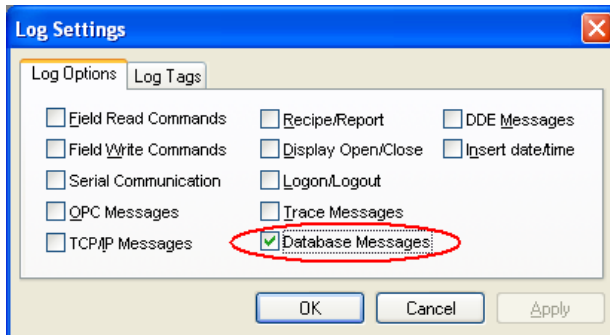
Database Troubleshooting

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 enable the Output Window (View-> Toolbars-> Output).
2. Right click on the Output Window (usually located in the right corner in the development environment), and select **Settings**;



3. In the *Settings Log Settings* window check the **Database Messages** option;



After enabling Database Messages, the *Output* window will display error messages related to the database. The Database FAQ section that follows lists some common errors that you can see in the *Output* window.

Database FAQ

GENERIC QUESTIONS

Q: I configured my database, but the Run-Time 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].

What should I do?

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 Window”), 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: I configured my Connection String using the browser and the Data Link Properties Window. When I click on the Test button, it says “Test succeeded”. However, when I run my application, 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 application works fine when I run in emulation mode. But when I send to the Windows CE device, it cannot communicate with my database.

A: It might be the case that your Windows CE 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: Why am I receiving the message Error to create connection class when I try to connect to the database?

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 *Assembly Cache*?

A: If your assembly has a strong name, you can register it in the *Assembly Cache* using the `gcautil` program. It should work if you copy your assembly to the same folder as the `StADOSvr.exe` (usually the `IWS\Bin` folder).

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 <http://support.microsoft.com/default.aspx?scid=kb;en-us;830173>.

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 it 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 application 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 you can get access to the database. By default, this property is set to `localhost`, so you will only be able to access the database 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. For more information, see “Database Configuration Dialog” on page 17–9.

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 application, I get the following error: Database: Error: Parse error: DSN 'MyDatabase' does not exist. What am I doing wrong?

A: Please refer to 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.Asacconnection]`?

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 shows `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.

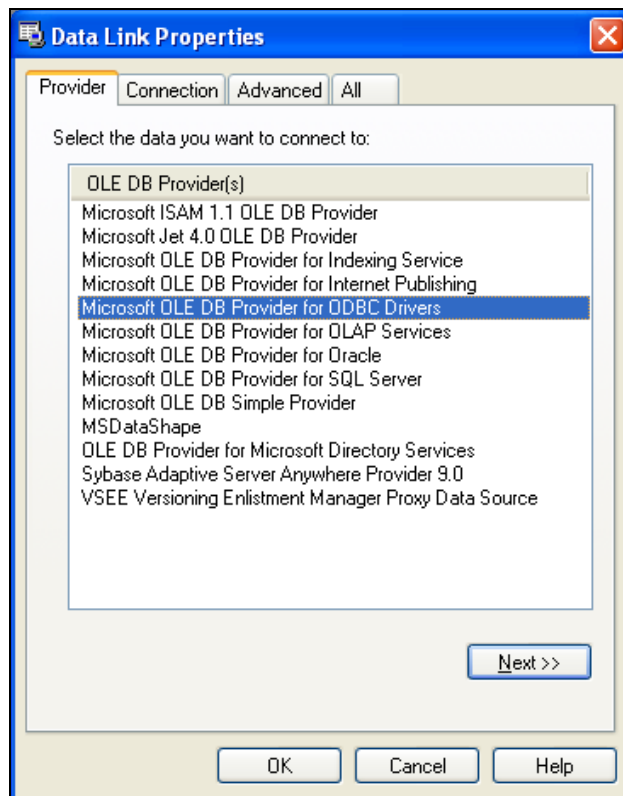
Database Appendix A: Using ODBC Databases

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.

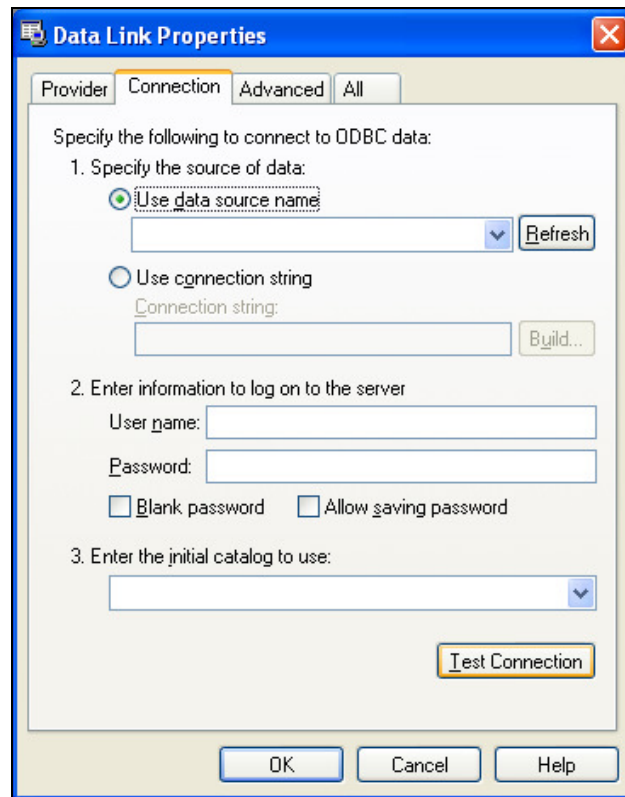
Note:

Microsoft *.NET Framework 1.1* is automatically installed, starting with IWS v.6 Service Pack 3.

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:



By clicking the **Next** button the following window will display:

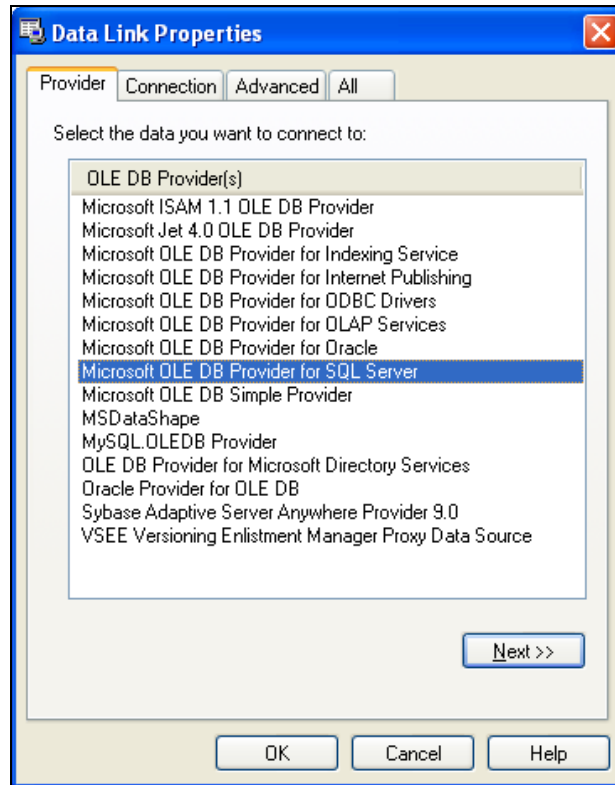


Select the DSN that you want to connect to and click **OK**. If you want to specify the user name and password on this window instead of specifying on the Database Configuration dialog, remember to check the **Allow saving password** checkbox.

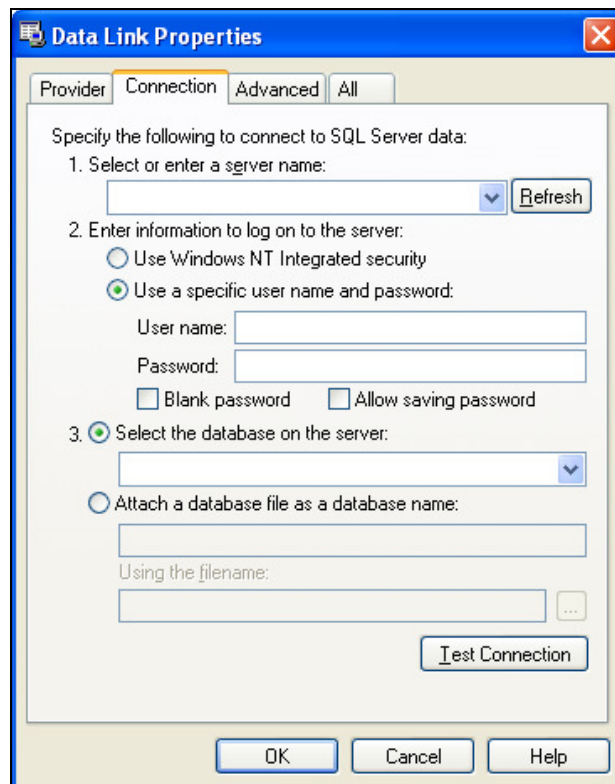
Database Appendix B: Using Microsoft SQL Server

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:



2. Select the Microsoft OLE Provider for SQL Server, and click **Next**. The following window will display:



- 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.
- Click **OK** to finish the Connection String configuration.

Your connection string should be very similar to this one:

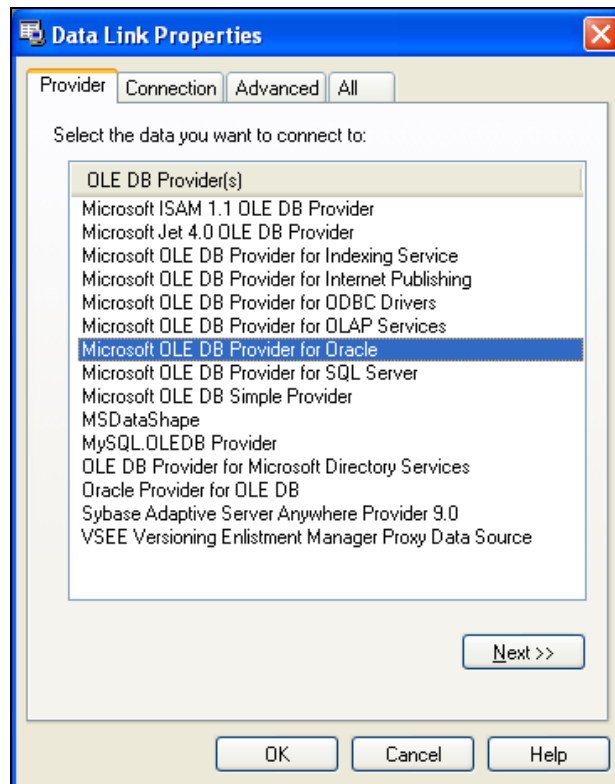
```
Provider=SQLOLEDB.1; Integrated Security=SSPI; Initial
Catalog=MyDatabase; Data Source=192.168.23.200
```

Note:
These procedures were tested using Microsoft SQL Server 2000.

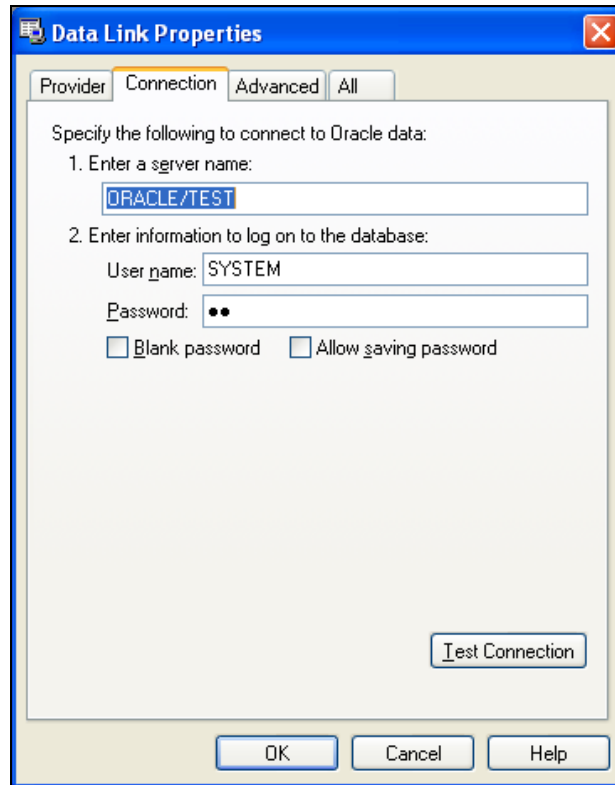
Database Appendix C: Using ORACLE databases

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:

- Click on the **Browse** button in the *Database Configuration Dialog* window. The following window will display:



- Select the Microsoft OLE Provider for Oracle and click **Next**. The following window will display:



- 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 workaroud this problem.

- Click **OK** to finish the Connection String configuration.

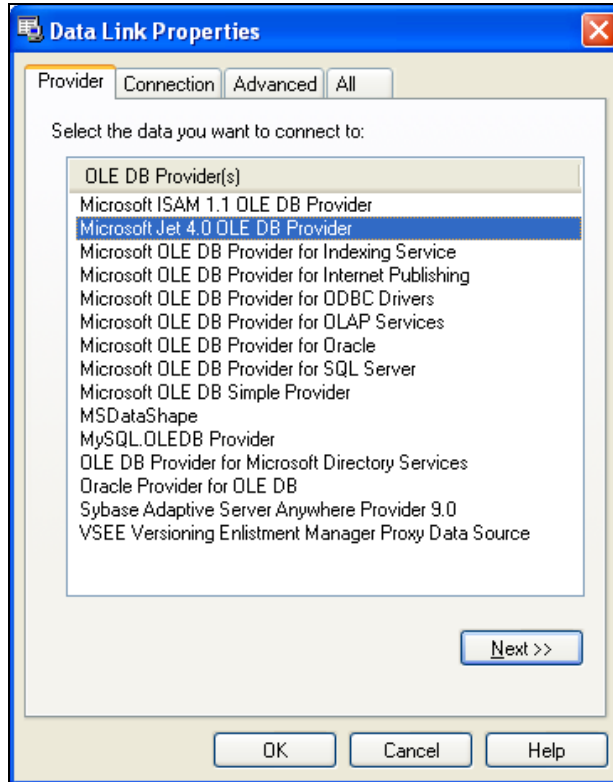
Note:

These procedures were tested using ORACLE 10g Release 1.

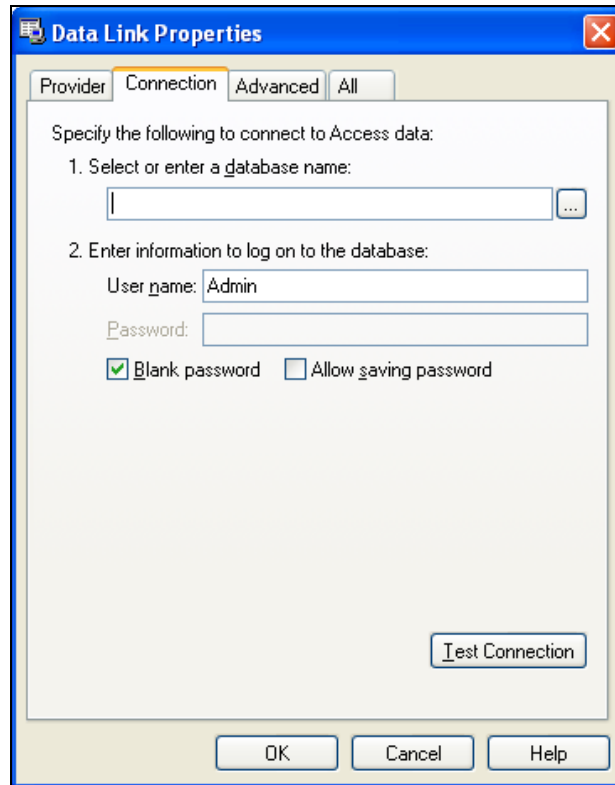
Database Appendix D: Using Microsoft Access Databases

IWS Database interface allows you to retrieve and store information on Microsoft Access Database files. You should follow the steps below in order to set up an access database:

1. Click on the **Browse** button in the *Database Configuration Dialog* window. The following window will display:



2. Select the Microsoft Jet 4.0 OLE DB Provider, and click **Next**. The following window will display:



3. Type the name of or select the database that you want to access.

Caution:

Databases such as Microsoft Access are not capable of handling large amounts of data efficiently. Therefore, if you try to store all your historical data in this type of database, the queries might be very slow, and you might not get the expected results. If you need to store large amounts of data, we recommend that you either use the proprietary format or a powerful relational database such as Microsoft SQL Server or ORACLE. Microsoft Access is recommended as a secondary database, using the option **Store and Forward**, or to exchange information with third party software.

Note:

These procedures were tested using Microsoft Access 2000 (9.0.3821).

Database Appendix E: Using SQL Server CE

IWS can interface with SQL Server CE databases by using the SQL Server CE provider. This provider is available only for Windows CE, and it must be installed on your Windows CE device. Because this provider is only available for Windows CE, you have to enter the **Connection String** manually. The **Connection String** should have the following format:

```
Provider=SQLCE; Data Source = <Database Path>
```

Examples:

1. Access a fixed database file located in `\Harddisk\MyDatabase.sdf`:

```
Provider=SQLCE; Data Source = \Harddisk\MyDatabase.sdf
```

2. Access a database file in the location indicated by the string tag

```
DatabaseFile:
```

```
Provider=SQLCE; Data Source = {DatabaseFile}
```

 **Caution:**

Databases such as SQL Server CE are not capable to handle large amounts of data efficiently. Therefore if you try to store all your historical data on this type of database, the queries might be very slow and you might not get the expected results. If you need to store large amounts of data, we recommend that you either use the proprietary format or a powerful relational database such as Microsoft SQL Server or ORACLE. SQL Server CE is recommended as a Secondary database using the option Store and Forward or to exchange information with third part software.

Database Appendix F: 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
```

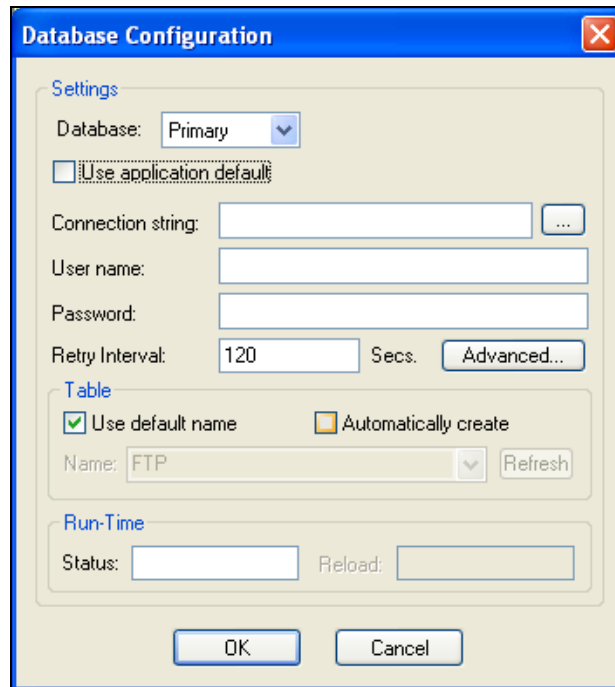
 **Note:**

These procedures were tested using Sybase Server Anywhere 9.0.1.1751.

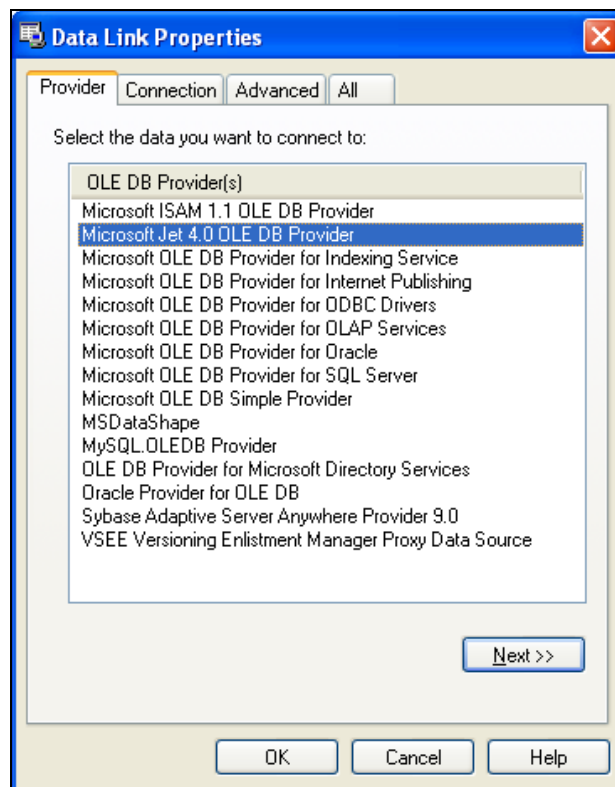
Database Appendix G: Using Microsoft Excel

IWS Database interface allows you to retrieve and store information in Excel files. The main use of an Excel database is with the *Grid* object. The steps below show how to configure the *Grid* object to retrieve data from an Excel file:

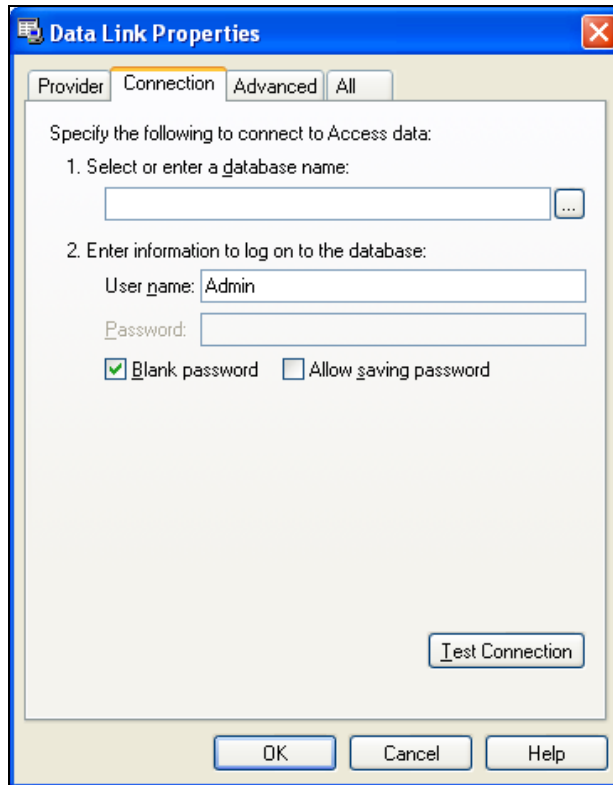
1. Insert a *Grid* object on your screen;
2. Select **Database** in the **Data Source** field.
3. Click the **Data** button. The following window will display:



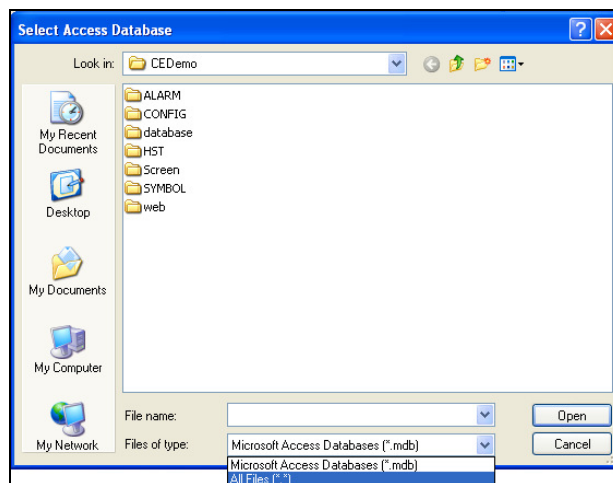
4. Uncheck the **Use application default** check box.
5. Click the **Browse** button to build the Connection String. The following window will display:



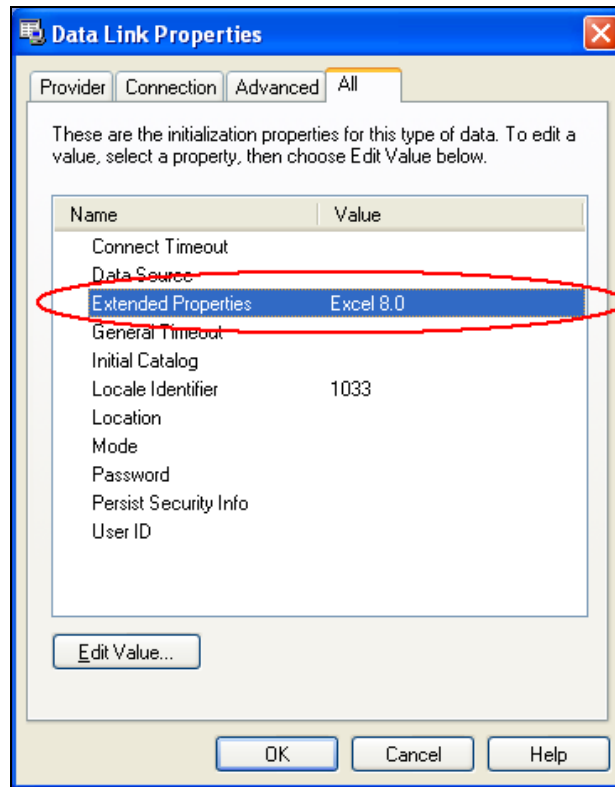
6. Select the Microsoft Jet 5.0 OLE DB Provider, and click **Next**. The following window will display:



7. Click the **Browse** button. The following window will display:



8. Select your Excel file, and click **Open**.
9. Select the tab **All** in the Data Link Properties Window.
10. In the extended properties, enter the value *Excel 8.0*.



Note:

The value in this field might change if you are using a different Excel version.

11. Click **OK**, and you will have a connection string very similar to this one:

```
Provider=Microsoft.Jet.OLEDB.4.0; Data Source=C:\Book1.xls;
Extended Properties="Excel 8.0"
```

Tip:

You can enter a tag between curly brackets to specify the Excel file that you want to access. In the example below, the tag TagExcelFile is used to indicate the Excel file:

```
Provider=Microsoft.Jet.OLEDB.4.0; Data
Source=C:\{TagExcelFile}.xls; Extended Properties="Excel 8.0"
```

12. Specify the worksheet that you want to access in the **Table** field. The worksheet name should have the dollar sign (\$) at the end, and it must be between square brackets ([]). The picture below shows the configuration to access a worksheet called Sheet1:

→ **Tip:**

You can enter a tag between curly brackets to specify the worksheet name (e.g. [{TagExcelWorksheet} \$])

🔗 **Note:**

These procedures were developed based on information provided on the Microsoft Knowledge Base Article 278973. They were tested using Microsoft Excel 2000 (9.0.3821).

Database Appendix H: Using MySQL

IWS Database interface can interface with MySQL databases; however, you must install the .Net provider for it in order have access. The provider required by IWS is the ByteFX.MySqlClient. At the time that this document was written, it could be downloaded from www.sourceforge.net

The connection string should have the following format:

```
Provider=MySqlProv; Data Source=MyDatabaseName;
Location=192.168.23.200
```

🔗 **Note:**

If you have the OLE DB Provider for MySQL installed on your computer, you can use the browse button on the *Database Configuration Dialog* window to build your connection string.

The ByteFX.MySqlClient connection uses different keywords for the Data Source and for the Location parameters in the connection string. The example above is passed to the connection class in the following format:

```
Database=MyDatabaseName; Data Source=192.168.23.200; Pooling=false
```

 **Note:**

These procedures were tested using MySQL v4.0.20a and the ByteFX.MySqlClient v0.76.

Chapter 18: 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**

Check the *Help* files of your product for more information on your issue. Help can be found on the menu bar within the application. For online documentation including *Release Notes* and downloads, see the IWS Web site (www.InduSoft.com). 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.

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.

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 machine have you tested on, and is the issue 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 application crashes completely during runtime, it will generate a debugger report and save it to:

- <Application Directory>\Web\Dump\WindDump.dmp
- Please have this file ready to send to Technical Support for analysis.

If your problem or question is not urgent, you may choose to email technical support. Email sent to support is answered daily (support@InduSoft.com).

If you require additional assistance while using InduSoft Web Studio or this service pack, the following resources are available by telephone at 877-INDUSOFT (877-463-8763)

Related IWS Publications:

- *InduSoft Web Studio Getting Started Guide*: Designed for first-time users, this publication contains information about the basic functions of InduSoft Web Studio. This publication is provided in the *Documentation* folder on the IWS CD-ROM or from the **Help** menu located on the main menu bar.



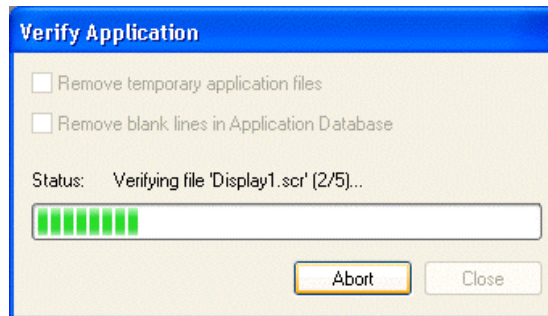
Using the Help Menu

- **Technical Reference**: Opens the Context Sensitive Help.
- **User Guide**: Opens the User Guide Manual.
- *InduSoft Web Studio Technical Reference Manual*: Describes all the features and tools that comprise the IWS development environment and provides detailed instructions for using the product. This publication is provided in the *Documentation* folder on the IWS CD-ROM or from the **Help** menu on the main menu bar.
- *Drivers User Guides*: Explain how to configure individual InduSoft drivers, according to their unique protocol characteristics. One customized user guide is included with each InduSoft driver. These publications are provided in the *DRV* subdirectory of the *InduSoft Web Studio* folder on the IWS CD-ROM or from the **Help** menu located on the main menu bar.

Visit the InduSoft Web Site at www.InduSoft.com.

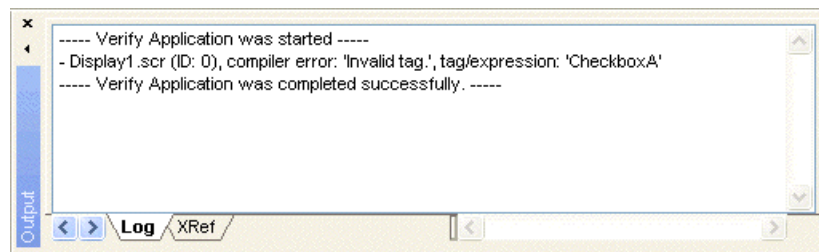
Verifying Your Application

From the menu bar, select **Tools > Verify Application** to recompile all Math worksheets, expressions, and screen logic, as well as to update current HTML files using the settings configured on the **Web** tab of the *Project Settings* dialog.



Verifying an Application

If there are any tags used in the application that are not defined in the database, then IWS will indicate where (i.e., the screen or worksheet file) those tags are used.



Invalid Tag Found in Screen File

⇒ Tip:

When you save a screen or worksheet, IWS includes a pointer to the current database version. When you execute the application, IWS compares the screen or worksheet database to the current application database and if there is a mismatch, IWS recompiles the expressions.

To avoid doing these tasks during application runtime, we recommend running the **Verify Application** tool before downloading and/or finishing an application. You should also use this function when converting an application to a new version of the program.

🔗 Note:

Selecting the **Verify Application** tool will delete all temporary files (e.g., *.txt, *.mac and *.tag files) from the application folder to save disk space and clean the application before downloading it to the runtime station.

Verifying an application *does not* automatically download it to the runtime station. To download the application, you must use the **Send project to target** tool.

Common Errors

Listed below are answers to frequently asked questions about InduSoft Web Studio.

Database & Security System

- **What does the Shared Tags folder store?**

The *Shared Tags* folder stores the tags imported from the PC Based Control linked to the IWS application. The PC Based Control is linked to the IWS application by the *New Project* wizard.

- **How do I count how many tags are configured in the application database?**

From the menu bar, select **Project** → **Status**, and then the **Information** tab to calculate the amount of tags configured in the application. Each array position and each class member of the tags configured in the IWS tag database are counted.

- **How do I see the list of “Users” I’ve added during runtime in my application that I have created with the `CreateUser()` function?**

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 window. You can see the users created by the `CreateUser()` function, and you can then create or delete users.

Graphics

- **How do I insert and configure an ActiveX object in a Studio application?**

To insert an *ActiveX* object in an IWS application:

- Select the menu option **Insert** → **ActiveX object...** or press the **ActiveX Control** button from the **Active Objects** toolbar.
- Select the *ActiveX* object to be inserted in the application from the list box, and press the **OK** button. The *ActiveX* object will then appear on the screen. (Unregistered *ActiveX* objects will not be available in this list box.)
- Double-click on the *ActiveX* object and assign a name to it (enter a value in the **Name** field). The dynamic properties 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 *Methods* 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 *Methods* 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 *Methods* dialog from the object, with the syntax **<Method Name> (Method)** (for example, **OpenFile (Method)**).

⇒ **Tip:**

Before inserting an *ActiveX* component (usually an OCX file) into the IWS application, make sure it has been properly registered in the computer. It's possible to register an *ActiveX* object by IWS. Select the menu option **Tools > Register Controls**, press the **Register...** button and select the *ActiveX* file (usually an OCX file) that must be registered.

🔍 **Note:**

The amount of parameters set in the **XRun ()** function can vary from 0 up to 255 and it depends each the *ActiveX* component. It's possible to use tags to set the parameters; however, the tag type must match the component parameter type (Boolean, integer, string or real).

- **How do I designate one screen that will open each time I start the application?**

Open the **Project Settings** dialog window by the **Project → Settings** menu, select the **Runtime Desktop** tab, and type the startup screen name in the **Startup screen** field.

- **How do I insert a background picture on the screen?**

Right click on the screen and select the option **Screen Attributes** from the popup menu. Enable the check-box **Enable Background** and choose the picture format in the combo-box besides this label. Copy the picture file to the **\Screen** folder of the application 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** folder and share this picture with more than one screen. In this case, it's necessary to type the bitmap name in the **Share image** field.

Tasks

- **How do I convert the History Trend to an ASCII file?**

To convert a **History Trend** file to an ASCII format, copy the file "`<StudioPath>\bin\hst2txt.exe`" to the path "`\<ApplicationPath>\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.

- **How do I exchange data with FOX Pro by an ODBC protocol?**

When exchanging data with FOX Pro database, it's necessary to set the parameter **UseQuote=0** from the [ODBC] section in the `<ApplicationName>.app` file.

- **How do I set a DATE field for an ODBC interface with an Oracle package?**

Configure the "Column" cells in the IWS *ODBC* worksheet with the syntax `<ColumnName>.ts` (for example: `MyDate.ts`).

- **How do I execute a Math worksheet during the startup and another Math worksheet during the application shutdown?**

Startup: Execute a *Math* worksheet during the startup by creating a *Math* worksheet and filling in its **Execution** field with the expression `<TagName>=0` (for example, `StartTag=0`). In the last line of the *Math* worksheet, set the value 1 to the `<TagName>` tag. The `<TagName>` tag type should be Boolean.

Shutdown: Instead of executing the `ShutDown()` function directly, execute one *Math* worksheet and configure the `ShutDown()` function in the last line of this *Math* worksheet.

Communication

▪ **How do I address timeout errors?**

A timeout error (error 1234) typically indicates that runtime instructions are not arriving at the connected PLC. To address the error, you can do the following:

- Verify that the selected driver's communication parameters match the PLC specifications. (See *Configuring a Driver* on page 10–2.)
- Verify the PLC's station number and make sure it is properly configured in the **Station** field of the driver worksheet. (See *Configuring the Driver Worksheets* on page 10–10.)
- Ensure that all wiring is securely connected and the PLC is set to run.

▪ **How do I set a “communication error” alarm?**

Configure a tag in the fields “**Write Status**” or “**Read Status**” of the driver worksheets and configure an alarm whenever this tag is different of 0 (zero).

▪ **How do I communicate with a Siemens S7-200 PLC without using Prosave software?**

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 and Studio Modbus driver (**MODBU**) you can exchange information between them.

▪ **How do I start and stop communication drivers during the runtime?**

There are three functions available to handle the execution of the communication drivers during the runtime:

- Start all drivers configured in the application:

Syntax: **StartTask**("Driver")

For example, **StartTask**("Driver")

- Start a specific driver configured in the application:

Syntax: **WinExec**("<StudioPath>\bin\StudioManager.exe"+ "+ "<StudioPath>\bin\Driver.dll"+ "+ "<DriverName>")

For example, **WinExec**(Asc2Str(34)+"C:\Program Files\InduSoft Web Studio v6.1\BIN\Studio Manager.exe+Asc2Str(34)+" "+Asc2Str(34)+"C:\Program Files\InduSoft Web Studio v6.1\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 application:

Syntax: **EndTask**("Driver<DriverName>")

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").

⚠ Caution:
 The drivers and tasks cannot be started or stopped during the runtime when running CEView (under the WinCE operating system).

▪ **What are the parameters of Studio DDE Server?**

The Studio 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>

▪ **How to exchange data with Excel by using NetDDE?**

NetDDE can be used to exchange data, via the DDE protocol, between networked stations.

- Start the DDEServer module from Studio (**Project > Status > DDE Server**)
- Run Excel in the remote station
- Open a Excel worksheet and fill the cells which must exchange data with Studio using the following syntax: ='\<computer name>\NDDE\$' | 'UNISOFT\$' !<tagname>

For example, ='\PC\NDDE\$' | 'UNISOFT\$' !second

📌 Notes:

- 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>\netdde.exe in both computers (for example, c:\Windows\netdde.exe).

▪ **Is the Studio OPC interface compliant with OPC specification v1.0a or v2.0?**

Studio OPC Client and OPC Server modules are compliant with both OPC specification v1.0a and v2.0.

General Troubleshooting

- **What operating systems are compatible with Studio and CView?**

See table below. The symbol “x” means that the operating system is NOT supported and the symbol “✓” means that the operating system is supported.

Operating System		Studio				CView			
Name	Version	v2.x	v3.x thru v4.1	v4.2 thru v6.0	v6.1 and higher	v3.x	v4.x	v5.x	v6.x
Windows Vista	Any	x	x	x	✓	x	x	x	x
Windows XP	Any	x	✓	✓	✓	x	x	x	x
Windows 2000	Any	x	✓	✓	✓	x	x	x	x
Windows NT	v4.0+SP4 or higher	✓	✓	✓	x	x	x	x	x
Windows ME	Any	x	✓	x	x	x	x	x	x
Windows 98	Any	x	✓	x	x	x	x	x	x
Windows 95	Any	✓	✓	x	x	x	x	x	x
Windows CE	v2.12	x	x	x	x	✓	x	x	x
	v3.x	x	x	x	x	✓	✓	✓	✓
Windows CE.net	v4.0	x	x	x	x	x	x	✓	✓
	v4.1	x	x	x	x	x	x	✓	✓
	v4.2	x	x	x	x	x	x	✓	✓
	v5.0	x	x	x	x	x	x	x	✓
	v6.0	x	x	x	x	x	x	x	✓

- **How do I start IWS automatically when the computer is powered on?**

Create a shortcut to the ...**<Studio Path>\bin\RunStartUp.exe** in the *Startup* folder from the *Programs* group (...**WINNT\Profiles\All Users\Start Menu\Programs\Startup**).

- **How to disable Dr. Watson?**

The procedure step-by-step to disable *Dr. Watson* under Windows NT is described below:

- Execute the program **<WindowsPath>\RegEdit.exe** (for example, **C:\WinNT\Regedit.exe**)
- Select the path **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AeDebug**
- Set the value **0** (zero) to the parameter “**Auto**” from the path selected.
- 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 `\<StudioPath>\Local Settings\Application Data\InduSoft Web Studio v6.1\Program Settings.ini` file.

 **Note:**

You will need to verify your application to apply date settings to previously saved Web pages.

- **On-Line:** Use the function **SetDateFormat (<strSeparator>, <strDateFormat>)**. For example, **SetDateFormat ("/", "MDY")**.

- **What features are not supported by CEView?**

The main features not supported by CEView are: DDE, NetDDE, ODBC, PasteLink, and a number of functions.

- **How do I show a splashwindow when starting an application in CEView?**

To enable your CE application to show a splash window during startup, add the following key to the device's CEView.ini file:

```
[OEM]
SpashWnd = <Path to bitmap file> // default is Splash.bmp
SplashWndTime = <Time in milliseconds> // default is 1000
```

- **What are the main steps to create a Web based application?**

Follow the procedure below:

- Develop the application locally. Don't use features which are not supported by Web Thin Clients for the screens which will be saved as HTML format.
- After saving the screens in the standard format (**File > Save**), save the screens which must be available for the Web Thin Client in HTML format by the menu **File > Save as HTML**.
- Open the *Project Settings* dialog window (menu **Project > Settings**) and select the **Web** tab.
- Configure the field **Data Server IP** with the IP address of the Server station (station where Studio is running).
- Open the Project Status dialog window (menu **Project > Status**), select the **Execution Tasks** tab and set the *TCP/IP Server* module as **Startup=Automatic**.
- Open the application tags database and set the option **Server** instead of **Local** in the **Web Data** column for all tags which must exchange value between the Server and the Web Thin Client station.
- Execute the command **Tools > Verify Application** to update the **Project Settings** in the HTML files.
- If there is not any Web Server running in the computer, copy the program **NTWebServer.exe** from the `\BIN` folder of Studio into the web root (for example, `\<ApplicationFolder>\Web`) and execute it. The path where this Web Server program is executed will be the web root of the station. The Web Server is necessary to export data (web files) in HTTP protocol to the Web Thin Clients.
- Execute the application in the Server station.
- Using a browser (for example, Internet Explorer v4.1+SP1 or newer version) in the Web Thin Client station, type the URL address to download the screen

which had been saved in HTML format (for example, `http://<ServerIPAddress>/ <ScreenName> .html`).

 **Note:**

The Web Thin Client requires an *ActiveX* component (**ISSymbol.ocx**) to handle the screens on the browser. If the Web 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 Web Thin Client and register it by the command **regsvr32 ISSymbol.ocx**. This file can be found in the `\BIN` folder from the IWS installation directory.

▪ **How do I maintain communication between a Web Thin Client connecting via proxy and a Web Gateway application running on Microsoft IIS?**

Microsoft Internet Information Services (IIS) has a configuration option to keep HTTP connections alive. When this option is enabled, it may conflict with Web 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 pop-up menu. The *Default Web Site Properties* dialog is displayed.
4. Select the **Web Site** tab of the *Default Web Site Properties* dialog.
5. In the *Connections* pane of the **Web Site** tab, uncheck the **HTTP Keep-Alives Enabled** option.
6. Click **OK** to save the change and close the dialog.

▪ **How do I send an email from the IWS application?**

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 application is shut down. So, most of application execute this function just once, after starting the application;
- 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() function	SendEmailExt() function
Execution	Synchronous	Asynchronous
Supports Subject text	✓	✓
Supports Message text	✓	✓
Supports TO addresses	✓	✓
Supports CC addresses	✗	✓
Supports BCC addresses	✗	✓
Supports attached files	✗	✓

- **The runtime task (TCP/IP, OPC, DDE, ODBC, etc) does not work.**

Make sure the runtime task is set to as *Automatic* in the **Execution Tasks** tab from the *Project Status* dialog window (**Project > Status** menu). Select the runtime tasks which must be executed (for example, TCP/IP Server), hit the **Startup...** button and set it as **Automatic**.

- **The Browser from the Web Thin Client does not display the screen and launches a warning message regarding *ISSymbol.ocx*.**

Make sure the runtime task is set to **Automatic** in the **Execution Tasks** tab from the *Project Status* dialog window (**Project > Status** menu). Select the runtime tasks which must be executed (for example, TCP/IP Server), hit the **Startup...** button and set it as **Automatic**.

- **The Browse of the Web Thin Client launches an error message missing the *ISSymbol.ocx* and does not display the screens from the Server.**

Issymbol.ocx is the IWS *ActiveX* object used by the browser from the Web Thin Client to view the Web pages. If the Web Thin Client is connected to the Internet, the *ISSymbol.ocx* control is automatically downloaded and registered in the Web Thin Client station. Otherwise, it's necessary to copy it to the **\WinNT\System32** folder of the Web 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 ISSymbol132.ocx** to register the *ActiveX* component in the Web Thin Client.

- **The screens are shown on the Web Thin Client (Browser); however, the data (tags values) are not read from the Server.**

Make sure the parameter in the column **Web Data** from the application tags database is set as **Server** instead of **Local**. The tags set as **Server** keep the same value in the Server and in the Web Thin Client (Browser). The tags set as **Local** have independent values in the Server and in the Web Thin Client (Browser).



Caution:

It's necessary to execute the command **Tools > Verify application** after modifying the tags settings. Otherwise, the changes will not be updated in the web files.

- **The “On Up” expressions configured in the Command dynamic are not executed.**

The “On Up” expressions from the **Command** dynamic are not executed if the mouse pointer is dragged out the object area before releasing it. If the check-box **Release** from the **Command Object Properties** window is enabled, the **On Up** expression is executed even if the mouse pointer is dragged out the object area before releasing it.

- **The Trend History does not work after adding or removing tags in the Trend worksheet.**

When a tag is inserted or removed FROM a *Trend* worksheet, the format of the history files (***.hst**) is modified. The same **.hst** file cannot have two different formats; otherwise, the data will not be retrieved from it properly by the Trend object. If you need to add or remove tags for history files, there are two valid procedures: Create a new *Trend* worksheet or delete the old ***.hst** files.

- **The value of indirect tags (@<TagName>) is not shown in the web thin client application.**

When a screen is saved as HTML, Studio saves a <ScreenName>.tag1 file in the \WEB subfolder. This file has the list of all tags configured in the screen (objects and dynamics). When a screen is opened in the Web 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 Web 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** dynamic to this rectangle and configure the tags (which can be pointed by indirect tags during the runtime in the Web 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.

- **Which functionalities are not supported by Pocket PC platforms (for example, IPaq, Cassiopeia, Jornada)?**

WinCE devices powered Pocket PC do not support some functionality which are supported by WinCE devices powered by the “standard” Windows CE version:

Functionality not supported by Pocket PC 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 Pocket PC devices

- **How do I enable the “Hibernate” options from the operating system after installing IWS on a notebook?**

Follow the procedure below:

- Run the *Registry Editor* (<Start button>\Run\regedit).
- Select the following path from the *Registry Editor*:
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Proteq\Parameters
- 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**).
- 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**.

Appendix A. InduSoft Web Studio Functions

This chapter contains tables and information describing the different functions available with *InduSoft Web Studio* and *CEView*.

Log Message Functions	Execution	2K/XP/Vista	Win CE	Web Client
Trace	Synchronous	✓	✓	✗

Arithmetic Functions	Execution	2K/XP/Vista	Win CE	Web Client
Abs ()	Synchronous	✓	✓	✓
Div ()	Synchronous	✓	✓	✓
Format ()	Synchronous	✓	✓	✓
GetBit ()	Synchronous	✓	✓	✓
Mod ()	Synchronous	✓	✓	✓
Pow ()	Synchronous	✓	✓	✓
ResetBit ()	Synchronous	✓	✓	✓
Round ()	Synchronous	✓	✓	✓
SetBit ()	Synchronous	✓	✓	✓
SQRT ()	Synchronous	✓	✓	✓
Swap16 ()	Synchronous	✓	✓	✓
Swap32 ()	Synchronous	✓	✓	✓
Trunc ()	Synchronous	✓	✓	✓

Statistical Functions	Execution	2K/XP/Vista	Win CE	Web Client
Ave ()	Synchronous	✓	✓	✓
Max ()	Synchronous	✓	✓	✓
Min ()	Synchronous	✓	✓	✓
Rand ()	Synchronous	✓	✓	✓

Logarithmic Functions	Execution	2K/XP/Vista	Win CE	Web Client
Exp ()	Synchronous	✓	✓	✓
Log ()	Synchronous	✓	✓	✓

Log10 ()	Synchronous	✓	✓	✓
----------	-------------	---	---	---

Logical Functions	Execution	2K/XP/Vista	Win CE	Web Client
False ()	Synchronous	✓	✓	✓
If ()	Synchronous	✓	✓	✓
Toggle ()	Synchronous	✓	✓	✓
True ()	Synchronous	✓	✓	✓

String Functions	Execution	2K/XP/Vista	Win CE	Web Client
Asc2Str ()	Synchronous	✓	✓	✓
CharToValue ()	Synchronous	✓	✓	✓
CharToValueW ()	Synchronous	✓	✓	✓
ClassMembersToStrVector ()	Synchronous	✓	✓	✓
Ncopy	Synchronous	✓	✓	✓
Num ()	Synchronous	✓	✓	✓
Str ()	Synchronous	✓	✓	✓
Str2Asc ()	Synchronous	✓	✓	✓
StrCompare ()	Synchronous	✓	✓	✓
StrCompareNoCase ()	Synchronous	✓	✓	✓
StrFromInt ()	Synchronous	✓	✓	✓
StrFromReal ()	Synchronous	✓	✓	✓
StrFromTime ()	Synchronous	✓	✓	✓
StrGetElement ()	Synchronous	✓	✓	✓
StrLeft ()	Synchronous	✓	✓	✓
StrLen ()	Synchronous	✓	✓	✓
StrLower ()	Synchronous	✓	✓	✓
StrRChr ()	Synchronous	✓	✓	✓
StrRight ()	Synchronous	✓	✓	✓
StrSetElement ()	Synchronous	✓	✓	✓
StrStr ()	Synchronous	✓	✓	✓

String Functions	Execution	2K/XP/Vista	Win CE	Web Client
StrStrPos()	Synchronous	✓	✓	✓
StrTrim()	Synchronous	✓	✓	✓
StrTrimAll()	Synchronous	✓	✓	✓
StrUpper()	Synchronous	✓	✓	✓
ValueToChar()	Synchronous	✓	✓	✓
ValueWToChar()	Synchronous	✓	✓	✓

Date & Time Functions	Execution	2K/XP/Vista	Win CE	Web Client
ClockGetDate()	Synchronous	✓	✓	✓
ClockGetDayOfWeek()	Synchronous	✓	✓	✓
ClockGetTime()	Synchronous	✓	✓	✓
DateTime2Clock()	Synchronous	✓	✓	✓
GetClock()	Synchronous	✓	✓	✓
Hour2Clock()	Synchronous	✓	✓	✓
SetSystemDate()	Synchronous	✓	✓	✓
SetSystemTime()	Synchronous	✓	✓	✓

Trigonometric Functions	Execution	2K/XP/Vista	Win CE	Web Client
Acos()	Synchronous	✓	✓	✓
Asin()	Synchronous	✓	✓	✓
Atan()	Synchronous	✓	✓	✓
Cos()	Synchronous	✓	✓	✓
Cot()	Synchronous	✓	✓	✓
Pi()	Synchronous	✓	✓	✓
Sin()	Synchronous	✓	✓	✓
Tan()	Synchronous	✓	✓	✓

Opening and Closing Windows Functions	Execution	2K/XP/Vista	Win CE	Web Client
Close()	Asynchronous	✓	✗	✗

Open ()	Asynchronous	✓	✓	✓
OpenPrevious ()	Asynchronous	✓	✓	✓

Security Functions	Execution	2K/XP/Vista	Win CE	Web Client
BlockUser ()	Synchronous	✓	✓	✓
CheckESign ()	Synchronous	✓	✓	✓
CreateUser ()	Synchronous	✓	✓	✓
GetUserNames ()	Synchronous	✓	✓	Exec. on Server
GetUserPwdAging ()	Synchronous	✓	✓	✓
GetUserState ()	Synchronous	✓	✓	✓
RemoveUser ()	Synchronous	✓	✓	✓
SetPassword ()	Synchronous	✓	✓	✓
UnblockUser ()	Synchronous	✓	✓	✓

Module Activity Functions	Execution	2K/XP/Vista	Win CE	Web Client
AppActivate ()	Asynchronous	✓	✓	✓
AppIsRunning ()	Synchronous	✓	✓	✓
AppPostMessage ()	Synchronous	✓	✓	✓
AppSendKeys ()	Synchronous	✓	✗	✓
CleanReadQueue ()	Synchronous	✓	✗	Exec. on Server
CloseSplashWindow ()	Synchronous	✓	✗	Exec. on Server
EndTask ()	Asynchronous	✓	✗	Exec. on Server
ExitWindows ()	Asynchronous	✓	✗	✓
IsScreenOpen ()	Synchronous	✓	✓	✗
IsTaskRunning ()	Synchronous	✓	✗	Exec. on Server
IsViewerInFocus ()	Synchronous	✓	✗	✗
KeyPad ()	Asynchronous	✓	✓	✓
LogOff ()	Asynchronous	✓	✓	✓
LogOn ()	Asynchronous	✓	✓	✓
Math ()	Synchronous	✓	✓	Exec. on Server

Module Activity Functions	Execution	2K/XP/Vista	Win CE	Web Client
PostKey ()	Synchronous	✓	✓	✓
Recipe ()	Synchronous	✓	✓	Exec. on Server
Report ()	Synchronous	✓	✓	Exec. on Server
RunGlobalProcedureOnServer ()	Synchronous	✓	✓	✓
RunVBScript ()	Synchronous	✓	✓	✓
SendKeyObject ()	Synchronous	✓	✗	✗
SetAppPath ()	Synchronous	✓	✗	Exec. on Server
SetKeyboardLanguage ()	Synchronous	✓	✓	✓
SetViewerInFocus ()	Synchronous	✓	✗	✓
SetViewerPos ()	Synchronous	✓	✗	✓
ShutDown ()	Synchronous	✓	✓	✗
StartTask ()	Asynchronous	✓	✗	Exec. on Server
ViewerPostMessage ()	Asynchronous	✓	✓	✓
Wait ()	Synchronous	✓	✓	✓
WinExec ()	Asynchronous	✓	✓	✓
WinExecIsRunning ()	Synchronous	✓	✓	✓

File Functions	Execution	2K/XP/Vista	Win CE	Web Client
DeleteOlderFiles ()	Synchronous	✓	✓	✓
DirCreate ()	Synchronous	✓	✓	✓
DirDelete ()	Synchronous	✓	✓	✓
DirLength ()	Synchronous	✓	✓	✓
DirRename ()	Synchronous	✓	✓	✓
FileCopy ()	Synchronous	✓	✓	✓
FileDelete ()	Synchronous	✓	✓	✓
FileLength ()	Synchronous	✓	✓	✓
FileRename ()	Synchronous	✓	✓	✓
FileWrite ()	Synchronous	✓	✓	✓
FindFile ()	Synchronous	✓	✓	✓

File Functions	Execution	2K/XP/Vista	Win CE	Web Client
FindPath ()	Synchronous	✓	✓	✓
GetFileAttributes ()	Synchronous	✓	✓	✓
GetFileTime ()	Synchronous	✓	✓	✓
GetHstInfo ()	Synchronous	✓	✓	✓
GetLine ()	Synchronous	✓	✓	✓
Hst2Txt ()	Asynchronous	✓	✓	✓
Hst2TxtIsRunning ()	Synchronous	✓	✗	Exec. on Server
PDFCreate ()	Synchronous	✓	✗	✓
Print ()	Synchronous	✓	✓	✓
RDFileN ()	Synchronous	✓	✓	✓

Graphic Functions	Execution	2K/XP/Vista	Win CE	Web Client
AutoFormat ()	Synchronous	✓	✓	✓
GetScrInfo ()	Synchronous	✓	✓	✓
PrintSetup ()	Asynchronous	✓	✓	✓
PrintWindow ()	Asynchronous	✓	✓	✗
ResetDecimalPointsTable ()	Synchronous	✓	✓	✓
RGBColor ()	Synchronous	✓	✓	✓
RGBComponent ()	Synchronous	✓	✓	✓
SetDecimalPoints ()	Synchronous	✓	✓	✓
SetDisplayUnit ()	Synchronous	✓	✓	✓
SetTagDisplayUnit ()	Synchronous	✓	✓	✓

Translation Functions	Execution	2K/XP/Vista	Win CE	Web Client
Ext ()	Synchronous	✓	✓	✓
SetTranslationFile ()	Synchronous	✓	✓	✓

Multimedia Functions	Execution	2K/XP/Vista	Win CE	Web Client
Play ()	Synchronous/Asynchronous	✓	✓	✓

System Info. Functions	Execution	2K/XP/Vista	Win CE	Web Client
DbVersion ()	Synchronous	✓	✓	✓
GetAppHorizontalResolution ()	Synchronous	✓	✗	Exec. on Server
GetAppPath ()	Synchronous	✓	✓	Exec. on Server
GetAppVerticalResolution ()	Synchronous	✓	✗	Exec. on Server
GetComputerIP ()	Synchronous	✓	✓	✓
GetComputerName ()	Synchronous	✓	✗	✓
GetCursorX ()	Synchronous	✓	✓	✓
GetCursorY ()	Synchronous	✓	✓	✓
GetDisplayHorizontalResolution ()	Synchronous	✓	✓	✓
GetDisplayVerticalResolution ()	Synchronous	✓	✓	✓
GetFreeMemoryCE ()	Synchronous	✗	✓	✓
GetHardKeyModel ()	Synchronous	✓	✗	Exec. on Server
GetHardKeySN ()	Synchronous	✓	✗	Exec. on Server
GetIPAll ()	Synchronous	✓	✓	✓
GetMemoryCE ()	Synchronous	✗	✓	✓
GetNetMACID ()	Synchronous	✓	✓	✓
GetOS ()	Synchronous	✓	✓	✓
GetPrivateProfileString ()	Synchronous	✓	✓	✓
GetProductPath ()	Synchronous	✓	✓	✓
GetRegValue ()	Synchronous	✓	✓	✗
GetRegValueType ()	Synchronous	✓	✓	✗
GetServerHostName	Synchronous	✗	✗	✓
GetTickCount ()	Synchronous	✓	✓	✓
InfoAppAlrDir ()	Synchronous	✓	✓	✓
InfoAppHSTDir ()	Synchronous	✓	✓	✓
InfoDiskFree ()	Synchronous	✓	✗	✓
InfoResources ()	Synchronous	✓	✓	✓
IsActiveXReg ()	Synchronous	✓	✓	✓

System Info. Functions	Execution	2K/XP/Vista	Win CE	Web Client
NoInputTime ()	Synchronous	✓	✓	✓
ProductVersion ()	Synchronous	✓	✓	✓
RegSaveCE ()	Synchronous	✗	✓	✗
SaveAlarmFile ()	Synchronous	✓	✓	✗
SetAppAlarmPath ()	Synchronous	✓	✓	Exec. on Server
SetAppHSTPath ()	Synchronous	✓	✓	Exec. on Server
SetDataFormat ()	Synchronous	✓	✓	✓
SetRegValue ()	Synchronous	✓	✓	✗
SetWebConfig ()	Synchronous	✓	✓	Exec. on Server
SNMPGet ()	Synchronous	✓	✗	✓
WritePrivateProfileSharing ()	Synchronous	✓	✓	✓

Tags Database Functions	Execution	2K/XP/Vista	Win CE	Web Client
ExecuteAlarmAck ()	NA	✓	✓	✓
ForceTagChange ()	Synchronous	✓	✓	✓
GetTagValue ()	Synchronous	✓	✓	✓
SetTagValue ()	Synchronous	✓	✓	✓

Loop Function	Execution	2K/XP/Vista	Win CE	Web Client
For () & Next	NA	✓	✓	✗

ODBC Functions	Execution	2K/XP/Vista	Win CE	Web Client
ODBCBeginTrans ()	Synchronous	✓	✗	Exec. on Server
ODBCBindCol ()	Synchronous	✓	✗	Exec. on Server
ODBCCanAppend ()	Synchronous	✓	✗	Exec. on Server
ODBCCanTransact ()	Synchronous	✓	✗	Exec. on Server
ODBCCanUpdate ()	Synchronous	✓	✗	Exec. on Server
ODBCClose ()	Synchronous	✓	✗	Exec. on Server
ODBCCommitTrans ()	Synchronous	✓	✗	Exec. on Server

ODBC Functions	Execution	2K/XP/Vista	Win CE	Web Client
ODBCDelete ()	Synchronous	✓	✗	Exec. on Server
ODBCExecuteSQL ()	Synchronous	✓	✗	Exec. on Server
ODBCInsert ()	Synchronous	✓	✗	Exec. on Server
ODBCIsBOF ()	Synchronous	✓	✗	Exec. on Server
ODBCIsDeleted ()	Synchronous	✓	✗	Exec. on Server
ODBCIsEOF ()	Synchronous	✓	✗	Exec. on Server
ODBCIsFieldNull ()	Synchronous	✓	✗	Exec. on Server
ODBCIsFieldNullable ()	Synchronous	✓	✗	Exec. on Server
ODBCMove ()	Synchronous	✓	✗	Exec. on Server
ODBCMoveFirst ()	Synchronous	✓	✗	Exec. on Server
ODBCMoveLast ()	Synchronous	✓	✗	Exec. on Server
ODBCMoveNext ()	Synchronous	✓	✗	Exec. on Server
ODBCMovePrev ()	Synchronous	✓	✗	Exec. on Server
ODBCOpen ()	Synchronous	✓	✗	Exec. on Server
ODBCQuery ()	Synchronous	✓	✗	Exec. on Server
ODBCRollback ()	Synchronous	✓	✗	Exec. on Server
ODBCSetFieldNull ()	Synchronous	✓	✗	Exec. on Server
ODBCSetFilter ()	Synchronous	✓	✗	Exec. on Server
ODBCSetSort ()	Synchronous	✓	✗	Exec. on Server
ODBCUnbindCol ()	Synchronous	✓	✗	Exec. on Server
ODBCUpdate ()	Synchronous	✓	✗	Exec. on Server

Email Functions	Execution	2K/XP/Vista	Win CE	Web Client
CNFEmail ()	Synchronous	✓	✓	✓
GetStatussendEmailExt ()	Synchronous	✓	✓	✓
SendEmail ()	Synchronous	✓	✓	✓
SendEmailExt ()	Asynchronous	✓	✓	✓

Dial-Up Functions	Execution	2K/XP/Vista	Win CE	Web Client
-------------------	-----------	-------------	--------	------------

Dial-Up Functions	Execution	2K/XP/Vista	Win CE	Web Client
DialError ()	Synchronous	✓	✓	✓
DialGetClientIP ()	Synchronous	✓	✓	✓
DialGetServerIP ()	Synchronous	✓	✗	✓
DialStatus ()	Synchronous	✓	✓	✓
DialUp ()	Asynchronous	✓	✓	✓
DialUpToCE ()	Asynchronous	✓	✗	✓
FindAllDevices ()	Synchronous	✓	✓	✓
FindModem ()	Synchronous	✓	✓	✓
HangUp ()	Synchronous	✓	✓	✓
PhoneDialUp ()	Asynchronous	✓	✗	✓
PhoneDisableListen ()	Synchronous	✓	✗	✓
PhoneEnableListen ()	Synchronous	✓	✗	✓
PhoneHangUp ()	Asynchronous	✓	✗	✓
PhoneStatus ()	Synchronous	✓	✗	✓

ActiveX Functions	Execution	2K/XP/Vista	Win CE	Web Client
XGet ()	Asynchronous	✓	✓	✓
XRun ()	Asynchronous	✓	✓	✓
XSet ()	Asynchronous	✓	✓	✓

Event Logger Functions	Execution	2K/XP/Vista	Win CE	Web Client
SendEvent ()	Synchronous	✓	✓	✓

FTP Functions	Execution	2K/XP/Vista	Win CE	Web Client
CNFFtp ()	Synchronous	✓	✓	✓
ftpGet ()	Asynchronous	✓	✓	✓
ftpPut ()	Asynchronous	✓	✓	✓
ftpStatus ()	Synchronous	✓	✓	✓

DB/ERP Functions	Execution	2K/XP/Vista	Win CE	Web Client
DBCursorClose ()	Synchronous	✓	✓	✓
DBCursorCurrentRow ()	Synchronous	✓	✓	✓
DBCursorGetValue ()	Synchronous	✓	✓	✓
DBCursorMoveTo ()	Synchronous	✓	✓	✓
DBCursorNext ()	Synchronous	✓	✓	✓
DBCursorOpen ()	Synchronous	✓	✓	✓
DBCursorOpenSQL ()	Synchronous	✓	✓	✓
DBCursorPrevious ()	Synchronous	✓	✓	✓
DBCursorRowCount ()	Synchronous	✓	✓	✓
DBDelete ()	Synchronous	✓	✓	✓
DBExecute ()	Synchronous	✓	✓	✓
DBInsert ()	Synchronous	✓	✓	✓
DBSelect ()	Synchronous	✓	✓	✓
DBUpdate ()	Synchronous	✓	✓	✓
SyncAlarm ()	Asynchronous	✓	✓	Exec. on Server
SyncAlarmStatus ()	Synchronous	✓	✓	Exec. on Server
SyncEvent ()	Asynchronous	✓	✓	Exec. on Server
SyncEventStatus ()	Synchronous	✓	✓	Exec. on Server
SyncTrend ()	Asynchronous	✓	✓	Exec. on Server
SyncTrendStatus ()	Synchronous	✓	✓	Exec. on Server

Function Prototypes and Descriptions

The function tag names used in IWS must conform to the following syntax:

- **num[Name]**: Numerical tag or value
- **str[Name]**: String tag or value
- **tag[Name]**: Tag Name
- **optNum[Name]**: Optional Numerical tag or value
- **optStr[Name]**: Optional String tag or value
- **optTag[Name]**: Optional Tag Name

This syntax identifies the argument types required for each parameter of the IWS function.

 **Note:**

These prototypes depict dynamic tags used to pass values to functions; however, you can also enter static values into these functions. To replace string tags with static character strings, enter the string between double-quotation marks (for example, enter "ABCDEFG" instead of **strTag**). You also can replace static numbers with numeric tags (for example, enter 45.6543 instead of **numTag**).

Log Message Functions

This section describes the InduSoft Web Studio Log Message function, **Trace (strOutputMessage)**.

Trace (strOutputMessage)

Group	Log Message
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	NOT supported

- **Description:** Displays the contents of **strOutputMessage** in the *LogWin* window.
- **Parameter:** This function accepts the following parameter:

StrOuputMessage	String tag containing a message to display in the LogWin
------------------------	--

- **Examples:**

Tag Name	Expression
	Trace("Starting Step 5") // Starting Step 5 displays in the LogWin window.
	Trace(Date) // The contents of the tag Date displays in the LogWin window.

 **Note:**

You will find this function useful for debugging purposes. For example, if you want to know when IWS is executing a specific math script.

 **Tip:**

You can concatenate text, expressions, and tag values to compose the **strOutputMessage** parameter. For example,

```
Trace("The tag second has the value"+second+" and the Internal Clock = "+GetTickCount( ))
```

Arithmetic Functions

This section describes the following InduSoft Web Studio Arithmetic functions:

- `ABS(numValue)`
- `Div(numNumerator, numDenominator)`
- `Format(strFlag, numValue)`
- `GetBit(tagName, numBitNumber)`
- `Mod(numNumerator, numDenominator)`
- `Pow(numBase, numExponent)`
- `ResetBit("tagName", numBitNumber)`
- `Round(numValue)`
- `SetBit("tagName", numBitNumber)`
- `SQRT(numValue)`
- `Swap32(numValue)`
- `Swap16(numValue)`
- `Trunc(numValue)`

ABS(numValue)

Group	Arithmetic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Performs the Absolute value function on the contents of the `numValue` tag.

- **Parameters:**

numValue	Integer or Real tag containing the number from which the function takes the absolute value.
-----------------	---

- **Returned Values:** Numerical result of the Absolute value function.

- **Examples:**

Tag Name	Expression
Tag	<code>ABS("-54.9788")</code> // Returned value = 54.9788
Tag	<code>ABS(numValue)</code> // Returned value = absolute value of the number in the numValue tag.

Div(numNumerator, numDenominator)

Group	Arithmetic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Divides the value in `numNumerator` by the value of `numDenominator` and returns only the whole integer number, omitting the remainder.

- **Parameters:**

numNumerator	Integer or Real tag containing the Numerator of the function.
numDenominator	Integer or Real tag containing the Denominator of the function.

- **Returned Values:** Numerical result of the function as an integer.

- **Examples:**

Tag Name	Expression
numValue	Div(100, 8) // Returns the value 12.5
numValue	Div(16, 4) // Returns the value 4
numValue	Div(100,12.5) //Returns the value 8

⇒ **Tip:**

Use the **MOD ()** function to get the remainder of the division.

Format (strFlag, numValue)

Group	Arithmetic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Modifies the format of a numerical value, according to the flag.
- **Parameters:**

strFlag	<p>Sets the format in accordance with the <code>%m.nF</code> syntax.</p> <p>Where:</p> <ul style="list-style-type: none"> • m (Applicable for flags d, x, X, o, b, f, e, E, g, G, s, c, and h): Sets the minimum number of characters returned by the function, adding blank space chars on the left of the returned value or 0 (zero) chars (see examples). • n (Applicable for flags f, e, E, g, and G): Sets the minimum number of decimal characters for the floating values returned by the function. • F: Determines how the value is formatted • d: Decimal • x: Hexadecimal (chars in lower case) • X: Hexadecimal (chars in uppercase) • o: Octal • b: Binary • f: Float • e: Scientific notation (e in lowercase) • E: Scientific notation (E in uppercase) • g: Round the value (e in lowercase, when applicable) • G: Round the value (E in lowercase, when applicable) • s: String • c: ASCII char • h: Hour (hh:mm:ss) <p>Alternatively, the format can be set using the <code>##.###</code> syntax, where the numerical value is rounded to the given number of decimal places.</p>
numValue	Numerical value to be formatted.

- **Returned Values:** String value that is formatted according to the parameters configured in the function.
- **Examples:**

Tag Name	Expression
Tag	Format("%d",12.34) // returned value = "12"
Tag	Format("%04d",12.34) // returned value = "0012"


Tag Name	Expression
Tag	Format("%4d",12.34) // returned value = "12"
Tag	Format("%x",26) // returned value = "1a"
Tag	Format("%04x",26) // returned value = "001a"
Tag	Format("%4x",26) // returned value = "1a"
Tag	Format("%X",26) // returned value = "1A"
Tag	Format("%04X",26) // returned value = "001A"
Tag	Format("%4X",26) // returned value = "1A"


Tag Name	Expression
Tag	Format("%o",16) // returned value = "20"
Tag	Format("%04o",16) // returned value = "0020"
Tag	Format("%4o",16) // returned value = "20"
Tag	Format("%b",2) // returned value = "10"
Tag	Format("%4b",2) // returned value = "0010"
Tag	Format("%04b",2) // returned value = "0010"
Tag	Format("%0.1f",12.34) // returned value = "12.3"
Tag	Format("%06.1f",12.34) // returned value = "0012.3"
Tag	Format("%6.1f",12.34) // returned value = "12.3"
Tag	Format("%e",12.34) // returned value = "1.234000e+001"
Tag	Format("%0.1e",12.34) // returned value = "1.2e+001"
Tag	Format("%09.1e",12.34) // returned value = "01.2e+001"
Tag	Format("%9.1e",12.34) // returned value = "1.2e+001"
Tag	Format("%E",12.34) // returned value = "1.234000E+001"
Tag	Format("%0.1E",12.34) // returned value = "1.2E+001"

Tag	Format("%09.1E",12.34) // returned value = "01.2E+001"
Tag	Format("%9.1E",12.34) // returned value = " 1.2E+001"
Tag	Format("%0.1g",12.34) // returned value = "1e+001"
Tag	Format("%0.2g",12.34) // returned value = "12"
Tag	Format("%0.3g",12.34) // returned value = "12.3"
Tag	Format("%05.3g",12.34) // returned value = "012.3"
Tag	Format("%5.3g",12.34) // returned value = " 12.3"

Tag Name	Expression
Tag	Format("%0.1G",12.34) // returned value = "1E+001"
Tag	Format("%0.2G",12.34) // returned value = "12"
Tag	Format("%0.3G",12.34) // returned value = "12.3"
Tag	Format("%05.3G",12.34) // returned value = "012.3"
Tag	Format("%5.3G",12.34) // returned value = " 12.3"
Tag	Format("%s",12.34) // returned value = "12"
Tag	Format("%04s",12.34) // returned value = "0012"
Tag	Format("%4s",12.34) // returned value = "12"
Tag	Format("%c",97) // returned value = "a"
Tag	Format("%4c",97) // returned value = "a"
Tag	Format("%04c",97) // returned value = "000a"
Tag	Format("%h",30) // returned value = "00:00:30"
Tag	Format("%h",60) // returned value = "00:01:00"
Tag	Format("%h",90) // returned value = "00:01:30"
Tag	Format("%h",3600) // returned value = "01:00:00"

Tag Name	Expression
Tag	Format("##.#", 26.56789) // Returned value = "26.6"
Tag	Format("#.##", 26.56789) // Returned value = "26.57"
Tag	Format("##.##", 26.56789) // Returned value = "26.57"

 **Note:**
The **Format (strFlag, numValue)** function allows the same flags to follow the % symbol that are used for the standard C function **printf ()**; however, you can format only one value in each cell.

 **Tip:**
You will find the **Format ()** function especially useful when you are formatting the number of characters for values to be printed in reports. Also, you must use this function to convert the number of seconds in the following format:
hh:mm:ss (strFlag =%h)
The **h** and **m** parameters are optional.

GetBit (tagName, numBitNumber)

Group	Arithmetic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Isolates a single bit in a numerical tag
- **Parameters:**

tagName	The name of an Integer tag where the bit value will be taken from.
numBitNumber	A numerical tag, which holds the number of the bit to be isolated. (0...31)

- **Returned Values:** Returns the numerical value (0 or 1) that corresponds to the value of the isolated bit.
- **Examples:**

Tag Name	Expression
Tag	GetBit(numSource, 4) // If the tag numSource held the value 15, this function would return the value 0.
Tag	GetBit(numSource,1) // If the tag numSource held the value 19, this function would return the value 1.

⇒ **Tip:**

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).

Mod(numNumerator, numDenominator)


Group	Arithmetic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Divides the value of `numNumerator` by the value of `numDenominator` only and returns the remainder as a real number.
- **Parameters:**

numNumerator	Integer or Real tag containing the Numerator of the function.
numDenominator	Integer or Real tag containing the Denominator of the function.

- **Returned Values:** Returns the remainder after dividing `numNumerator` by `numDenominator`.
- **Examples:**

Tag Name	Expression
Tag	Mod(50, 4) // Returned value = 2
Tag	Mod(16,4) // Returned value = 0
Tag	Mod(100, 8.2) //Returned value = 1.600

 **Note:**
Use the **DIV()** function to get the integer result of the division.

Pow(numBase, numExponent)

Group	Arithmetic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the result of raising `numBase` to the power of `numExponent`.
- **Parameters:**

numBase	Integer or Real tag containing the Base of the function.
numExponent	Integer or real tag containing the Exponent of the function.

- **Returned Values:** Returns the result of raising the base to the exponent.
- **Examples:**

Tag Name	Expression
Tag	Pow(2, 3) // Returned value = 8
Tag	Pow(10,4) // Returned value = 10000

ResetBit("tagName", numBitNumber)

Group	Arithmetic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Sets a single bit in a numerical tag to 0.
- **Parameters:**

"tagName"	Name of an Integer tag where the bit value will be reset.
numBitNumber	Numerical tag holding the number of the bit to be reset. (0...31)

- **Returned Values:**

0	No error
1	Invalid parameter
2	Tag does not exist

- **Examples:**

Tag Name	Expression
Tag	ResetBit("numSource", 4) // If the tag numSource held the value 16, this function would return the value 0 and numSource would hold the value 0.
Tag	ResetBit("numSource",1) // If the tag numSource held the value 19, this function would return the value 0 and numSource would hold the value 17.

 **Note:**

To enter the name of the integer tag directly (instead of using the **tagName** parameter) you must enter the name between double-quotes. For example, **SetBit ("Second", 1)**.

 **Tip:**

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).

Round(numValue)

Group	Arithmetic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Rounds **numValue** to the nearest integer.

- **Parameters:**

numValue	A Real tag that holds the value to be rounded.
-----------------	--

- **Returned Values:** Returns the integer result of the round function.

- **Examples:**

Tag Name	Expression
Tag	Round("345.87") // Returned value = 346
Tag	Round("65.323") // Returned value = 65

SetBit("tagName", numBitNumber)

Group	Arithmetic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Sets a single bit in a numerical tag to 1.
- **Parameters:**

"tagName"	Name of an Integer tag where the bit value will be set.
numBitNumber	Numerical tag holding the number of the bit to be set. (0...31)

- **Returned Values:**

0	No error
1	Invalid parameter
2	Tag does not exist

- **Examples:**

Tag Name	Expression
Tag	GetBit("numSource", 4) // If the tag <code>numSource</code> held the value 0, this function would return the value 0 and <code>numSource</code> would hold the value 16.
Tag	GetBit("numSource",1) // If the tag <code>numSource</code> held the value 17, this function would return the value 0 and <code>numSource</code> would hold the value 19.

 **Note:**

To enter the name of the integer tag directly (instead of using the `tagName` parameter) you must enter the name between double-quotes. For example, **SetBit ("Second", 1)**.

 **Tip:**

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).

SQRT (numValue)

Group	Arithmetic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Takes the square root of **numValue**.
- **Parameters:**

NumValue	Integer or Real tag to be square rooted.
-----------------	--

- **Returned Values:** Returns the square root of the value in the **numValue** tag.
- **Examples:**

Tag Name	Expression
Tag	SQRT(25) // Returns the value 5
Tag	SQRT(67) // Returns the value 8.185353

**Note:**

If **numValue** has a negative value, then this function returns the value **0** and sets the quality of the returned tag to **BAD**.

Swap16(numValue)

Group	Arithmetic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Swaps the two lower bytes of a tag.
- **Parameters:**

numValue	Integer tag that holds the numeric value of the bytes to be swapped.
-----------------	--

- **Returned Values:** Returns the numeric value after swapping the bytes.
- **Examples:**

Tag Name	Expression
Tag	Swap16(16) // 16 = 0000000000010000 in binary. Returned value = 4096 = 0001000000000000 in binary.
Tag	Swap16(43760) // 43760 = 1010010111110000 in binary. Returned value = 61610 = 1111000010100101 in binary.

Trunc (numValue)

Group	Arithmetic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Truncates the value of **numValue**.
- **Parameters:**

numValue	Real tag to be truncated.
-----------------	---------------------------

- **Returned Values:** Returns the integer portion of the real number value of **numValue**.
- **Examples:**

Tag Name	Expression
Tag	Trunc(234.987) // Returned value = 234
Tag	Trunc(-3465.9) // Returned value = -3465.9

Statistical Functions

This section describes the following InduSoft Web Studio Statistical functions:

- `Avg(numValue1, numValue2, ..., numValueN)`
- `Avg("tagArray", numSample, optnumIgnore)`
- `Max(numValue1, numValue2, ..., numValueN)`
- `Max("tagArray", numSample, optnumIgnore)`
- `Min(numValue1, numValue2, ..., numValueN)`
- `Min("tagArray", numSample, optnumIgnore)`
- `Rand()`

Avg(numValue1, numValue2, ..., numValueN)
Avg("tagArray", numSample, optnumIgnore)

Group	Statistical
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Calculates the average value of a set of numbers.
- **Parameters:**

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	Integer tag containing 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 Values:** Returns the average of the values.
- **Examples:**

Tag Name	Expression
Tag	Avg(1,2.34,5,7,4,8,9.4) // Returned value = 5.248571
Tag	Avg(1, 5, -9, 0, 5, 3) // Returned value = 0.833333
Tag	Avg("tagArray[1]",3) // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 30
Tag	Avg("tagArray[1]",3,10) // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 40

 **Note:**

This function has two formats:

- If the first parameter is a numerical 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.

Max(numValue1, numValue2, ..., numValueN)
Max("tagArray", numSample, optnumIgnore)

Group	Statistical
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the maximum value of a set of numbers.
- **Parameters:**

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	Integer tag containing the number of array elements to be analyzed.
optnumIgnore	Integer or Real tags containing the value to be ignored in the analysis.

- **Returned Values:** Returns the maximum value of the set.
- **Examples:**

Tag Name	Expression
Tag	Max(1,2.34,5,7,4,8,9.4) // Returned value = 9.4
Tag	Max(1, 5, -9, 0, 5, 3) // Returned value = 5
Tag	Max("tagArray[1]",3) // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 60
Tag	Max("tagArray[1]",3,10) // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 60

 **Note:**

This function has two formats:

- If the first parameter is a numerical 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.

Min(numValue1, numValue2, ..., numValueN)
Min("tagArray", numSample, optnumIgnore)

Group	Statistical
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the minimum value of a set of numbers.
- **Parameters:**

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	Integer tag containing the number of array elements to be analyzed.
optnumIgnore	Integer or Real tags containing a value to be ignored in the analysis.

- **Returned Values:** Returns the minimum value of the set.
- **Examples:**

Tag Name	Expression
Tag	Min(1,2.34,5,7,4,8,9.4) // Returned value = 1
Tag	Min(1, 5, -9, 0, 5, 3) // Returned value = -9
Tag	Min("tagArray[1]",3) // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 10
Tag	Min("tagArray[1]",3,10) // If tagArray[1]=10, tagArray[2]=20 and tagArray[3]=60, then the Returned Value = 20

 **Note:**

This function has two formats:

If the first parameter is a numerical 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.

Rand ()

Group	Statistical
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Generates a random number between 0 and 1.
- **Returned Values:** Returns a real number between 0 and 1.
- **Examples:**

Tag Name	Expression
Tag	Rand() // Returned value = ?, Where: $0 < ? < 1$

Logarithmic Functions

This section describes the following InduSoft Web Studio Logarithmic functions:

- `Exp(numValue)`
- `Log(numValue)`
- `Log10(numValue)`

Exp(numValue)

Group	Logarithmic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Calculates the value of **e** ($e = 2.718282$) raised to the power of `numValue`
- **Parameters:**

NumValue	Integer or Real tag containing the exponent of e .
-----------------	---

- **Returned Values:** Returns the value of $e^{(numValue)}$.
- **Examples:**

Tag Name	Expression
Tag	<code>Exp(1)</code> // Returned value = 2.718282
Tag	<code>Exp(5.25896)</code> // Returned value = 192.281415

Log(numValue)

Group	Logarithmic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Calculates the natural log of `numValue`
- **Parameters:**

NumValue	Integer or Real tag from which the natural log is taken.
-----------------	--

- **Returned Values:** Returns the value of `ln (numValue)`.
- **Examples:**

Tag Name	Expression
Tag	Log(2.718282)// Returned value = 1
Tag	Log(100) // Returned value = 4.605170

 **Note:**

If `numValue` has a negative value, this function returns the value 0 and sets the quality of the returned tag to **BAD**.

Log10 (numValue)


Group	Logarithmic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Calculates the log base 10 of **numValue**.
- **Parameters:**

NumValue	Integer or Real tag, from which the log base 10 is take.
-----------------	--

- **Returned Values:** Returns the value of **log10 (numValue)**.
- **Examples:**

Tag Name	Expression
Tag	Log10(1000)// Returned value = 3
Tag	Log10(43.05) // Returned value = 1.633973

 **Note:**
If the **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**.

Logical Functions

This section describes the following InduSoft Web Studio Logical functions:

- `False(numExpression)`
- `If(numExpression, numThen, optnumElse)`
- `Toggle(numValue)`
- `True(numExpression)`

False (numExpression)

Group	Logical
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Determines whether the content of **numExpression** is logically false.
- **Parameters:**

numExpression	Tag or expression to be used in the function.
----------------------	---

- **Returned Values:**

0	If the tag or expression is <i>not</i> logically false.
1	If the tag or expression is logically false.

- **Examples:**

Tag Name	Expression
Tag	<code>False(1)</code> // Returned value = 0
Tag	<code>False(5 < 2)</code> // Returned value = 1

⇒ Tip:

You will find this function especially useful if you need to return the value 0 when the expression returns any value other than 0.

If (numExpression, numThen, optNumElse)

Group	Logical
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Determines whether the contents of `numExpression` are logically true, and then returns the value of `numThen` or `optnumElse` accordingly.

- **Parameters:**

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 Values:**

numThen	If the numExpression is logically true.
optnumElse	If the numExpression is logically false.
No value returned	If the numExpression is logically false and there is no <code>optnumElse</code> in the function.

- **Examples:**

Tag Name	Expression
Tag	If(5>4,10, 6)// Returned value = 10
Tag	If(5<2, 0, 2) // Returned value = 0
Tag	If(3=9, 67) // No returned value. (Tag retains previous value.)

⇒ **Tips:**

- The `numThen` argument 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)`.
- The `numExpression` parameter can be a combination of logic statements (**AND**, **OR**, and **NOT**). For example, `If (TagA>TagB AND TagA=10, 1, 0)`.

Toggle(numValue)

Group	Logical
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the toggled value from the contents of **numValue** tag.
- **Parameters:**

numValue	Boolean tag containing the value to be toggled.
-----------------	---

- **Returned Values:** Numerical result (0 or 1) of the value to be toggled.
- **Examples:**

Tag Name	Expression
Tag	Toggle(myBoolTag) // Returned value = 1 if myBoolTag equals 0, or 0 if myBoolTag equals 1
Tag	Toggle(numValue) // Returned value = toggled value of the number in the numValue tag

⇒ **Tip:**

This function does not actually change the value of the tag, but it can be used in a command or operation that does.

True (numExpression)

Group	Logical
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Determines whether the contents of **numExpression** are logically true.
- **Parameters:**

numExpression	Tag or expression to be used in the function.
----------------------	---

- **Returned Values:**

0	If the tag or expression is not logically true.
1	If the tag or expression is logically true.

- **Examples:**

Tag Name	Expression
Tag	True(1) // Returned value = 1
Tag	True(5 < 2) // Returned value = 0

⇒ **Tip:**

You may find this function especially useful if you need to return the value 1 when the expression returns a value other than 0.

String Functions

This section describes the following InduSoft Web Studio String functions:

- Asc2Str(numChar1, numChar2, ..., numCharN)
- CharToValue("tagName", "tagArray")
- CharToValueW("tagName", "tagArray")
- ClassMembersToStrVector("strClassTag", numStartPos, numNumPos, "strArrayTag", optBooStartPosTarget)
- NCopy(strSource, numStartChar, numQtdChar)
- Num(strValue)
- Str(numValue)
- Str2Asc(strChar)
- StrCompare(strValue1, strValue2)
- StrCompareNoCase(strValue1, strValue2)
- StrFromInt(numValue, numBase)
- StrFromReal(numValue, numPrecision, "strType")
- StrFromTime(numUTCTime, numType)
- StrGetElement(strSource, strDelimiter, numElementNumber)
- StrLeft(strSource, numQtdChar)
- StrLen(strSource)
- StrLower(strSource)
- StrRChr(strSource, strCharSequence)
- StrRight(strSource, numQtdChar)
- StrSetElement(strSource, strDelimiter, numElementNumber, strValue)
- StrStr(strSource, strCharSequence)
- StrStrPos(strSource, strCharSequence)
- StrTrim(strReference, optNumFlag)
- StrTrimAll(strReference, optStrTrimChar)
- StrUpper(strSource)
- ValueToChar("tagArray", numChars)
- ValueWToChar("tagArray", numChars)

ASCII Tables

Character Set (0 - 127)							
Code	Char	Code	Char	Code	Char	Code	Char
0		32	[space]	64	@	96	`
1		33	!	65	A	97	a
2		34	"	66	B	98	b
3		35	#	67	C	99	c
4		36	\$	68	D	100	d
5		37	%	69	E	101	e
6		38	&	70	F	102	f
7		39	'	71	G	103	g
8	**	40	(72	H	104	h
9	**	41)	73	I	105	i
10	**	42	*	74	J	106	j
11		43	+	75	K	107	k
12		44	,	76	L	108	l
13	**	45	-	77	M	109	m
14		46	.	78	N	110	n
15	☼	47	/	79	O	111	o
16	+	48	0	80	P	112	p
17	◀	49	1	81	Q	113	q
18	↓	50	2	82	R	114	r
19		51	3	83	S	115	s
20		52	4	84	T	116	t
21		53	5	85	U	117	u
22	T	54	6	86	V	118	v
23	┆	55	7	87	W	119	w
24	↑	56	8	88	X	120	x
25	┆	57	9	89	Y	121	y
26	→	58	:	90	Z	122	z
27		59	;	91	[123	{
28		60	<	92	\	124	
29		61	=	93]	125	}
30	-	62	>	94	^	126	~
31		63	?	95	_	127	□

Character Set (128 – 255)							
Code	Char	Code	Char	Code	Char	Code	Char
128	€	160	[space]	192	À	224	à
129	□	161	¡	193	Á	225	á
130	,	162	¢	194	Â	226	â

131	f	163	£	195	Ä	227	ã
132	„	164	¤	196	Å	228	ä
133	...	165	¥	197	À	229	å
134	†	166		198	Æ	230	æ
135	‡	167	§	199	Ç	231	ç
136	^	168	¨	200	È	232	è
137	‰	169	©	201	É	233	é
138	Š	170	ª	202	Ê	234	ê
139	‹	171	«	203	Ë	235	ë
140	Œ	172	¬	204	Ì	236	ì
141	⏏	173		205	Í	237	í
142	Ž	174	®	206	Î	238	î
143	⏏	175	¯	207	Ï	239	ï
144	⏏	176	°	208	Ð	240	ð
145	'	177	±	209	Ñ	241	ñ
146	'	178	²	210	Ò	242	ò
147	"	179	³	211	Ó	243	ó
148	"	180	´	212	Ô	244	ô
149	·	181	µ	213	Õ	245	õ
150	–	182	¶	214	Ö	246	ö
151	—	183	·	215	×	247	÷
152	~	184	¸	216	Ø	248	ø
153	™	185	¹	217	Ù	249	ù
154	š	186	º	218	Ú	250	ú
155	›	187	»	219	Û	251	û
156	œ	188	¼	220	Ü	252	ü
157	⏏	189	½	221	Ý	253	ý
158	ž	190	¾	222	Þ	254	þ
159	ÿ	191	¿	223	ß	255	

 **Note:**

** Values 8, 9, 10, and 13 convert to backspace, tab, linefeed, and carriage return characters, respectively. They have no graphical representation, but depending on the application, they may affect the visual display of text.

⏏ indicates that it is not supported on the current platform.

Asc2Str(numChar1, numChar2, ..., numCharN)

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Converts decimal values into their corresponding ASCII characters.
- **Parameters:**

numChar (1-N)	Tag or expression containing a decimal value to be converted into an ASCII character.
----------------------	---

- **Returned Values:** Returns a string of ASCII characters corresponding to the decimal values entered.
- **Examples:**

Tag Name	Expression
Tag	Asc2Str(67) // Returned value = C
Tag	Asc2Str(83, 116, 117, 100, 105, 111) // Returned value = IWS

CharToValue("tagName", "tagArray")


Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Converts a string into an array of integer values (using bytes).
- **Parameters:**

"tagName"	The name of the string tag, whose value will be converted.
"tagArray"	The name of the array tag receiving the integer values.

- **Returned Values:** No values are returned from this function.
- **Examples:**

Tag Name	Expression
	CharToValue("StrTag", "Array[1]") // If StrTag = " IWS" then Array[1] will be set to 83 ("S" in ASCII), Array[2] to 116 ("t" in ASCII) and so on.

 **Note:**
You cannot use this function for UNICODE characters because it converts the lower byte of UNICODE characters only.

CharToValueW("tagName", "tagArray")


Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Converts a string into an array of integer values (using words instead of bytes).
- **Parameters:**

"tagName"	The name of the string tag, whose value will be converted.
"tagArray"	The name of the array tag receiving the integer values.

- **Returned Values:** No values are returned from this function.
- **Examples:**

Tag Name	Expression
	CharToValue("StrTag", "Array[1]") // If StrTag = " IWS" then Array[1] would be set to 29779 ("St" in ASCII), Array[2] to 25717 ("ud" in ASCII) and so on.

 **Note:**
This function can be useful when converting UNICODE characters into codes.

ClassMembersToStrVector ("strClassTag", numStartPos, numNumPos, "strArrayTag", optBooStartPosTarget)

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Transfers values from class tags to array tags.

- **Parameters:**

strClassTag	The string value containing the class tag name.
numStartPos	Start position (array index) of the strClassTag
numNumPos	Number of positions (array indexes) to be transferred from the strClassTag.
strArrayTag	String value containing the array tag that will receive the values from the strClassTag.
optBooStartPosTarget	Start position (array index) of the strArrayTag. If omitted, the default value 1 is used.

- **Returned Values:**

-6	Array size of strClassTag is not big enough for numStartPos
-5	strClassTag tag is not a Class Tag
-4	strClassTag tag not found
-3	strArrayTag tag not found
-2	Invalid data type of the parameters
-1	Invalid number of parameters
0	Transferred successfully

- **Examples:**

Tag Name	Expression
Tag	ClassMembersToStrVector ("Classtag", 5, 3, "Arraytag")
Tag	ClassMembersToStrVector ("Classtag", 5, 3, "Arraytag" , 0)

Tag	ClassMembersToStrVector (TagName, 0, 1, ArrayName)
-----	---

**Note:**

If the strClassTag tag has more than one member, the value of each member will be transferred to the strArrayTag. Therefore, it is important to make sure that the array size of the strArrayTag tag is big enough to receive all values from the strClassTag tag.

NCopy(strSource, numStartChar, numQtdChar)

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Copies a defined section of a larger string.
- **Parameters:**

strSource	String tag containing the source string.
numStartChar	Integer tag containing a number corresponding to the first character being copied.
numQtdChar	Integer tag containing the number of characters to be copied.

- **Returned Values:** Returns a string that is part of the source string (as defined by the function).
- **Examples:**

Tag Name	Expression
Tag	Ncopy("IWS version 6.1", 7, 7) // Returned value = version
Tag	Ncopy("Technical Reference Manual", 0, 9) // Returned value = Technical

**Note:**

The first character in the string will be assigned the value 0.

Num(strValue)


Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Converts a string into a float.
- **Parameters:**

StrValue	String tag containing the number of characters to be converted into float format.
-----------------	---

- **Returned Values:** Returns the number (formerly in a string format) in float format.
- **Examples:**

Tag Name	Expression
Tag	Num("321654.987") // Returned value = 321654.987
Tag	Num("5.6589626246") // Returned value = 5.6589626246

 **Note:**
The float string cannot use characters other than the numbers (0...9) and a decimal point (.) or the program returns the value 0.0.

Str(numValue)

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Converts a number into a string.
- **Parameters:**

numValue	Integer or float tag containing a number to be converted to a string.
-----------------	---

- **Returned Values:** Returns the string, in a float format.
- **Examples:**

Tag Name	Expression
Tag	Str(321654.987) // Returned value = "321654.987"
Tag	Str(5.65896246) // Returned value = "5.658962"

Str2Asc(strChar)

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Converts an ASCII character into its corresponding ASCII code.
- **Parameters:**

strChar	String Tag containing an ASCII character to be converted into ASCII code.
----------------	---

- **Returned Values:** Returns a decimal value corresponding to the ASCII character entered.
- **Examples:**

Tag Name	Expression
Tag	Str2Asc("C") // Returned value = 67
Tag	Str2Asc("o") // Returned value = 111

StrCompare(*strValue1*, *strValue2*)

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Compares two strings to see if they are identical.
- **Parameters:**

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 Values:**

-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	StrCompare("Text1", "Text2") // Returned value = -1
Tag Tag1 = "Text " Tag2 = "Text "	StrFromReal(Tag1, Tag2) // Returned value = 0
Tag Tag1 = "Text1" Tag2 = "Text2"	StrFromReal(Tag1, Tag2) // Returned value = -1

StrCompareNoCase(*strValue1*, *strValue2*)

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** 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”).

- **Parameters:**

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 Values:**

-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	StrCompareNoCase("Text1", "TEXT1") // Returned value = 0
Tag Tag1 = "Text1" Tag2 = "TEXT1"	StrFromRealNoCase(Tag1, Tag2) // Returned value = 0

StrFromInt (numValue, numBase)

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Converts an integer into its string representation in another base number system, such as binary (base-2) or octal (base-8).

 **Note:**
If you do not need to change the base, then use the *Str()* function instead.

- **Parameters:**

numValue	An integer, or a tag of Integer type. This is the numeric value to be converted into a string.
numBase	An integer, or a tag of Integer type. This specifies which base number system to convert into.

- **Returned Values:** 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.
- **Examples:**

Tag Name	Expression
Tag	StrFromInt(26, 2) // Returned value = "11010"
Tag	StrFromInt(26, 8) // Returned value = "32"

StrFromReal(numValue, numPrecision, "strType")

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Converts a real number into its string representation, in either floating-point or exponential notation.
- **Parameters:**

numValue	Any numeric value, or a tag of Real type. This is the value to be converted into a string.								
numPrecision	An integer, or a tag of Integer type. This specifies how many decimal places will be shown.								
strType	A single-character string. This specifies how the returned value will be formatted, as described in the following table: <table border="1" data-bbox="662 974 1382 1203"> <thead> <tr> <th>Value of strType</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>f</td> <td>Formatted in floating-point notation.</td> </tr> <tr> <td>e</td> <td>Formatted in exponential notation with a lower-case "e".</td> </tr> <tr> <td>E</td> <td>Formatted in exponential notation with an upper-case "E".</td> </tr> </tbody> </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".
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 Values:** This function returns a string representation of the given numeric value, with the specified precision and notation. The returned value can be stored in any tag of [String](#) type.
- **Examples:**

Tag Name	Expression
Tag	StrFromReal(263.355, 2, "f") // Returned value = "263.36"
Tag	StrFromReal(263.355, 2, "e") // Returned value = "2,63e+002"
Tag	StrFromReal(263.355, 2, "E") // Returned value = "2,63E+002"

StrFromTime(numUTCTime, numType)

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** 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.

 **Note:**

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.

- **Parameters:**

numUTCTime	An integer, or a tag of Integer type. A timestamp given in UTC standard notation.												
numType	An integer, or a tag of Integer type. Specifies the format of the resulting string, as described in the following table: <table border="1" data-bbox="662 1129 1382 1549"> <thead> <tr> <th>Value of numType</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Displays the <i>date</i> in the same format that is selected in the Control Panel on the local computer.</td> </tr> <tr> <td>2</td> <td>Displays the <i>time</i> in the same format that is selected in the Control Panel on the local computer.</td> </tr> <tr> <td>3</td> <td>Displays a standard 24-character string that shows both date and time.</td> </tr> <tr> <td>4</td> <td>Displays the abbreviated name of the day of the week.</td> </tr> <tr> <td>5</td> <td>Displays the full name of the day of the week.</td> </tr> </tbody> </table>	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.
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 Values:** 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.
- **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)</code> // Returned value = "1/1/70"
Tag	<code>StrFromTime(86400, 2)</code> // Returned value = "07:00:00 PM"
Tag	<code>StrFromTime(86400, 3)</code> // Returned value = "Thu Jan 01 19:00:00 1970"
Tag	<code>StrFromTime(86400, 4)</code> // Returned value = "Thu"
Tag	<code>StrFromTime(86400, 5)</code> // Returned value = "Thursday"

StrGetElement(strSource, strDelimiter, numElementNumber)

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Gets a specific element from a string source.
- **Parameters:**

strSource	String tag containing 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 on.

- **Returned Values:** Returns the element (string value) retrieved from the **strSource**.
- **Examples:**

Tag Name	Expression
Tag	StrGetElement("a b c", " ", 2) // Returned value = "b"
Tag	StrGetElement("a,b,c", ",", 3) // Returned value = "c"

StrLeft(strSource, numQtdChar)

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Copies the first characters of a larger string.
- **Parameters:**

strSource	String tag containing the source string.
numQtdChar	Integer tag containing the number of characters to be copied.

- **Returned Values:** Returns a string containing the left-most characters in the source string.
- **Examples:**

Tag Name	Expression
Tag	StrLeft("IWS version 6.1", 8) // Returned value = IWS v
Tag	StrLeft ("Technical Reference Manual", 9) // Returned value = Technical

StrLen(strSource)

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Determines the length of a string.
- **Parameters:**

strSource	String tag containing the string.
------------------	-----------------------------------

- **Returned Values:** Returns an integer that is the number of characters in the string.
- **Examples:**

Tag Name	Expression
Tag	StrLen("IWS version 6.1") // Returned value = 18
Tag	StrLen("Technical Reference Manual") // Returned value = 26

StrLower(strSource)

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Converts a string to all lower case characters.
- **Parameters:**

strSource	String tag containing the string to be converted.
------------------	---

- **Returned Values:** Returns the string, where all the characters are in lowercase.
- **Examples:**

Tag Name	Expression
Tag	StrLower("IWS version 6.1") // Returned value = "IWS version 6.1"
Tag	StrLower("Technical Reference Manual") // Returned value = "technical reference manual"

StrRChr(strSource, strChrSequence)

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Isolates the final occurrence of a character sequence within a string.
- **Parameters:**

StrSource	String tag containing the source string.
StrCharSequence	String tag containing the reference string.

- **Returned Values:** Returns a string of characters following the last occurrence of a character within the source string.
- **Examples:**

Tag Name	Expression
Tag	StrRChr("IWS version 6.1", "I") // Returned value = "ion 6.1"
Tag	StrRChr("Technical Reference Manual", "a") // Returned value ="a"

StrRight(strSource, numQtdChar)

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Copies the last characters in a larger string.
- **Parameters:**

StrSource	String tag containing the source string.
NumQtdChar	Integer tag containing the number of characters to be copied.

- **Returned Values:** Returns a string containing the right-most characters in a source string.
- **Examples:**

Tag Name	Expression
Tag	StrRight("IWS version 6.1", 8) // Returned value = "sion 6.1"
Tag	StrRight("Technical Reference Manual", 9) // Returned value = "ce Manual"

StrSetElement(strSource, strDelimiter, numElementNumber, strValue)

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Gets a specific element from a string source.
- **Parameters:**

strSource	String tag containing 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 on.
strValue	String value that will be written to the numElementNumber of the strSource string tag.

- **Returned Values:** Returns the string value updated from the **strValue**.
- **Examples:**

Tag Name	Expression
Tag	StrSetElement(StringTag," ", 2, "abcd")
Tag	StrSetElement(StringTag,"", 3, "defg")

StrStr(strSource, strCharSequence)

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Isolates the first occurrence of a character sequence within a string.
- **Parameters:**

strSource	String tag containing the source string.
strCharSequence	String tag containing the reference string.

- **Returned Values:** Returns the string of characters following the first occurrence of a character within the source string.
- **Examples:**

Tag Name	Expression
Tag	StrStr("IWS version 6.1", "i") // Returned value = "ion 6.1"
Tag	StrStr("Technical Reference Manual", "a") // Returned value = "al Reference Manual"

StrStrPos(strSource, strCharSequence)


Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Finds the first occurrence of a character within a string.
- **Parameters:**

strSource	String tag containing the source string.
strCharSequence	String tag containing the reference string.

- **Returned Values:** Returns an integer corresponding to the first occurrence of a character within the source string.
- **Examples:**

Tag Name	Expression
Tag	StrStrPos("IWS version 6.1", "i") // Returned value = 4
Tag	StrStrPos("Technical Reference Manual", "a") // Returned value = 7

 **Note:**
The first character in the string assigned the value 0.

StrTrim(strReference, optNumFlag)

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Removes unwanted spaces from a string.
- **Parameters:**

strReference	A string tag containing the source string.
optNumFlag	<p>An <i>optional</i> integer tag.</p> <ul style="list-style-type: none"> • If <code>optNumFlag = 0</code>, then IWS removes spaces from both the beginning and end of the string. • If <code>optNumFlag = 1</code>, then IWS removes spaces from the beginning of the string only. • If <code>optNumFlag = 2</code>, then IWS removes spaces from the end of the string only. • If <code>optNumFlag = 3</code>, then IWS removes all spaces except for single spaces between words.

- **Returned Values:** 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 6.1 ")</code> // Returned value = "Studio version 6.1"
Tag	<code>StrTrim(" Studio version 6.1 ", 0)</code> // Returned value = "Studio version 6.1"
Tag	<code>StrTrim(" Studio version 6.1 ", 1)</code> // Returned value = "Studio version 6.1 "
Tag	<code>StrTrim(" Studio version 6.1 ", 2)</code> // Returned value = " Studio version 6.1"
Tag	<code>StrTrim(" Studio version 6.1 ", 3)</code> // Returned value = "Studio version 6.1"

StrTrimAll(strReference, optStrTrimChar)

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Eliminates a specific char from the whole string.
- **Parameters:**

strReference	A string tag containing 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 Values:** Returns a string equal to `strReference` minus the characters removed by the function.
- **Examples:**

Tag Name	Expression
Tag	StrTrimAll("IWS version 6.1 ", "") // Returned value = "IWS version 6.1"
Tag	StrTrimAll("IWS version 6.1 ", ".") // Returned value = "IWS version 6.1"

StrUpper(strSource)

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Converts a string to all uppercase characters.
- **Parameters:**

strSource	String tag containing the string.
------------------	-----------------------------------

- **Returned Values:** Returns the string with all characters in uppercase.
- **Examples:**

Tag Name	Expression
Tag	StrUpper("IWS version 6.1") // Returned value = "IWS version 6.1"
Tag	StrUpper("Technical Reference Manual") // Returned value = "TECHNICAL REFERENCE MANUAL"

ValueToChar("tagArray", numChars)

Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Converts a string into an array of integer values (using bytes).
- **Parameters:**

"tagArray"	Name of array tag containing the integer values of the characters to be converted.
NumChars	Integer tag containing the number of characters to be converted.

- **Returned Values:** Returns a string with characters defined by values in the Array tags.
- **Examples:**

Tag Name	Expression
Tag	ValueToChar("Array", 3) // If Array[0] = 65, Array[1] = 66, and Array[2] = 67 then the returned value will be "ABC"
Tag	ValueToChar("Array[10]", 3) // If Array[10] = 65, Array[11] = 66, and Array[12] = 67 then the returned value will be "ABC"

 **Note:**

You cannot use this function for UNICODE characters, because it converts the lower bytes of UNICODE characters only.

ValueWToChar("tagArray", numChars)


Group	String
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Converts a string into an array of integer values (using words).
- **Parameters:**

"tagArray"	Name of array tag containing integer values of the characters to be converted.
numChars	Integer tag containing the number of characters to be converted.

- **Returned Values:** Returns a string with characters defined by values in the Array tags.
- **Examples:**

Tag Name	Expression
Tag	ValueWToChar("Array", 3) // If Array[0] = 29779, Array[1] = 25717, and Array[2] = 28521 then the returned value will be " IWS"

 **Note:**
You may find this function especially useful when converting UNICODE characters into codes.

Date and Time Functions

This section describes the following InduSoft Web Studio Date and Time functions:

- `ClockGetDate (numSeconds)`
- `ClockGetDayOfTheWeek (numSeconds)`
- `ClockGetTime (numSeconds)`
- `DateTime2Clock (strDate, strTime)`
- `GetClock ()`
- `Hour2Clock (strTime)`
- `SetSystemDate (strDate)`
- `SetSystemTime (strTime)`

`ClockGetDate (numSeconds)`


Group	Date and Time
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Calculates the date, based on how many seconds have elapsed since 19:00:00 EST 12/31/1969 (taking into account the current time zone of the computer).
- **Parameters:**

numSeconds	Integer tag containing the number of seconds elapsed since 19:00:00 ETS 12/31/1969.
-------------------	---

- **Returned Values:** 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.

ClockGetDayOfTheWeek (numSeconds)

Group	Date and Time
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Calculates the day of the week, based on how many seconds have elapsed since 19:00:00 EST 12/31/1969 (taking into account the current time zone of the local computer).

- **Parameters:**

numSeconds	Integer tag containing the number of seconds elapsed since 19:00:00 ETS 12/31/1969
-------------------	--

- **Returned Values:** 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	ClockGetDayOftheWeek(0)// If the computer is in the Central time zone. Returned value = 3
Tag	ClockGetDate(1018886359) // 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 (numSeconds)

Group	Date and Time
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Calculates the time based on how many seconds have elapsed since 19:00:00 EST 12/31/1969 (taking into account the current time zone as specified on the local computer).

- **Parameters:**

NumSeconds	Integer tag containing the number of seconds elapsed since 19:00:00 ETS 12/31/1969.
-------------------	---

- **Returned Values:** Returns the time calculated in string format.

- **Examples:**

Tag Name	Expression
Tag	ClockGetTime(0) // If the computer is in the Central time zone. Returned value = 18:00:00
Tag	ClockGetTime(1018886359) // 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(strDate, strTime)

Group	Date and Time
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Calculates how many seconds have elapsed since 19:00:00 EST 12/31/1969 (taking into account the current time zone specified on the local computer.)

- **Parameters:**

StrDate	String tag containing the date to be used in the calculation.
StrTime	String tag containing the time to be used in the calculation.

- **Returned Values:** Returns the number of seconds that have elapsed since 19:00:00 EST 12/31/1969.

- **Examples:**

Tag Name	Expression
Tag	DateTime2Clock ("12/31/1969", "18:00:00") // If the computer is in the Central time zone. Returned value = 0
Tag	DateTime2Clock ("04/15/2002", "10:59:19") // 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 ()

Group	Date and Time
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Calculates how many seconds have elapsed since 19:00:00 EST 12/31/1969 at the moment the function was run (taking into account the current time zone, as specified on the local computer).
- **Returned Values:** Returns the number of seconds that have elapsed since 19:00:00 EST 12/31/1969 at the moment the function was run.
- **Examples:**

Tag Name	Expression
Tag	GetClock() // If executed at 10:59:19 AM April 15 th 2002 CST. Returned value = 101886359
Tag	GetClock() // 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(strTime)

Group	Date and Time
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Converts time in the **HH:MM:SS** format into seconds.
- **Parameters:**

strTime	String tag containing the number of hours, minutes, and seconds in HH:MM:SS format.
----------------	--

- **Returned Values:** Returns the number of seconds equivalent to the total number of hours, minutes, and seconds specified.
- **Examples:**

Tag Name	Expression
Tag	Hour2Clock("01:00:00") // Returned value = 3600
Tag	Hour2Clock("10:01:01") // Returned value = 36061

SetSystemDate(strDate)

Group	Date and Time
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Sets the date in the operating system's clock.
- **Parameters:**

strDate	String tag containing the date in MM/DD/YYYY format in which to set the clock.
----------------	---

- **Returned Values:** Returns no values.
- **Examples:**

Tag Name	Expression
	SetSystemDate("04/15/2002") // Sets the system clock to April 15 th 2002.

SetSystemTime(strTime)

Group	Date and Time
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Sets the time in the operating system's clock.
- **Parameters:**

strTime	String tag containing the time in HH:MM:SS format in which to set the clock.
----------------	---

- **Returned Values:** No Returned Value.
- **Examples:**

Tag Name	Expression
	SetSystemTime("15:45:18") // Sets the system clock to 3:45:18 PM.

Trigonometric Functions

This section describes the following InduSoft Web Studio Trigonometric functions:

- `ACos(numValue)`
- `ASin(numValue)`
- `ATan(numValue)`
- `Cos(numAngle)`
- `Cot(numAngle)`
- `Pi()`
- `Sin(numAngle)`
- `Tan(numAngle)`

ACos(numValue)

Group	Trigonometric
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Calculates the Arc Cosine of a value.
- **Parameters:**

NumValue	Numerical tag from which the Arc Cosine will be taken.
-----------------	--

- **Returned Values:** Returns the Arc Cosine of **numValue** in radians.
- **Examples:**

Tag Name	Expression
Tag	<code>ACos(1)</code> // Returned value = 0.000000
Tag	<code>ACos(0)</code> // Returned value = 1.570796

ASin(numValue)

Group	Trigonometric
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Calculates the Arc Sine of a value.
- **Parameters:**

NumValue	Numerical tag, from which to take the Arc Sine.
-----------------	---

- **Returned Values:** Returns the Arc Sine of **numValue** in radians.
- **Examples:**

Tag Name	Expression
Tag	ASin(1) // Returned value = 1.570796
Tag	ASin(0) // Returned value = 0.000000

ATan (numValue)

Group	Trigonometric
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Calculates the Arc Tangent of a value.
- **Parameters:**

NumValue	Numerical tag, from which to take the Arc Tangent.
-----------------	--

- **Returned Values:** Returns the Arc Tangent of **numValue** in radians.
- **Examples:**

Tag Name	Expression
Tag	ATan(1) // Returned value = 0.785398
Tag	ATan(0) // Returned value = 1.570796

Cos (numAngle)

Group	Trigonometric
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Calculates the Cosine of a value.
- **Parameters:**

NumAngle	Numerical tag containing the Angle (in radians) from which to calculate the Cosine.
-----------------	---

- **Returned Values:** Returns the Cosine of **numAngle**.
- **Examples:**

Tag Name	Expression
Tag	Cos(1.570796) // Returned value = 0.000000
Tag	Cos(0) // Returned value = 1.000000

Cot (numAngle)


Group	Trigonometric
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Calculates the Cotangent of a value.
- **Parameters:**

NumAngle	Numerical tag containing the Angle (in radians) from which to calculate the Cotangent.
-----------------	--

- **Returned Values:** Returns the Cotangent of **numAngle**.
- **Examples:**

Tag Name	Expression
Tag	Cot(0.785398) // Returned value = 1.000000
Tag	Cot(0) // Returned value = 0.00000


 **Note:**
Although mathematically the tangent of Pi is infinite, IWS only returns the largest number possible.

Pi ()

Group	Trigonometric
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Calculates Pi.
- **Returned Values:** Returns Pi.
- **Examples:**

Tag Name	Expression
Tag	Pi() // Returned value = 3.141593

 **Note:**
Even though the **Pi ()** function does not have any arguments, you must include the parentheses or IWS will look for a tag named **Pi**.

Sin(numAngle)

Group	Trigonometric
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Calculates the Sine of a value.
- **Parameters:**

NumAngle	Numerical tag containing the Angle (in radians) from which to calculate the Sine.
-----------------	---

- **Returned Values:** Returns the Sine of **numAngle**.
- **Examples:**

Tag Name	Expression
Tag	Sin(0) // Returned value = 0.000000
Tag	Sin(1.570796) // Returned value = 1.000000

Tan(numAngle)


Group	Trigonometric
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Calculates the Tangent of a value.
- **Parameters:**

NumAngle	Numerical tag containing the Angle (in radians) from which to calculate the Tangent.
-----------------	--

- **Returned Values:** Returns the Tangent of **numAngle**.
- **Examples:**

Tag Name	Expression
Tag	Tan(0) // Returned value = 0.00000
Tag	Tan(0.785398) // Returned value = 1.00000

 **Note:**
Although mathematically the tangent of $\frac{1}{2}$ Pi is infinite, IWS only returns the largest number possible.

Opening and Closing Windows Functions

This section describes the following InduSoft Web Studio Screen functions:

- `Close(strScreen, optNumID)`
- `Open(strScreen, optnumX1 , optnumY1 , optnumX2 , optnumY2, numResizeFlag, optNumID)`
- `OpenPrevious(optNumX1 , optNumY1 , optNumX2 , optNumY2)`

Close (strScreen, optNumID)

Group	Opening and Closing Windows
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Not Supported

 **Note:**

This function cannot be used with Tasks (see page 8–1) or in a VBScript sample in the Global Procedures interface (see page 15–8).

- **Description:** Closes an open screen.
- **Parameters:**

strScreen	String tag containing the name of the screen to be closed.
optNumID	A numeric value or tag; 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 Values:** No returned values.
- **Examples:**

Tag Name	Expression
	<code>Close("main")</code> // Closes the "main" screen.
	<code>Close("alarms")</code> // Closes the "alarms" screen.
	<code>Close("main", 10)</code> // Closes the "main" screen with ID 10.

Caution:

When you open a screen using the Replace style, it automatically closes the screens with Replace and/or Popup attributes that are overlapped by new screen. In this case, it is not necessary to call the **Close ()** function.

Open(strScreen, optnumX1, optnumY1, optnumX2, optnumY2, numResizeFlag, optNumID)

Group	Opening and Closing Windows
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

Note:

This function cannot be used with Tasks (see page 8–1) or in a VBScript sample in the Global Procedures interface (see page 15–8).

- **Description:** Opens the specified screen.
- **Parameters:**

strScreen	String tag containing the name of the screen to be opened.
optnumX1	<i>Optional</i> Integer tag contains the X coordinate for upper-left corner of screen in pixels.
optnumY1	<i>Optional</i> Integer tag contains the Y coordinate for upper-left corner of screen in pixels.
optnumX2	<i>Optional</i> Integer tag contains the X coordinate for lower-right corner of screen in pixels.
optnumY2	<i>Optional</i> Integer tag contains the X coordinate for lower-right corner of screen in pixels.
numResizeFlag	<p>Specifies whether objects in the screen will be resized when the screen is opened:</p> <ul style="list-style-type: none"> • 0 — Objects in the screen will not be resized. • 1 — Objects in the screen will be automatically resized to fit the new dimensions of the screen, as specified by the coordinates described above. The resizing is done at the moment the screen is opened, so if the user changes the screen size after the screen is opened, then the objects will not be resized. <p>NOTE: This parameter is required if all four coordinates are specified.</p>

optNumID	You can open multiple instances of the same screen. In this case, each instance is identified by a different ID, and to close a specific instance you must specify the ID in the Close () 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 opened. This string has the following syntax... <pre>#<Label>: <Value></pre> <p>...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.</p>

Note:

- You can open the screen at the current mouse location by using **Open ("ScreenName", 1)**, or **Open ("ScreenName", 1, -1, -1, -1, ...)** if the parameters at the end are needed.
- If **optNumX2** and **optNumY2** are less than **optNumX1** and **optNumY1**, or if all four coordinates are set to -1, then the parameters are ignored and the Screen Attributes settings are used by default.
- If **optNumX1** equals **optNumX2** and **optNumY1** equals **optNumY2**, then the screen size specified in Screen Attributes is used by default but the screen location is centered at **optNumX1** and **optNumY1**.

▪ **Returned Values:**

0	Function executed successfully.
1	Function did not execute successfully.

▪ **Examples:**

Tag Name	Expression
Tag	<code>Open("main")</code> // Opens the screen using the Screen Attributes settings.
Tag	<code>Open("main", 1)</code> // Opens the screen at the current mouse location.
Tag	<code>Open("main", 1, -1, -1, -1, 0, 10)</code> // Opens the screen at the current mouse location, and assigns the screen ID 10.
Tag	<code>Open("main", 500, 250, 500, 250, 0, 10)</code> // Opens the screen centered at X=500 Y=250.

Tag	<pre>Open("main", -1, -1, -1, -1, 1, 0, "#Mne1:Tag1 #Mne2:Tag2")</pre> <p>// Opens the screen, replacing the custom properties Mne1 and Mne2 with Tag1 and Tag2, respectively.</p>
-----	--

 **Caution:**

- Some Web servers are case-sensitive. If you plan to convert your screens into HTML format you should use only lowercase letters for the screen name.
- If you call the **Open ()** function from a Task such as the Script Task, then the screen will open only on the server station. In other words, the function will not work on Web Thin Clients or Secure Viewers.

OpenPrevious(optNumX1 , optNumY1 , optNumX2 , optNumY2)

Group	Opening and Closing Windows
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

Note:

This function cannot be used with Tasks (see page 8–1) or in a VBScript sample in the Global Procedures interface (see page 15–8).

- **Description:** Re-opens the last screen that was closed.
- **Parameters:**

optnumX1	Optional integer tag contains the X coordinate for upper-left corner of screen in pixels.
optnumY1	Optional integer tag contains the Y coordinate for upper-left corner of screen in pixels.
optnumX2	Optional integer tag contains the X coordinate for lower-right corner of screen in pixels.
optnumY2	Optional integer tag contains the Y coordinate for lower-right corner of screen in pixels.

- **Returned Values:**

0	Function did not execute successfully
1	Function executed successfully

- **Examples:**

Tag Name	Expression
	OpenPrevious()
	OpenPrevious0,0,800,600()

Security Functions

This section describes the following InduSoft Web Studio Security functions:

- `BlockUser(strUserName)`
- `CheckESign()`
- `CreateUser(strUserName, strGroup, strPassw, strUserFullName)`
- `GetUserNames(strUsers , nUserType , strGroups)`
- `GetUserPwdAging(strUser)`
- `GetUserState(strUserName)`
- `RemoveUser(strUserName)`
- `SetPassword(strUserName, optStrNewPassword)`
- `UnblockUser(strUserName)`

BlockUser(strUserName)

Group	Security
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Used to block an existing user from the security system.
- **Parameters:**

StrUserName	
	String tag containing the name of the user to block.

- **Returned Values:**

0	User blocked successfully
1	Invalid number of parameters
2	Wrong parameter type
3	User does not exist
4	It is not possible to write the data successfully

- **Examples:**

Tag Name	Expression
Tag	<code>BlockUser("Bob")</code>
Tag	<code>BlockUser("Albert")</code>

Note:

You cannot use this function to create a user name that is already being used in the application. Users created with this function do not display in the development environment *Security* folder because they are stored in a secondary database.

Tip:

You can use the **ExtUser.exe** program (located in the **Bin** folder) to manage the users in this secondary database.

CheckESign ()

Group	Security
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Prompts the user to enter their User Name and Password. This can be used to secure specific expressions, just as the **E-Sign** option is used to secure Active Objects and Dynamic Properties.
- **Parameters:** There are no parameters for this function. Calling the function displays a security dialog, where the user must enter their User Name and Password.

▪ **Returned Values:**

FALSE	User Name and Password not accepted
TRUE	User Name and Password accepted

▪ **Examples:**

Tag Name	Expression
	CheckESign()

CreateUser(strUserName, strGroup, strPassw, strUserFullName)

Group	Security
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Creates a new user.
- **Parameters:**

strUserName	String tag containing the name of the user to be created.
strGroup	String tag containing the name of the group to which the user will belong.
strPassword	String tag containing a password for the user.
strUserFullName	String tag containing the full name of the user.

- **Returned Values:**

0	New user created successfully
1	Invalid number of parameters
2	Wrong parameter type
3	User already exists
4	Group does not exist
5	It is not possible to safely write the data
6	It is not possible to use the CreateUser () function
7	User full name already exists
8	Reentrant function call not allowed
9	User clicked on Cancel button when using the standard Create User window
10	Invalid password, check the minimum password size specified for the group

- **Examples:**

Tag Name	Expression
Tag	CreateUser("Bob", "Admin", "Chocolate", "Bob Smith")
Tag	CreateUser("Albert", "Engineering", "EMC2", "Albert Jones")

 **Note:**

You cannot use this function to create a user name that is already being used in the application. Users created with this function do not display in the development environment *Security* folder because they are stored in a secondary database.

 **Tip:**

You can use the **ExtUser.exe** program (located in the **Bin** folder) to manage the users in this secondary database.

GetUserNames ("tagUsers" , optnumUserType , "opttagGroups")

Group	Security
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported / Executed on Server

- **Description:**

- **Parameters:**

"tagUsers"	Name of the array tag that will receive users
optnumUserType	0- Return all users 1- Only users created during run-time 2- Only users created using the development environment
"opttagGroups"	Name of the array tag that will receive the group for each specific user

- **Returned Values:** 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 web thin client

Natural number set: Number of users

- **Examples:**

Tag Name	Expression
NumberOfUsers	GetUserNames("UsersArray") //Retrieves all users, store their names in the UsersArray tag and the number of users in the NumberOfUsers tag.
NumberOfUsers	GetUserNames("UsersArray", 1) //Retrieves all users created during run-time, store their names in the UsersArray tag and the number of users in the NumberOfUsers tag.
NumberOfUsers	GetUserNames("UsersArray", 2) //Retrieves all users created in the development environment, store their names in the UsersArray tag and the number of users in the NumberOfUsers tag.
NumberOfUsers	GetUserNames("UsersArray", 2, "Groups") //Retrieves all users created in the

	development environment, store their names in the UsersArray tag and the number of users in the NumberOfUsers tag. The group name per each user is stored in the Groups tag.
--	--

GetUserPwdAging(strUser)


Group	Security
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the time remaining before the password a specific user expires.
- **Parameters:**

strUser	String tag containing the name of the user.
----------------	---

- **Returned Values:**

≤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	GetUserPwdAging("John") // returns the number of hours before the password for the User "John" expires.
TagHoursToExpire	GetUserPwdAging(UserName) // returns the number of hours before the password for current User logged on the system expires.

GetUserState(strUserName)

Group	Security
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Use to see the current status of a selected user.
- **Parameters:**

StrUserName	String tag containing the name of the user.
--------------------	---

- **Returned Values:**

0	User is unblocked
1	User is blocked
3	User does not exist
4	It is not possible to safely write the data

- **Examples:**

Tag Name	Expression
Tag	GetUserState("Bob")
Tag	GetUserState("Albert")

RemoveUser (strUserName)

Group	Security
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Removes a user from the system.
- **Parameters:**


StrUserName	String tag containing the name of the user to be removed.
--------------------	---

- **Returned Values:**

0	User removed successfully
1	Invalid number of parameters
2	Wrong parameter type
3	User does not exist
4	It is not possible to safely write the data

- **Examples:**

Tag Name	Expression
Tag	RemoveUser("Bob")
Tag	RemoveUser("Albert")

 **Note:**
You can use this function to remove only those users you created using the **CreateUser ()** function.

SetPassword(strUserName, optStrNewPassword)

Group	Security
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Use to specify a new user password.
- **Parameters:**


StrUserName	String tag containing the name of the user.
OptStrNewPassword	<i>Optional</i> string tag containing the new password.

- **Returned Values:**

0	The new password has been set
1	User is blocked
3	User does not exist
4	It is not possible to safely write the data
5	Operation was cancelled

- **Examples:**

Tag Name	Expression
Tag	SetPassword("Bob")
Tag	SetPassword("Albert," "anemarie")

 **Note:**
If you omit the **optStrNewPassword** parameter, the **SetPassword()** function will launch an *Enter a new password* dialog, so the user can configure a new password.

UnlockUser(strUserName)

Group	Security
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Use to unblock a blocked user in the system.
- **Parameters:**

StrUserName	String tag containing the name of the user to unblock.
--------------------	--

- **Returned Values:**

0	User unblocked successfully
1	Invalid number of parameters
2	Wrong parameter type
3	User does not exist
4	It is not possible to safely write the data

- **Examples:**

Tag Name	Expression
Tag	UnlockUser("Bob")
Tag	UnlockUser("Albert")

Module Activity Functions

This section describes the following InduSoft Web Studio Module Activity functions:

- `AppActivate(strAppTitle, optnumActiv)`
- `AppIsRunning(strAppTitle)`
- `AppPostMessage(strAppTitle, numwParam, numlParam)`
- `AppSendKeys(strKeys1, strKeys2, ... , strKeysN)`
- `CleanReadQueue()`
- `CloseSplashWindow()`
- `EndTask(strTask)`
- `ExitWindows(numExitCode)`
- `IsScreenOpen(strScreen, optNumID)`
- `IsTaskRunning(strTask)`
- `IsViewerInFocus()`
- `KeyPad("strTagName", "optStrKeyboardName", optnumIsPassword, "optStrHint", optnumMin, optnumMax)`
- `LogOff()`
- `Logon(optStrUser, optStrPassword)`
- `Math(numWorksheet)`
- `PostKey(numKeyDownOrKeyUp, numwParam, numlParam)`
- `Recipe(strFunction)`
- `Report("strFunction", optnumOrientation)`
- `RunGlobalProcedureOnServer(strNameProcedure, param1, param2, ...)`
- `RunVBScript(strScript, "opttagReturnError")`
- `SendKeyObject(numEvent, strMainKey, optnumShift, optnumCtrl, optnumAlt, optStrTargetScreen, optNumID)`
- `SetAppPath(strPath)`
- `SetKeyboardLanguage(strLanguage)`
- `SetViewerInFocus()`
- `SetViewerPos(numLeft, numTop, optnumWidth, optnumHeight)`
- `ShutDown()`
- `StartTask(strTask)`
- `ViewerPostMessage(strScrTitle, numwParam, numlParam, optNumID)`
- `Wait(numMillisec)`
- `WinExec(strCommand, optnumState)`
- `WinExecIsRunning(numHandle, "stroptReturn")`

AppActivate(strAppTitle, optnumActiv)

Group	Module Activity
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Activates an application that is already open.
- **Parameters:**

StrAppTitle	String tag containing the name of the application to be activated.
OptnumActive	<p><i>Optional</i> integer tag containing the integer that corresponds to an activation option. See Windows documentation for more details on these options.</p> <ul style="list-style-type: none"> • 0 = SW_HIDE • 1 = SW_SHOWNORMAL • 2 = SW_SHOWMINIMIZED • 3 = SW_SHOWMAXIMIZED • 4 = SW_SHOWNOACTIVATE • 5 = SW_SHOW • 6 = SW_MINIMIZE • 7 = SW_SHOWMINNOACTIVATE • 8 = SW_SHOWNA • 9 = SW_RESTORE (<i>Default</i>)

- **Returned Values:** No returned values.
- **Examples:**

Tag Name	Expression
	AppActivate("Microsoft Word – test.doc", 5)

AppIsRunning(strAppTitle)

Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Verifies whether an application is open and running.
- **Parameters:**

strAppTitle	String tag containing the name of the application.
--------------------	--

- **Returned Values:**

1	Application is running.
0	Application is not running.

- **Examples:**

Tag Name	Expression
Tag	AppIsRunning("Microsoft Word – test.doc")

AppPostMessage(strAppTitle, numwParam, numlParam)

Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Sends a message to the active application.
- **Parameters:**

strAppTitle	String tag containing the name of the application.
numwParam	Integer tag containing an integer corresponding to the Windows message wParam
numlParam	Integer tag containing an integer corresponding to the Windows message lParam

- **Returned Values:** No returned values.
- **Examples:**

Tag Name	Expression
Tag	AppPostMessage("Microsoft Word – test.doc", 3, 1)

AppSendKeys(strKeys1, strKeys2, ..., strKeysN)


Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported

- **Description:** Sends keyboard commands to the active application.
- **Parameters:**

StrKeys (1-N)	String tags containing the keyboard commands to be used.
----------------------	--

- **Returned Values:** No returned values.
- **Examples:**

Tag Name	Expression
	AppSendKeys("S", "t", "u", "d", "l", "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 ()

Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Removes all reading messages from the communications module.
- **Returned Values:** No returned values.
- **Examples:**

Tag Name	Expression
	CleanReadQueue()

 **Note:** You should not use this function in new applications, but this function is still valid for applications built using earlier versions of the InduSoft Web Studio.

CloseSplashWindow ()

Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Closes the IWS splash screen.
- **Returned Values:** No returned values.
- **Examples:**

Tag Name	Expression
	CloseSplashWindow()

EndTask (strTask)

Group	Module Activity
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Stops the IWS module that is currently running.
- **Parameters:**

strTask	String tag containing the name of the task to stop (must be one of the following): <ul style="list-style-type: none"> • BGTASK: Background Tasks • VIEWER: Viewer • DBSPY: Database Spy • LOGWIN: LogWin • DRIVER <DriverName>: Driver • UNIDDECL: DDE client • UNINDDE: DDE server • UNIODBC: ODBC • TCPSERVER: TCP/IP Server • TCPCLIENT: TCP/IP Client • OPCCLIENT: OPC
----------------	---

- **Returned Values:** No returned values.
- **Examples:**

Tag Name	Expression
	EndTask("Viewer")

 **Note:**

To close a driver, you must use the following syntax:

EndTask ("Driver<DriverName>")

Where <DriverName> is the name of the driver's .dll file. For example,

EndTask ("DriverMODBU")

ExitWindows (numExitCode)

Group	Module Activity
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported

- **Description:** Closes windows in a specified manner.
- **Parameters:**

numExitCode	Integer tag containing a number (0–2) specifying how Windows will be exited. <ul style="list-style-type: none"> • 0 = Reboot Windows • 1 = Log Off Windows • 2 = Shutdown Windows
--------------------	--

- **Returned Values:** No returned values.
- **Examples:**

Tag Name	Expression
	ExitWindows(1)

IsScreenOpen(strScreen, optNumID)

Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Not Supported

Note:

This function cannot be used with Tasks (see page 8–1) or in a VBScript sample in the Global Procedures interface (see page 15–8).

- **Description:** Verifies that a screen is open.
- **Parameters:**

strScreen	String tag containing the name of the screen to be verified.
optNumID	A numeric value or tag; 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 Values:**

0	Screen is not open.
1	Screen is open.

- **Examples:**

Tag Name	Expression
Tag	IsScreenOpen("main") // Is "main" screen open?
Tag	IsScreenOpen("main", 10) // Is "main" screen with ID 10 open?

IsTaskRunning(strTask)

Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Verifies that a task is running.
- **Parameters:**

strTask	String tag containing the name of the task to be verified.
----------------	--

- **Returned Values:**

0	Task is not running.
1	Task is running.

- **Examples:**

Tag Name	Expression
Tag	IsTaskRunning("viewer")

IsViewerInFocus ()

Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Not Supported

- **Description:** Verifies that the Viewer module is in focus on the screen.
- **Returned Values:**

0	Viewer is not in focus.
1	Viewer is in focus.

- **Examples:**

Tag Name	Expression
Tag	IsViewerInFocus()

**KeyPad("strTagName", "optStrKeyboardName",
optnumIsPassword, "optStrHint", optnumMin, optnumMax)**

Group	Module Activity
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Opens a *Virtual Keyboard* dialog to write to the specified Tag.
- **Parameters:**

"strTagName"	The name of the Tag to which the <i>Virtual Keyboard</i> will write.
"optStrKeyboardName"	Type of <i>Virtual Keyboard</i> that will be launched (e.g. AlphaNumeric, EnhKeypad, or Keypad). If this parameter is omitted, then the default <i>Virtual Keyboard</i> will be launched.
optnumIsPassword	If this parameter is set with any value different from 0 (zero), the characters typed in the <i>Virtual Keyboard</i> 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 <i>Virtual Keyboard</i> .
optnumMin	Minimum and maximum numeric values for the Tag when using the Keypad keyboard type. (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.
optnumMax	

- **Returned Values:**

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 ()

Group	Module Activity
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Logs off the current user and logs on the guest user.
- **Returned Values:** No returned values.
- **Examples:**

Tag Name	Expression
	LogOff()

Logon (optStrUser, optStrPassword)

Group	Module Activity
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Logs on the specified user using the specified password. If no user is provided, opens a *logon* dialog.
- **Parameters:**

optStrUser	<i>Optional</i> string tag containing the name of the User to log on.
optStrPassword	<i>Optional</i> string tag containing the user's log-on password.

- **Returned Values:** No returned values.
- **Examples:**

Tag Name	Expression
	Logon()
	Logon("Albert", "EMC2")

Math(numWorksheet)

Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Executes the specified math worksheet.
- **Parameters:**

numWorksheet	Numerical tag containing the number of the math worksheet to be executed.
---------------------	---

- **Returned Values:** No returned values.
- **Examples:**


Tag Name	Expression
	Math(6)

⚠ 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 (numKeydownOrKeyup, numwParam, numlParam)

Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

 **Note:**
This function cannot be used with Tasks (see page 8–1) or in a VBScript sample in the Global Procedures interface (see page 15–8).

- **Description:** Posts keys to the screen in the viewer.
- **Parameters:**

numKeydownOrKeyup	Numerical tag containing a 0 (to indicate a Key down event) or a 1 (to indicate a Key up event).
numwParam	Numerical tag containing key code to be sent.
numlParam	Numerical tag containing message lParam .

- **Returned Values:** No returned values.
- **Examples:**

Tag Name	Expression
	PostKey(0, 0x24, 0)

Recipe(strFunction)

Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Activates a specified recipe function.
- **Parameters:**

strFunction	String tag specifying the operation to be performed and the recipe sheet to be used in the [Operation] : [Recipe sheet] format. Operations: <ul style="list-style-type: none"> • 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 Values:**

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	Recipe(“Save:Recipe1”)
Tag	Recipe(“Load:Recipe5”)

⚠ Cautions:

- You must be running the *Background Task* (**Execution Tasks** tab on the *Project Status* window) to execute the recipe functions.
- When this function is called on a Web 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 **Server** will still be updated on the Server.

Report ("strFunction" , optnumOrientation)

Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Sends a Report worksheet to output (hard disk, printer, or PDF).
- **Parameters:**

"strFunction"	String specifying the operation to perform and the report sheet to use in the syntax "[Operation] : [Report worksheet]" format, where: [Operation]: <ul style="list-style-type: none"> • Disk: Saves data to a data file. • Prn: Prints data directly to the default printer. • Pdf: Generates a PDF file of the report. [Report worksheet]: Name of the report that must be either saved to the disk, sent to the printer or sent/generate a pdf file (the report file name must include the .REP extension).
optnumOrientation	Set the paper orientation as follows: 0 (default) = Portrait 1 = Landscape This parameter is ignored when the Operation configured in the "strFunction" parameter is other than Prn.

 **Note:**

Some features of this function are not supported when running the application on a Windows CE 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 Values:**

0	Success
1	"strFunction" is configured with a numeric value (invalid)
2	Expression does not contain ":" (invalid)
3	Expression contains invalid Operation 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 “Project Status” on page 6-31.

▪ **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>

RunGlobalProcedureOnServer (strNameProcedure, param1, param2, ...)

Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Runs a function or sub-routine, as declared in the Global Procedures interface. The procedure is run on the server, but it can be called by any local or remote client.

- **Parameters:**

strNameProcedure	A string or a tag of String type; the name of the function or sub-routine to be run.
param1, param2, ...	Parameters that are passed to the procedure. Parameters must be passed as strings.

- **Returned Values:** This function returns whatever value is returned by the called procedure.
- **Examples:** Given the following procedure that is declared in the Global Procedures interface:

```
Function AddMe (intNumber)
  If intNumber >= 6 Then
    AddMe = 0
  Else
    AddMe = intNumber + 2
  End If
End Function
```

Tag Name	Expression
Tag	RunGlobalProcedureOnServer ("AddMe", "2") // Executes the procedure and returns a value of 4.

RunVBScript (strScript , "opttagReturnError")

Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Not Supported

- **Description:** Executes an statement in VBScript language.
- **Parameters:**

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 Values:**
 - 0: Error.
 - 1: Success.
- **Examples:**

Tag Name	Expression
TagResult	RunVBScript ("MsgBox (Time) ") // Executes the MsgBox function from VBScript and displays the current time.
	RunVBScript (TagStatement) // Executes the statement configured in the value of the string tag TagStatement.
	RunVBScript ("\$TagC=\$TagA/\$TagB", "TagError") // Writes in TagC the result of TagA divided by TagB. The error generated by the operation (if any) is written to the string tag TagError (e.g.: "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.

SendKeyObject(numEvent, strMainKey, optnumShift, optnumCtrl, optnumAlt, optStrTargetScreen, optNumID)

Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Not Supported

 **Note:**

This function cannot be used with Tasks (see page 8–1) or in a VBScript sample in the Global Procedures interface (see page 15–8).

- **Description:** Sends key codes to objects on the open screens. You can trigger "Command" dynamics from these objects using this function.
- **Parameters:**

numEvent	Numerical tag specifying the key event code, as follows: <ul style="list-style-type: none"> • 0: On Down • 1: While Down • 2: On Up
strMainKey	String tag containing the key to be sent to the object. The following tags are acceptable: <ul style="list-style-type: none"> • "F1" ... "F20" • "+ " • "- " • "/" • "*" • "HOME" • "END" • "INSERT" • "DELETE" • "DOWN" • "UP" • "LEFT" • "RIGHT" • "PAGEUP" • "PAGEDOWN" • "SPACE" • "RETURN" • "BACKSPACE" • "ESCAPE" • "A" ... "Z"
optnumShift	<i>Optional</i> numerical tag, which indicates whether to send Shift .
optnumCtrl	<i>Optional</i> numerical tag, which indicates whether to send Ctrl .
optnumAlt	<i>Optional</i> numerical tag, which indicates whether to send Alt .
optStrTargetScreen	A string or a tag of String type; the name of the screen to receive the key event code.
optNumID	A numeric value or tag; 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 Values:** No returned values.

- **Examples:**

Tag Name	Expression
	<pre>SendKeyObject(0, "R", 1, 0, 0, "main", 10) // Sends Shift-R to the "main" screen with ID 10.</pre>

 **Notes:**

- **numEvent** defines how the function executes expressions specified for **On Down**, **On While**, or **On Up** of the object's **Command** dynamic.
- 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** dynamic) just once.
- The **numShift**, **numCtrl**, **numAlt**, and **strTargetScreen** parameters are optional; however, if you configure one of them, you must configure the others too.

SetAppPath (strPath)

Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Sets the new path for the application. After this function is executed, IWS will look for all the configuration files (screens, alarms, trends, database, events) in this new path.

- **Parameters:**

StrPath	String tag containing the file path.
----------------	--------------------------------------

- **Returned Values:**

0	Failed to set path.
1	Succeeded in setting path

- **Examples:**

Tag Name	Expression
	SetAppPath("C:\InduSoft Web Studio\")

- **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.

SetKeyboardLanguage (strLanguage)

Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Use this function to set the language of the Virtual Keyboards.
- **Parameters:**

strLanguage	String tag with the language used for the virtual keyboards. The currently available options are: "EN" (English, <i>default</i>), and "GE" (German).
--------------------	---

- **Returned Values:**

0	Success
1	Error

- **Examples:**

Tag Name	Expression
Tag	SetKeyboardLanguage("EN")
Tag	SetKeyboardLanguage(TagLanguage)

SetViewerInFocus ()

Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported

- **Description:** Sets the focus to the Viewer task.
- **Returned Values:** No returned values.
- **Examples:**

Tag Name	Expression
	SetViewerInFocus()

SetViewerPos (numLeft, numTop, optnumWidth, optnumHeight)

Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported

- **Description:** Sets the height, width, and position of the Viewer.
- **Parameters:**

NumLeft	Numerical tag specifying the left-side position of the Viewer in pixels.
NumTop	Numerical tag specifying the top-side position of the Viewer in pixels.
OptnumWidth	<i>Optional</i> numerical tag specifying the Viewer width in pixels.
OptnumHeight	<i>Optional</i> numerical tag containing the Viewer height in pixels.

- **Returned Values:**

0	Error
1	Success

- **Examples:**

Tag Name	Expression
Tag	SetViewerPos(50, 50, 640, 480)

 **Note:**

When you omit the optional parameters (`numWidth` and `numHeight`), IWS gets size of the Viewer window from the application resolution.

ShutDown ()

Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Not Supported

- **Description:** Shuts down all of the active application modules.
- **Returned Values:** No returned values.
- **Examples:**

Tag Name	Expression
	ShutDown()

 **Caution:**
This function does not close the development environment, *Database Spy*, or *LogWin*.

StartTask (strTask)

Group	Module Activity
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Starts an IWS module not currently running.
- **Parameters:**

StrTask	String tag containing the name of the task to start (must be one of the following tasks): <ul style="list-style-type: none"> • BGTASK: Background Tasks • VIEWER: Viewer • DBSPY: Database Spy • LOGWIN: LogWin • DRIVER: Driver • UNIDDECL: DDE Client • UNINDDE: DDE Server • UNIODBC: ODBC • TCPSEVER : TCP/IP Server • TCPCLIENT: TCP/IP Client • OPCCIENT: OPC
----------------	---

- **Returned Values:** No returned values.
- **Examples:**

Tag Name	Expression
	EndTask("Viewer")

Note:
 The **StartTask("Driver")** function starts all drivers configured in the application.
 To start a specific driver, you must use the **Winexec()** function. For example,
Winexec("<IWS path>\BIN\Studio Manager.exe" + " " + "<IWS Path>\Bin\Driver.dll" + " " + "<DriverName>")

ViewerPostMessage(**strScrTitle**, **numwParam**, **numlParam**, **optNumID**)

Group	Module Activity
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

 **Note:**

This function cannot be used with Tasks (see page 8–1) or in a VBScript sample in the Global Procedures interface (see page 15–8).

- **Description:** Sends an internal message to the Viewer.
- **Parameters:**

strScrTitle	String tag containing the name of the screen to which the message will be posted.
numwParam	Numerical tag containing the <code>wParam</code> of the Windows message.
numlParam	Numerical tag containing the <code>lParam</code> of the Windows message.
optNumID	A numeric value or tag; 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 Values:** No returned values.
- **Examples:**

Tag Name	Expression
	<pre>ViewerPostMessage("main", 16, 3, 1, 10) // Sends message 16 to the "main" screen with ID 10.</pre>

Wait (numMillisec)

Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Interrupts the Math worksheet's execution for a specified number of milliseconds.
- **Parameters:**

NumMillisec	Integer tag containing the number of milliseconds to wait.
--------------------	--

- **Returned Values:** No returned values.
- **Examples:**

Tag Name	Expression
	Wait(1000)

 **Caution:**

You can use the **Wait (numMillisec)** function in *Math* worksheets only. However, **it is dangerous** to use this function anywhere in your application. **Wait (numMillisec)** pauses the application, and any information coming into the application during a wait will be ignored.

WinExec(strCommand, optnumState, optnumSync, "opttagReturnOrHandle")

Group	Module Activity
Execution	Asynchronous / Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Executes a Windows command line.
- **Parameters:**

strCommand	String tag containing the command line to execute.
optnumState	<p><i>Optional</i> numeric tag or constant containing an integer that defines the initial state of a new application:</p> <ul style="list-style-type: none"> • 0: Hides the application and gives control to another one. • 1: Activates and displays the application (default). • 2: Activates the application and displays it as an icon. • 3: Activates the application and maximizes it. • 4: Shows the application with its recent size. The application is still active. • 7: Shows the application as an icon. The application is still active.
optnumSync	<p><i>Optional</i> numeric tag or constant that specifies if the function will execute synchronously or asynchronously. When executing synchronously, the function will return when the executable finish its execution; when executing asynchronously the function will return immediately. To verify if a program started asynchronously has been finished, one should use the fourth parameter and the WinExeclsRunning function.</p> <ul style="list-style-type: none"> • 0: Asynchronous mode. • 1: Synchronous mode. <p>Default is 0.</p>
"opttagReturnOrHandle"	You should specify a string with the tag name that will receive either the Return code (when executing Synchronously) of the executable file or the program handle that can be used in the WinExeclsRunning function to determine whether the executable is still running.

- **Returned Values:**

0	Command was not executed successfully.
1	Command was executed successfully.

- Examples:

Tag Name	Expression
Tag	WinExec("C:\WinNT\System32\notepad.exe", 4) // Starts the Notepad and continues executing the next lines in the script
Tag	WinExec("C:\WinNT\System32\mspaint.exe") // Starts MS Paint and continues executing the next lines in the script
Tag	WinExec("C:\MyTasks.bat", 0, 1, "result") // Starts a batch file, executes it in hidden mode, and continues executing the next lines in the script only when the batch finishes its execution. The result is stored in the result integer tag.
Tag	WinExec("C:\MyTasks.bat", 0, 0, "handle") // Starts a batch file, executes it in hidden mode and goes to the next line in the script. The tag handle receives a number that can be passed to the WinExecIsRunning function to determine whether the batch is still executing or not [WinExecIsRunning(handle)].

WinExecIsRunning(numHandle, "opttagReturn")

Group	Module Activity
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- Description:** Indicates whether a program started by the **WinExec()** function is still running.

- Parameters:**

numHandle	Handle number stored in the tag in the WinExec stroptReturnorHandle parameter.
"opttagReturn"	Tag that receives the code returned by the program executed by the WinExec() function.

- Returned Values:**

- 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

This section describes the following InduSoft Web Studio File functions:

- DeleteOlderFiles (strPath, strMask, strDate)
- DirCreate (strDirectory , optBooFullPath)
- DirDelete (strDirectory , optBooEmptyOnly)
- DirLength (strPath)
- DirRename (strPath , strDirectoryFrom , strDirectoryTo)
- FileCopy (strSourceFile, strTargetFile, optnumTimeOut)
- FileDelete(strFile)
- FileLength(strFile)
- FileRename(strOldName, strNewName)
- FileWrite(strFileName, strWriteText, numoptAppend)
- FindFile(strFile, "opttagFilesFound", optnumTimeOut)
- FindPath(strPathName)
- GetFileAttributes(strFile)
- GetFileTime(strFileName, numFormat)
- GetHstInfo(strFileName, numInfoType)
- GetLine(strFileName, strSeqChar, "tagStore", optnumCase, "optOveflowTag")
- HST2TXT(strStartDate, strStartTime, numDuration, numGroup, strTargetFile, optStrSeparator, optnumMilliseconds, optStrFormat, optnumInterval)
- HST2TXTIsRunning()
- PDFCreate(strSourceFile, optStrPdfFile)
- Print()
- RDFileN("tagSelectedFile", strSearchPath, strMask, numChangeDir)

DeleteOlderFiles(strPath, strMask, strDate)


Group	File
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Deletes files that are older than a date matching the configured mask from the configured path.
- **Parameters:**

StrPath	String tag containing the path to the files that will be deleted.
StrMask	String tag containing the mask of the files to be deleted.
StrDate	String tag containing the cut-off date. Any files older than this date will be deleted.

- **Returned Values:** Returns the number of files deleted.
- **Examples:**

Tag Name	Expression
Tag	DeleteOlderFiles("C:\IWS\Application\HST\", "*.hst", "04/12/2002")

 **Note:** You must configure the third parameter (**StrDate**) using the date format specified for the application (such as **MDY** or **DMY**) with the appropriate separator (/ , : , . , and so forth.)

DirCreate(strDirectory, optBooFullPath)

Group	File
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Creates the specified directory.
- **Parameters:**


strDirectory	String tag or value containing 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 Values:**

-1	Invalid parameters
0	Failed to create the directory (e.g. Drive does not exist.)
1	Directory created successfully.

- **Examples:**

Tag Name	Expression
Tag	DirCreate("C:\Studio\Temp") // The Temp folder is created in the C:\Studio path (only if the C:\Studio path already exists).
Tag	DirCreate("C:\Studio\Temp",1) // The C:\Studio\Temp full path is created.

 **Note:**
When this function is executed from a Web Thin Client station, the directory is checked on the server station (but not on the Web Thin Client).

DirDelete(strDirectory, optBooEmptyOnly)

Group	File
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Deletes the specified directory.
- **Parameters:**

strDirectory	String tag or value containing the name and file path of the directory to delete.
optBooEmptyOnly	Optional flag. If this parameter has a value of 1, then the directory is deleted only if it is empty. By default (i.e., the parameter is omitted or has a value of 0), the directory is deleted whether it is empty or not.

- **Returned Values:**

-2	Attempted to delete a non-empty directory when this action is not allowed (optBooEmptyOnly <> 0)
-1	Invalid parameters
0	Failed to delete the directory (e.g. Directory does not exist.)
1	Directory deleted successfully.

- **Examples:**

Tag Name	Expression
Tag	DirDelete("C:\Studio\Temp") // The Temp folder from C:\Studio is deleted.
Tag	DirDelete("C:\Studio\Temp",1) // The Temp folder from C:\Studio is deleted only if it is empty.

 **Note:**

When this function is executed from a Web Thin Client station, the directory is checked on the server station (but not on the Web Thin Client).

 **Tip:**

This function supports wildcard (* and ?).

DirLength (strPath)

Group	File
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the size of a specific directory.
- **Parameters:**

strPath	String tag or value containing the path of the directory that will be checked.
----------------	--

- **Returned Values:**

-2	Directory does not exist.
-1	Invalid parameters
>=0	Size (in bytes) of the files and sub-folders in the directory

- **Examples:**

Tag Name	Expression
Tag	DirLength("C:\Studio") // Returns the size (in bytes) of all files and sub-folders from C:\Studio.

 **Caution:**

This function is synchronous, and it may take some seconds to return the correct value; therefore, it must be used with caution.

 **Note:**

When this function is executed from a Web Thin Client station, the directory is checked on the server station (but not on the Web Thin Client).

DirRename(strPath, strDirectoryFrom, strDirectoryTo)

Group	File
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Renames directories.
- **Parameters:**

strPath	String tag or value containing the path of the directory that will be renamed.
strDirectoryFrom	String tag or value containing the original name of the directory that will be renamed.
strDirectoryTo	String tag or value containing the target name used to rename the original directory.

- **Returned Values:**

-1	Invalid parameters
0	Failed to rename the directory (e.g. strDirectoryFrom does not exist.)
1	Directory renamed successfully.

- **Examples:**

Tag Name	Expression
Tag	DirRename("C:\Studio\","Temp", "New") // The Temp folder from the C:\Studio\ folder is renamed to New.

 **Note:**

When this function is executed from a Web Thin Client station, the directory is renamed on the server station (but not on the Web Thin Client).

 **Tip:**

This function supports wildcard (* and ?).

FileCopy (strSourceFile, strTargetFile, optnumTimeOut)

Group	File
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Copies the file(s) configured in the **strSourceFile** parameter to the path/file configured in the **strTargetFile** parameter.
- **Parameters:**

strSourceFile	String tag containing the path and mask of the file(s) to be copied.
strTargetFile	String tag containing the path where the file(s) are to be copied.
optnumTimeOut	Numerical tag containing an integer to set the timeout time for the operation.

- **Returned Values:**

-1	Timeout time expired.
0	Failed to copy file(s).
1	File(s) copied successfully.

- **Examples:**

Tag Name	Expression
Tag	FileCopy("C:\IWS\Application\HST*.hst", "C:\Temp\Hst\", 1000)
Tag	FileCopy("C:\IWS\Application\ropert.txt", "C:\Temp\Tuesday_Report.txt", 500)

 **Caution:**

This function is originally synchronous (for example, the scan does not go on until the function finishes the copying procedure). Consequently, using this function for slow network connections may cause problems.

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.

FileDelete (strFile)

Group	File
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Deletes the specified file.
- **Parameters:**

strFile	String tag containing the name and file path of the file to delete.
----------------	---

- **Returned Values:**

0	Failed to delete file
Real	Returns the size of the file deleted

- **Examples:**

Tag Name	Expression
Tag	FileDelete("C:\ IWS\readme.txt")

FileLength (strFile)

Group	File
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the size of a file.
- **Parameters:**

strFile	String tag containing the name and file path of the file.
----------------	---

- **Returned Values:** Returns the size of the specified file in bytes.
- **Examples:**

Tag Name	Expression
Tag	FileLength("C:\readme.txt")

FileRename (strOldName, strNewName)

Group	File
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Renames the specified file.
- **Parameters:**

strOldName	String tag containing the path and old name of the file.
strNewName	String tag containing the path and new name of the file.

- **Returned Values:** No returned values.
- **Examples:**

Tag Name	Expression
	FileRename("C:\readme.txt", "C:\pleasereadme.txt")

FileWrite(strFileName, strWriteText, numoptAppend)

Group	File
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Writes an ASCII string into a specified file. If the file does not exist, the function will create the file.

- **Parameters:**

strFileName	String tag containing the file name.
strWriteText	String tag containing text to be written to the specified file.
numoptAppend	Number tag: <ul style="list-style-type: none"> ▪ If you omit this parameter or specify zero (0), the function replaces the contents of the existing file with the text specified for strWriteText. ▪ If you specify a value other than zero (1, 2, 3, ...), the function adds the new text specified using strWriteText as a new line to the file.

- **Returned Values:**

- 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	FileWrite("c:\test.txt", "This is a test")
Tag	FileWrite(TagFileName, TagText)
Tag	FileWrite(TagFileName, TagText, 1)

FindFile(strFile, "opttagFilesFound", optnumTimeOut)

Group	File
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Searches a file
- **Parameters:**

strFile	String tag contains the file mask for which to search.
"opttagFilesFound"	<i>Optional</i> string tag array receives path and name of each file found.
optnumTimeOut	<i>Optional</i> numerical tag contains an integer to set the timeout in milliseconds for this function.

- **Returned Values:**

-1	Timeout
0	No files found
N	Number of files found

- **Examples:**

Tag Name	Expression
Tag	FindFile("*.txt")
Tag	FindFile("*.doc", "StringArray", 1000)

 **Caution:**

This function is synchronous originally (for example, the scan does not go on until the function returns a value). If you have a slow network connection, this function may cause application problems.

If you use the **optnumTimeOut** parameter, the function returns the value **-1** after the specified timeout time and the scan continues.

FindPath (strPathName)

Group	File
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Verifies whether a directory exists.
- **Parameters:**

strPathName	String tag containing the file path for which to search.
--------------------	--

- **Returned Values:**

0	Path not found
1	Path found

- **Examples:**

Tag Name	Expression
Tag	FindPath("C:\WINNT\")

GetFileAttributes (strFile)

Group	File
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Reads the attributes of a specified file.
- **Parameters:**

strFile	String tag, containing the file from which to read the attributes.
----------------	--

- **Returned Values:**

-1	Error
1	Read only
2	Hidden
4	System
16	Directory
32	Archive
128	Normal
256	Temporary

- **Examples:**

Tag Name	Expression
Tag	GetFileAttributes("C:\readme.txt")

GetFileTime(strFileName, numFormat)

Group	File
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Reads the time and date the file was last modified.
- **Parameters:**

strFileName	String tag containing the file name to be read.
numFormat	Numerical tag, which specifies the format of the returned data. <ul style="list-style-type: none"> • 0: Returns the date and time from the file. • 1: Returns only the file date. • 2: Returns only the file time.

- **Returned Values:** Returns the date and or time the file was last modified.
- **Examples:**

Tag Name	Expression
Tag	GetFileTime("C:\readme.txt")

GetHstInfo(strFileName, numInfoType)

Group	File
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the Start Time, End Time, and Duration of the specified history (*.HST) file.
- **Parameters:**

strFileName	String tag containing the name of the history file to be read.
numInfoType	Numerical tag, which specifies the type of information to be returned. <ul style="list-style-type: none"> • 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 Values:** 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 application error; please contact Technical Support.
-4	The Studio TCP/IP server returned a Time that is incompatible with the format specified in the application screen or Web page. Please use the Verify Application command to update the application and try again.

- **Examples:**

Tag Name	Expression
Tag	GetHstInfo("batch", 0)
Tag	GetHstInfo("hst/02060801.hst", 1)
Tag	GetHstInfo("C:\batch.bat", 2)

GetLine(strFileName, Search, "tagStore", optnumCase, optOverflowTag, optRunFromServer)

Group	File
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Searches a sequence of characters (string) or for a specific line in an ASCII file and stores (in a string tag) the contents of the whole line in a string tag.
- **Parameters:**

strFileName	String tag containing the path and name of the file to be searched. The name can be a fully qualified name (e.g.: C:\File.txt) or simply a file name (e.g.: File.txt). In the second case the application will search for the file in the following paths: Local Station: The application will search for the file in the Application Folder, and if it is not found, in the Application\Web folder. Web Thin Client: If the parameter <code>optRunFromServer</code> 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 applications, we recommend setting the <code>optRunFromServer</code> parameter to 1 and placing your files in the Application\Web folder.
Search	There are two options for this parameter, based on its data type: Numeric Tag/Constant: Number of the line pointed by the function. The function will try to find the line specified in this parameter and copy its contents to the "tagStore" parameter. The first line in the file is line 0. String Tag/Constant: Sequence of characters for which to search. The function will try to find a line in the file that contains this sequence of characters and copy the contents of the first occurrence to the "tagStore" parameter.
"tagStore"	Name of the string tag receiving the contents of the line pointed by the function.
optnumCase	<i>Optional</i> numerical tag specifying whether the search is case-sensitive. <ul style="list-style-type: none"> • 0: Not case-sensitive • 1: Case-sensitive
optOverflowTag	<i>Optional</i> name of a numerical tag receiving the result of overflow verification. <ul style="list-style-type: none"> • 0: OK • 1: Overflow

OptOverflowTag	<p><i>Optional</i> numerical tag ignored when the function is called on local stations. On Web Thin Clients this parameter indicates the following:</p> <ul style="list-style-type: none"> • 0: Retrieves the file from the Web 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 the function will ignore the application path and search for the file in the application URL where the screen files are located.
-----------------------	--

▪ **Returned Values:**

N	Amount of lines in which the sequence was found in the target ASCII file.
0	String was not found in the target ASCII file
-1	ASCII File was 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
-7	Invalid number of parameters
-8	Invalid line number

▪ **Examples:**

Tag Name	Expression
Tag	<code>GetLine("C:\TechRef v51.doc", "Studio version", "ReturnLine")</code>
Tag	<code>GetLine("C:\readme.txt", 1, "ReturnedLine", 0, "Overflow")</code> // returns the second line of the file

 **Note:**

This function only searches for text in ASCII files.

HST2TXT(strStartDate, strStartTime, numDuration, numGroup, strTargetFile, optStrSeparator, optnumMilliseconds, optStrFormat, optnumInterval)

Group	File
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Convert information from the Trend history file(s) in proprietary binary format (*.hst) to a plain text (*.txt) or comma-delimited (*.csv) file.
- **Parameters:**

strStartDate	String tag containing the start date of the data.
strStartTime	String tag containing 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. If omitted, the function creates a file with the same name of the proprietary binary file, but using the extension TXT.
optStrSeparator	<i>Optional</i> string tag containing the data separator character for file. If omitted, the function uses the TAB character (\t) to separate the values in the text file.
optnumMilliseconds	<i>Optional</i> 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	<i>Optional</i> numeric tag specifying the sampling interval. Only line items at this interval are written to the TXT 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 Values:

-3	Invalid number of parameters
-2	Dll functions not found
-1	IndHst.dll not found
0	Function was executed successfully
1	Error. Previous execution of the HST2TXT has not yet been completed

- Examples:

Tag Name	Expression
Tag	HST2TXT("04/14/2002", "06:30:00", 0.1, 3, "C:\Studio\data.txt", "\")")
Tag	HST2TXT("04/14/2002", "06:30:00", 0.1, 3, "C:\Studio\data.csv", "\", " , \"MDY\")")
Tag	HST2TXT("04/14/2002", "06:30:00", 0.1, 3, "C:\Studio\data.csv", "\", " , \"MDY\", 10)")

⇒ **Tip:**

When using the comma (,) as **optStrSeparator**, the function creates a file in the CSV format (Comma Separated Values). It is useful for exporting the Trend history data into a file that can be opened with Microsoft Excel.

HST2TXTIsRunning ()

Group	File
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Returns the status of the **HST2TXT ()** function.
- **Returned Values:**

0	HST2TXT 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 HST file.
-5	Cannot create/open ASCII file.
-6	Cannot read file information from HST file
-7	Invalid file type.
-8	Cannot read header information from HST file.
-9	Invalid number of tag in the header information ($0 > nTags > 250$)
-10	Cannot create Header file (.hdr)
-20	IndHst . dll was not found.
-30	Cannot access dll function.

- **Examples:**


Tag Name	Expression
Tag	HST2TXTIsRunning()

PDFCreate(strSourceFile, optStrPdfFile)

Group	File
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported


- **Description:** Creates a PDF file from the specified source file.
- **Parameters:**

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 application folder.
optStrPdfFile	<i>Optional</i> 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 -> \<path>\MyDocument.pdf

 **Note:**
When entering the file name without a path, a leading backslash ("\") is optional.

- **Returned Values:**

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

 **Caution:**
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
	PDFCreate("C:\Report1.rtf")
	PDFCreate("C:\Report2.doc", "C:\Converted1.pdf")
	PDFCreate("C:\Report3.txt", "C:\Data\Converted1.pdf")

Print (strFilePath , optnumOrientation)

Group	File
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Prints a text file.
- **Parameters:**

strFilePath	Path and name of the text file that will be printed.
optnumOrientation	Set the paper orientation as follows: 0 (default) = Portrait 1 = Landscape

▪ **Note:**
The **optnumOrientation** parameter is not supported when running the application under the Windows CE operating system.

- **Returned Values:** No returned values.
- **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("tagSelectedFile", strSearchPath, strMask, optNumChangeDir)

Group	File
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Launches a *File Browser* window allowing you to select a file.
- **Parameters:**

"tagSelectedFile"	Name of the string tag receiving the name and path of a selected file. The tag name must be configured between quotes. Moreover, it must be a valid tag name — it cannot be a VBScript variable name, for example.
strSearchPath	String tag containing the file path (directory) to search.
strMask	String tag containing the mask used to filter the files.
optNumChangeDir	<i>Optional</i> numerical 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 Values:**

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	RDFileN("FileName", "C:\IWS\","*.doc", 1)

Graphic Functions

This section describes the following InduSoft Web Studio Graphic functions:

- `AutoFormat(numValue)`
- `GetScrInfo(strScreenName, "tagResult", optNumResultType, optNumID)`
- `PrintSetup()`
- `PrintWindow(strScreenName , optnumOrientation, optNumID, optStrMnemonicList)`
- `ResetDecimalPointsTable()`
- `RGBColor(numRed, numGreen, numBlue)`
- `RGBComponent (numColor , numComponent)`
- `SetDecimalPoints(numBaseValue , numDecimalPoints)`
- `SetDisplayUnit(strUnitOrigin, strDisplayUnit, numDiv, numAdd)`
- `SetTagDisplayUnit(strTagName, strDisplayUnit, numDiv, numAdd)`

AutoFormat (numValue)

Group	Graphic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** 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.)

- **Parameters:**

numValue	A numeric value, or a tag of Real type; the real number to be formatted.
-----------------	--

- **Returned Values:** This function returns a formatted string.
- **Examples:** In the following examples, the **SetDecimalPoints()** function has already been used to set three decimal places for values greater than equal to 1.5 and one 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(strScreenName, "tagResult", optNumResultType, optNumID)

Group	Graphic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Retrieves information from the application about an open screen.
- **Parameters:**
 - **strScreenName:** A string or a tag of String type; the name of the screen for which information is required.
 - **tagResult:** A string enclosed by quotes; the name of the tag that will receive the information retrieved by the function.
 - **optStrInfoType:** A numeric value or tag; the type of information to be retrieved by the function. This is an optional parameter; the default value is 0.

Value	Description
0	Writes the following coordinates to each consecutive position of the array tag configured in the tagResult parameter: TOP, LEFT, BOTTOM and RIGHT.
1	Writes the TOP coordinate to the tag configured in the tagResult parameter.
2	Writes the LEFT coordinate to the tag configured in the tagResult parameter.
3	Writes the BOTTOM coordinate to the tag configured in the tagResult parameter.
4	Writes the RIGHT coordinate to the tag configured in the tagResult parameter.

- **optNumID:** A numeric value or tag; 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 Values:**

Value	Description
0	Success
-1	The first and second parameters are not strings.
-2	Memory allocation error
-3	optStrInfoType is zero, but the tagResult parameter is not an array tag.
-4	Invalid tag configured in the tagResult parameter.


- **Examples:**

Tag Name	Expression
TagErrorCode	<pre>GetScrInfo("main" , "TagXY[0]") // 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.</pre>

Tag Name	Expression
TagErrorCode	GetScrInfo("main", "TagXY", 3) // Retrieves the BOTTOM coordinate of the "main" screen and then writes it to TagXY.
TagErrorCode	GetScrInfo("main" , "TagXY", 2, 10) // Retrieves the LEFT coordinate of the "main" screen with ID 10 and then writes it to TagXY.

PrintSetup ()

Group	Graphic
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

 **Note:**
This function cannot be used with Tasks (see page 8–1) or in a VBScript sample in the Global Procedures interface (see page 15–8).

- **Description:** Opens the standard setup dialog from the operating system, where the printer can be selected and configured.
- **Returned Values:** No returned values.
- **Examples:**

Tag Name	Expression
Tag	PrintSetup()

PrintWindow(strScreenName , optnumOrientation)

Group	Graphic
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

Note:

This function cannot be used with Tasks (see page 8-1) or in a VBScript sample in the Global Procedures interface (see page 15-8).


- **Description:** Prints a screenshot of an application 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.
- **Parameters:**

strScreenName	A string or a tag of String type; 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 under Windows CE.
optnumOrientation	Set the paper orientation as follows: 0 (default) = Portrait 1 = Landscape NOTE: The optNumOrientation parameter is not supported when running the application under Windows CE.
optNumID	A numeric value or tag; 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 Values:** No returned values.

▪ **Examples:**

Tag Name	Expression
	PrintWindow() // Prints the currently active screen in portrait orientation.
	PrintWindow("Main", 1) // Prints the "Main" screen in landscape orientation.
	PrintWindow(TagScreenName) // Prints the screen specified by TagScreenName.
	PrintWindow("Main", 1, 10) // Prints the "Main" screen with ID 10.
	PrintWindow ("Main", 1, 0, "#Mne1:Tag1 #Mne2:Tag2") // Prints the "Main" screen, replacing the custom properties Mne1 and Mne2 with Tag1 and Tag2, respectively.

 **Note:**
You can use the **PrintWindow()** function to print graphical reports that include Alarm/Event Control and Trend Control objects.

ResetDecimalPointsTable()

Group	Graphic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Resets the virtual table of settings created by the **SetDecimalPoints()** function.
- **Parameters:** There are no parameters for this function.
- **Returned Values:** There are no returned values for this function.
- **Examples:**

Tag Name	Expression
	ResetDecimalPointsTable() // Resets the virtual table of settings.

RGBColor (numRed, numGreen, numBlue)

Group	Graphic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the number of the color defined by the RGB (Red, Green and Blue) codes.
- **Parameters:**

numRed	Red code from the RGB code
numGreen	Green code from the RGB code
numBlue	Blue code from the RGB code

- **Returned Values:** This function returns the number of the color defined by the RGB (Red, Green and Blue) codes.
- **Examples:**

Tag Name	Expression
TagColor	<code>RGBColor(51,153,102)</code> // This function returns the value 13434828, which is the color code for Sea Green.
TagColor	<code>RGBColor(TagRed,TagGreen,TagBlue)</code> // This function returns the color code of the RGB values set in the tags TagRed, TagGreen and TagBlue, respectively.

⇒ **Tip:**

See the list of RGB Codes and Color values for the most used colors in the *IWS Development Environment -> Standard Interfaces -> Color Interface* section.

RGBComponent (numColor , numComponent)

Group	Graphic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the code of one of the RGB components from the given color code.
- **Parameters:**

numColor	Color code from which one component value will be extracted.
numComponent	Specify which RGB component will be extracted from the color (0=Red ; 1=Green; 2=Blue)

- **Returned Values:** This function returns the code of one of the RGB components from the given color code.
- **Examples:**

Tag Name	Expression
TagRed	RGBColor(13434828,0) // This function returns the value 51, which is the Red component of the color code 13434828 (Sea Green).
TagGreen	RGBColor(13434828,1) // This function returns the value 153, which is the Green component of the color code 13434828 (Sea Green).
TagBlue	RGBColor(13434828,2) // This function returns the value 102, which is the Blue component of the color code 13434828 (Sea Green).
TagComponent	RGBColor(TagCode, TagComponent) // This function returns the value of the color code and component specified by the tags TagCode and TagComponent, respectively.

⇒ **Tip:**
See the list of RGB Codes and Color values for the most used colors in the Color Interface chapter.

SetDecimalPoints (numBaseValue , numDecimalPoints)

Group	Graphic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Sets the number of decimal places to be displayed, for a specified range of real numbers. This setting will be used by all objects and dynamic properties that have the **Auto Format** option enabled, as well as by the **AutoFormat ()** function. Furthermore, 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.

- **Parameters:**

numBaseValue	A numeric value, or a tag of Integer or Real type; 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	A numeric value, or a tag of Integer type; the number of decimal places to be displayed, for the range of real numbers specified by numBaseValue .

- **Returned Values:**

Value	Description
0	Error
1	Success

- **Examples:**

Tag Name	Expression
Tag	SetDecimalPoints(1.5, 3) // Displays three decimal places for all real numbers greater than or equal to 1.5.
Tag	SetDecimalPoints(-3, 1) // Displays one decimal place for all real numbers less than or equal to -3.

SetDisplayUnit(strUnitOrigin, strDisplayUnit, numDiv, numAdd)

Group	Graphic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** 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. For more information, see “Setting Tag Properties” on page 5–19.

 **Note:**

This function only affects how the tag values are displayed on screen; it does not change the actual tag values in any way.

- **Parameters:**

strUnitOrigin	A string or a tag of String type; the engineering unit to be matched.
strDisplayUnit	A string or a tag of String type; the new value for the DisplayUnit tag field.
numDiv	A numerical value or tag; the new value for the UnitDiv tag field.
numAdd	A numerical value or tag; the new value for the UnitAdd tag field.

- **Returned Values:**

Value	Description
0	Success.
-1	Wrong number of parameters.
-2	strUnitOrigin parameter is empty.
-3	numDiv parameter is invalid (equal to 0).

- **Examples:**

Tag Name	Expression
Tag	<pre>SetDisplayUnit("C", "F", 0.555556, 32)</pre> <p>// For all tags and object values with a Unit of "C", the DisplayUnit tag field is set to "F", the UnitDiv tag field is set to 0.555556, and the UnitAdd tag field is set to 32.</p>

SetTagDisplayUnit(strTagName, strDisplayUnit, numDiv, numAdd)

Group	Graphic
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Sets the **DisplayUnit**, **UnitDiv**, and **UnitAdd** tag fields on a specific tag. For more information, see “Setting Tag Properties” on page 5–19.

 **Note:**

This function only affects how the tag values are displayed on screen; it does not change the actual tag values in any way.

- **Parameters:**

strTagName	A string or a tag of String type; 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	A string or a tag of String type; the new value for the DisplayUnit tag field.
numDiv	A numerical value or tag; the new value for the UnitDiv tag field.
numAdd	A numerical value or tag; the new value for the UnitAdd tag field.

- **Returned Values:**

Value	Description
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	<pre>SetTagDisplayUnit("TagTemp", "F", 0.555556, 32)</pre> <p>// For the tag “TagTemp”, the DisplayUnit tag field is set to “F”, the UnitDiv tag field is set to 0.555556, and the UnitAdd tag field is set to 32.</p>

Translation Functions

This section describes the following InduSoft Web Studio Translation functions:

- `Ext(strText)`
- `SetTranslationFile(strFileName, optStrColumnName)`

Ext(strText)

Group	Translation
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Translates specified text.
- **Parameters:**

strText	String tag containing the text to be translated.
----------------	--

- **Returned Values:** Returns the text translation using the active translation file.
- **Examples:**

Tag Name	Expression
Tag	<code>Ext("Start")</code> // Returned value if translating to Portuguese = "Iniciar"
Tag	<code>Ext("Stop")</code> // Returned value if translating to German = "Anschlag"

SetTranslationFile(strFileName, optStrColumnName)

Group	Translation
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Sets the active translation file and translates all enabled text within the application.
- **Parameters:**

strFileName	String tag containing the name of a translation file.
optStrColumnName	String tag or value containing the name of the column from the translation file, which must be used to translate the texts on the application. If omitted, then the second column from the translation file will be used by default.

- **Returned Values:**

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	SetTranslationFile("Portuguese.tra")
Tag	SetTranslationFile("German.tra")
Tag	SetTranslationFile("Mytranslation.csv" , "Portuguese")
Tag	SetTranslationFile("Mytranslation.csv" , "German")

 **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 File* utility.

Multimedia Functions

This section describes the InduSoft Web Studio Multimedia function, `Play(strFileName)`.

`Play(strFileName, optNumSynchronous)`


Group	Multimedia
Execution	Synchronous/Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Plays a specified **.WAV** file.
- **Parameters:**

strFileName	String tag containing the .WAV to translate.
optNumSynchronous	Numerical tag that controls whether the function executes synchronously or asynchronously. Where: <ul style="list-style-type: none"> • Specifying 0 (<i>default</i>) or not specifying the parameter enables the function to execute asynchronously. • Specifying 1 enables the function to execute synchronously.

- **Returned Values:** No returned values.
- **Examples:**

Tag Name	Expression
	<code>Play("C:\Sounds\Wav\alarm.wav")</code>

 **Note:**
For this function to work in a Web Thin Client, the target **.WAV** file must be located in the same file path on the remote station.

System Information Functions

This section describes the following InduSoft Web Studio System Information functions:

- DbVersion()
- GetAppHorizontalResolution()
- GetAppPath()
- GetAppVerticalResolution()
- GetComputerIP()
- GetComputerName()
- GetCursorX()
- GetCursorY()
- GetDisplayHorizontalResolution()
- GetDisplayVerticalResolution()
- GetFreeMemoryCE(optnumType)
- GetHardKeyModel()
- GetHardkeySN()
- GetIPAll("tagArrayIP", optRefresh)
- GetMemoryCE(optnumType)
- GetNetMACID()
- GetProductPath()
- GetOS()
- GetPrivateProfileString(strSection, strName, strDefault, strFileName)
- GetRegValue(numMainKey, strKey, strValueName)
- GetRegValueType(numMainKey, strKey, strValueName)
- GetServerHostName
- GetTickCount()
- InfoAppAlrDir()
- InfoAppHstDir()
- InfoDiskFree(strDisk)
- InfoResources(numSelect)
- IsActiveXReg(numType, strProgIDorFileName)
- NoInputTime(optUpdateTrigger)
- ProductVersion()
- RegSaveCE()
- SaveAlarmFile(numType, optRemotePath)
- SetAppAlarmPath(strPath)
- SetAppHSTPath(strPath)
- SetDateFormat(strSeparator, strFomat)
- SetRegValue(numMainKey, strKey, strValueName, numType, strOrNumValue)
- SetWebConfig(strServerIP, optStrBackupURL, optStrPath, optNumHostPort, optStrSecondaryServerIP, optNumProtocolFlag,

optNumGtwPort, optStrGtwIP, optStrSecondaryGtwIP,
optStrISSymbolURL)

- SNMPGet(strAddress, strCommunity, strOID, strTagName)
- WritePrivateProfileString(strSection, strName, strValue, strFileName)

DbVersion ()

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the database version number of the current application.
- **Returned Values:** Returns the database version in numerical format.
- **Examples:**

Tag Name	Expression
Tag	DbVersion()

GetAppHorizontalResolution ()

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Obtains the horizontal screen resolution from the **.APP** file, section [Info].
- **Returned Values:** Returns the [Info] value, but does not test the Windows configuration.
- **Examples:**


Tag Name	Expression
Tag	GetAppHorizontalResolution() // Returned value = 640
Tag	GetAppHorizontalResolution() // Returned value = 800

GetAppPath ()

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Returns the directory of the current application.
- **Returned Values:** Returns the directory of the current application as a string.
- **Examples:**

Tag Name	Expression
Tag	GetAppPath() // Returned value = C:\DemoApp\
Tag	GetAppPath() // Returned value = C:\Studio\Projects\App\

 **Note:**
This function must return the current path of the application, including the “\” at the end of the path.

GetAppVerticalResolution ()

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Gets the vertical screen resolution from the **.APP** file, section [Info].
- **Returned Values:** Returns the [Info] value, but does not test the Windows configuration.
- **Examples:**

Tag Name	Expression
Tag	GetAppVerticalResolution() // Returned value = 480
Tag	GetAppVerticalResolution() // Returned value = 600

GetComputerIP ()

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the first IP Address of the local station.
- **Returned Values:** Returns the first IP Address of the local station as a string.
- **Examples:**

Tag Name	Expression
Tag	GetComputerIP() // Returned value = 192.168.0.1
Tag	GetComputerIP() // Returned value = 248.12.2.78

GetComputerName ()

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported

- **Description:** Returns the local computer name.
- **Returned Values:** Returns the local computer name as a string.
- **Examples:**

Tag Name	Expression
Tag	GetComputerName() // Returned value = Terminal53
Tag	GetComputerName() // Returned value = BobsComputer

GetCursorX ()

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Gets the X-coordinate of the cursor on the screen.
- **Returned Values:** This function returns the X-coordinate of the cursor on the screen, or -1 if an error occurs.
- **Examples:**

Tag Name	Expression
	GetCursorX() // Returned value = 1024

GetCursorY ()

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Gets the Y-coordinate of the cursor on the screen.
- **Returned Values:** This function returns the Y-coordinate of the cursor on the screen, or -1 if an error occurs.
- **Examples:**

Tag Name	Expression
	GetCursorY() // Returned value = 768

GetDisplayHorizontalResolution()

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Gets the horizontal resolution, in pixels, of the display connected to the local computer.
- **Returned Values:** This function returns the horizontal resolution of the display as an integer.
- **Examples:**

Tag Name	Expression
	<pre>GetDisplayHorizontalResolution() // Returned value = 1024</pre>

GetDisplayVerticalResolution()

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Gets the vertical resolution, in pixels, of the display connected to the local computer.
- **Returned Values:** This function returns the vertical resolution of the display as an integer.
- **Examples:**

Tag Name	Expression
	<pre>GetDisplayVerticalResolution() // Returned value = 768</pre>

GetFreeMemoryCE (optnumType)

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Not Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the free memory available in a Windows CE device.
- **Parameters:**

optnumType	<i>Optional</i> numerical tag that specifies which type of free memory IWS should retrieve from the Windows CE device, where: 0: Total free Program memory
-------------------	--

- **Returned Values:**

>0	Size of free memory in bytes.
-1	Coredll.dll file not found.
-2	GetFreeMemoryCE function not found.
-3	Invalid optional parameter.
-4	Type of memory unavailable.

- **Examples:**

Tag Name	Expression
Tag	GetFreeMemoryCE(opt2)

GetHardKeyModel ()

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Returns the name of your hardkey model.
- **Returned Values:**
 - **Hardkey located:** Returns a string with the hardkey model name.
 - **Hardkey not installed or not found:** No values returned.
- **Examples:**

Tag Name	Expression
Tag	GetHardKeyModel() // Returned value = Local Interface
Tag	GetHardKeyModel() // Returned value = Advanced Server

⚠ Caution:
You must install the hardkey before executing this function or the function will not execute properly.

GetHardkeySN()

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Returns the serial number of the hardkey.
- **Returned Values:**
 - **Hardkey located:** Returns a string with the hardkey serial number.
 - **Hardkey not installed or not found:** Returns a 0.
- **Examples:**

Tag Name	Expression
Tag	GetHardkeySN() // Returned value = 120.745
Tag	GetHardkeySN() // Returned value = 224.941

⚠ Caution:
You must install the hardkey before executing this function or the function will not execute properly.

GetIPAll ("tagArrayIP", optRefresh)

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the number of IP Addresses assigned to the local station and stores the IP Addresses in a string array tag.
- **Parameters:**

"tagArrayIP"	Name of the string array tag receiving the IP addresses found.
optRefresh	<i>Optional</i> tag that triggers a refresh of this function, if you use it in a text I/O dynamic. Every time this tag changes value, IWS will refresh the function.

- **Returned Values:**

N	Number of IP addresses found
-1	Invalid number of parameters
-2	Invalid parameter type

- **Examples:**

Tag Name	Expression
Tag	GetIPAll("TagArrayIP") // Returned value = 1
Tag	GetIPAll("TagArrayIP", second) // Returned value = 2

GetMemoryCE (optnumType)

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Not Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the total memory available in a Windows CE device.
- **Parameters:**

optnumType	<p><i>Optional</i> numerical tag that specifies which type of memory IWS should retrieve from the Windows CE device, as follows.</p> <ul style="list-style-type: none"> • 0: Total Program memory • 1: Total Storage memory • 2: Total memory
-------------------	---

- **Returned Values:**

>0	Size of memory in bytes.
-1	Coredll.dll file not found.
-2	GetMemoryCE function not found.
-3	Invalid optional parameter.

- **Examples:**

Tag Name	Expression
Tag	GetMemoryCE(opt1)

GetNetMACID("optStrMACID", "optStrAdapterName")

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Gets the MAC ID unique code from the currently installed network adapter(s).
- **Parameters:**

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 Values:**

>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 optStrMACID does not exist.
-4	Tag configured in optStrAdapterName does not exist.

- **Examples:**

Tag Name	Expression
NumNIC	GetNetMACID("MACIDTag")
NumNIC	GetNetMACID("MACIDTag", "AdapterName")
NumNIC	GetNetMACID("MACIDTag[1]", "AdapterName[1]")

GetProductPath ()

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the path to the IWS directory.
- **Returned Values:** Returns the path to the IWS directory as a string.
- **Examples:**

Tag Name	Expression
Tag	GetProductPath() // Returned value = C:\Program Files\ IWS 51\

GetOS ()

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Reports the current operating system.
- **Returned Values:**

0	Windows 3.11
1	Windows 95/98/ME
2	Windows 2K/XP/Vista
3	Windows CE

- **Examples:**

Tag Name	Expression
Tag	GetOS() //Returned value = 2

GetPrivateProfileString(strSection, strName, strDefault, strFileName)

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Reads a specified parameter from an **.INI** file using the standard **.INI** format.
- **Parameters:**

strSection	String tag containing the section name to be read.
strName	String tag containing the parameter name to be read.
strDefault	String tag containing the default setting for this parameter. If the parameter is not found in the .ini file, the function will return this default setting.
StrFileName	String tag containing the path and name of the .ini file to be read.

- **Returned Values:** Returns the value of the specified parameter.
- **Examples:**

Tag Name	Expression
Tag	GetPrivateProfileString("boot loader", "timeout", "50", "C:\boot.ini") // Returned value = 30

GetRegValue(numMainKey, strKey, strValueName)

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Not Supported

- **Description:** Retrieves a variable value in the registry.
- **Parameters:**

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.
strValueName	Name of the variable to get. The maximum length is 255 characters.

- **Returned Values:** 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	GetRegValue(0, "HARDWARE\DESCRIPTION\System", "SystemBiosDate") // Returned value = "08/14/03"
Tag	GetRegValue(2, "Control Panel\Current", "Color Schemes") // Returned value = "Windows Standard "

GetRegValueType (numMainKey, strKey, strValueName)

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Not Supported

- **Description:** Retrieves the type of a variable value in the registry.
- **Parameters:**

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.
strValueName	Name of the variable to get. The maximum length is 255 characters.

- **Returned Values:** If the function succeeds, then it will return 0 for DWord, 1 for String. Otherwise one of the following 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	GetRegValueType (0, "HARDWARE\DESCRIPTION\System", "SystemBiosDate") // Returned value = 1
Tag	GetRegValueType (2, "Control Panel\Desktop", "Smooth Scroll") // Returned value = 0

GetServerHostName

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Not Supported
Windows CE	Not Supported
Web Thin Client	Supported

- **Description:**
- **Parameters:** None
- **Returned Values:** Server host name for ISSymbol and 127.0.0.1 for others.
- **Examples:**

Tag Name	Expression

GetTickCount ()

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the current value of the clock ticks counter.
- **Returned Values:** Returns an integer with the number of milliseconds counted by the clock for each initialization of the operational system.
- **Examples:**

Tag Name	Expression
Tag	GetTickCount // Returned value = 9400907

InfoAppAlrDir ()

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the *Alarm* directory for the current application.
- **Returned Values:** Returns the *Alarm* directory of the current application as a string.
- **Examples:**

Tag Name	Expression
Tag	InfoAppAlrDir() // Returned value = C:\DemoApp\alarm\
Tag	InfoAppAlrDir() // Returned value = C:\IWS\Projects\App\alarm\

InfoAppHstDir ()

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the *History* directory for the current application.
- **Returned Values:** Returns the *History* directory for the current application as a string.
- **Examples:**

Tag Name	Expression
Tag	InfoAppAlrDir() // Returned value = C:\DemoApp\HST\
Tag	InfoAppAlrDir() // Returned value = C:\IWS\Projects\App\HST\

InfoDiskFree (strDisk)

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported

- **Description:** Returns disposable free space on the disk.
- **Parameters:**

strDisk	String tag containing the name of the disk to be checked.
----------------	---

- **Returned Values:** Returns disposable free space in the disk in bytes.
- **Examples:**

Tag Name	Expression
Tag	InfoDiskFree("C") // Returned value = 2803804605.000000

InfoResources (numSelect)


Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the Window's disposable resources.
- **Parameters:**

numSelect	Integer tag containing the resource to examine. <ul style="list-style-type: none"> • 0: System functions (%) • 1: GDI functions (%) • 2: USER functions (%) • 3: Memory (in bytes)
------------------	--

- **Examples:**

Tag Name	Expression
Tag	InfoResources(0) // Returned value = 76.000000
Tag	InfoResources(1) // Returned value = 76.000000
Tag	InfoResources(2) // Returned value = 80.000000
Tag	InfoResources(3) // Returned value = 16150528.000000

 **Note:**
The only valid selection on a Windows 2K/XP/Vista station is **3**. Selecting **0-2** returns **0.000000** only.

IsActiveXReg (numType, strProgIDorFileName)

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Determines whether an ActiveX control is registered.
- **Parameters:**

NumType	Numerical tag specifying a format for the <code>strProgIDorFileName</code> tag. <ul style="list-style-type: none"> • 0: Verify by Program ID • 1: Verify by File Name
strProgIDorFileName	String tag containing Program ID or File Name of the ActiveX Control.

- **Returned Values:**

0	ActiveX is not registered.
1	ActiveX is registered.

- **Examples:**

Tag Name	Expression
Tag	<code>IsActiveXReg(0, "ISSYMBOL.ISSymbolCtrl.1")</code> // Returned value = 0
Tag	<code>IsActiveXReg(1, "C:\winNT\system32\MediaPlayer.ocx")</code> // Returned value = 1

NoInputTime (optUpdateTrigger)


Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported (for keyboard input only)

- **Description:** Returns the time elapsed since the last keyboard action.
- **Parameters:**

optUpdateTrigger	<i>Optional</i> tag that triggers an update when this function is used in a text I/O property. Every time this tag's value changes, IWS triggers the function.
-------------------------	--

- **Returned Values:** 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 I/O object.

ProductVersion ()

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the IWS version number.
- **Returned Values:**

0	Success
1	Error

RegSaveCE ()

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Not Supported
Windows CE	Supported
Web Thin Client	Not Supported

- **Description:** Saves the Windows CE Registry. This function will only work if the save registry capability is enabled in the Windows CE image.
- **Returned Values:** If the function succeeds and the registry is saved, then the returned value is 0. Otherwise one of the following error codes is returned:

-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 CE devices.

SaveAlarmFile (numType, optRemotePath)

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Not Supported

- **Description:** 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.
- **Parameters:**

NumType	Tag containing the number and operation, as follows: <ul style="list-style-type: none"> ▪ 0 – Disable save the alarm file to the local disk. ▪ 1 – Enable save the alarm file to local disk. ▪ 2 – Enable save the alarm file to local disk and to the remote path specified in the OptRemotePath parameter.
OptRemotePath	Tag containing the name of the remote computer where the alarm file will be saved simultaneously to the local computer and to the remote path when numType = 2 .

- **Returned Values:**

0	Success
1	Second parameter is not a string.
2	Second parameter is missing.

- **Examples:**

Tag Name	Expression
Tag	SaveAlarmFile(0)
Tag	SaveAlarmFile(1)
Tag	SaveAlarmFile(2, "Z:\Apps\AppDemo")

SetAppAlarmPath (strPath)

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Sets the *Alarm* path for the current application.
- **Parameters:**

strPath	String tag containing the new <i>Alarm</i> path for the current application.
----------------	--

- **Returned Values:** No returned values.
- **Examples:**

Tag Name	Expression
	SetAppAlarmPath("C:\IWS\Alarm\")

SetAppHSTPath (strPath)

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Sets the file path (directory) where Trend history files will be saved, in the proprietary format (.HST). 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.

- **Parameters:**

strPath	String tag containing the file path (directory) where Trend history files will be saved.
----------------	--

- **Returned Values:** No returned values.
- **Examples:**

Tag Name	Expression
	SetAppHstPath("C:\IWS\History\")

SetDateFormat(strSeparator, strFormat)

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Sets the format and separator for the date string.
- **Parameters:**

strSeparator	String tag containing 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 Values:**

0	No error
1	Invalid parameter

- **Examples:**

Tag Name	Expression
Tag	SetDateFormat("/", "MDY") // Date = 04/18/2002
Tag	SetDateFormat(".", "MYD") // Date = 04:2002:18

SetRegValue(numMainKey, strKey, strValueName, numType, strOrNumValue)

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Not Supported

- **Description:** Sets a variable value in the registry.
- **Parameters:**

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 variable is located in the Main Key.
strValueName	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 Values:** If the function succeeds and the variable is set, then the returned value is 0. Otherwise one of the following error codes is returned:

-1	Invalid number of parameters or invalid Main Key
-2	Invalid Type
-3	Failed to set the variable value, check if you have the proper security rights

- **Examples:**

Tag Name	Expression
Tag	SetRegValue(0, "HARDWARE\DEVICEMAP\SERIALCOMM", "\Device\Serial1", 1, "COM3") // Returned value = 0 if successful
Tag	SetRegValue(2, "Control Panel\Desktop", "Smooth

	<pre>Scroll", 0, 1) // Returned value = 0 if successful</pre>
--	--

```
SetWebConfig( strServerIP, optStrBackupURL, optStrPathFile,
optNumHostPort, optStrSecondaryServerIP,
optNumProtocolFlag, optNumGtwPort, optStrGtwIP,
optStrSecondaryGtwIP, optStrISSymbolURL )
```

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Configures the Web settings for the current application. Configures the secondary data server and secondary URL settings for the Web solution. The settings configured in the function are updated on the HTML files of the application.
- **Parameters:**

strServerIP	Data Server IP Address. IP address (or hostname) of the computer where the TCP Server module of IWS is running.
optStrBackupURL	<i>Optional</i> alternative URL for the application Web pages. The Web Thin Client will look for the Web pages in this URL if it does not find them in the same URL written in the Address field of the Web browser.
optStrPathFile	<i>Optional</i> file path and name of the HTML file to be updated. If you specify only the file path without a file name, then all of the HTML files in the specified file path will be updated.
optNumHostPort	<i>Optional</i> TCP Port number that the Web Thin Client must use to exchange data with the TCP Server module of IWS.
optStrSecondaryServerIP	<i>Optional</i> alternative data server IP address. The Web Thin Client will attempt to connect to the TCP Server module of IWS in this IP Address if it is not able to connect to the TCP Server module running in the IP Address specified in the strServerIP parameter.
optNumProtocolFlag	<i>Optional</i> - When you use the Web Tunneling Gateway option, this parameter specifies whether the Web Thin Client will use either HTTP to exchange data with the Web Server or HTTPS (SSL – Secure Socket Layer). If this flag has the value 0, the Web Thin Client will use HTTP. If this flag has the value 1, the Web Thin Client will use HTTPS (SSL).
optNumGtwPort	<i>Optional</i> TCP Port number that the Web Thin Client must use to exchange data with the Web Server when using the Web Tunneling Gateway.
optStrGtwIP	<i>Optional</i> IP Address (or hostname) of the computer where the Web Tunneling Gateway is running.

optStrSecondaryGtwIP	<i>Optional</i> Alternative IP Address (or hostname) of the computer where the Web Tunneling Gateway is running. The Web 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	<i>Optional</i> URL from where the updated version of ISSymbol (ActiveX control) must be downloaded if it is not properly registered in the Web Thin Client station.

▪ **Returned Values:**

0	No error
1	Invalid number of parameters
2	Invalid Server IP address
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/")



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 the runtime, according to the network settings of each project (IP address, Web Server URL, and so forth).
- Only the first parameter of this function is mandatory (strServerIP). The other parameters are optional. The parameters that are not configured in the function assume the default value configured in the **Project** → **Settings** → **Web** window of the development environment.
- The following parameters must be omitted, unless you intend to use the Web Tunneling Gateway: optNumProtocolFlag, optNumGtwPort, optStrGtwIP, optStrSecondaryGtwIP, optStrISSymbolURL.

SNMPGet(strAddress, strCommunity, strOID, strTagName)

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported

- **Description:** Retrieve information from computers or network devices through the SNMP protocol.

- **Parameters:**

strAddress	The address of the machine/computer. Ex: "127.0.0.1" or "localhost"
strCommunity	SNMP community name used when communicating with the computer. Ex: "public"
strOID	OID to be consulted. Ex: ".1.3.6.1.2.1.1.1.0"
strTagName	Name of the tag that will receive the requested value.

- **Returned Values:**

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	SNMPGet("127.0.0.1", "public", ".1.3.6.1.2.1.1.1.0", "SysDescrTag") // 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.

WritePrivateProfileString(strSelection, strName, strValue, strFileName)

Group	System Info.
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Writes a specified parameter to an **.ini** file using the standard **.ini** format.

 **Note:**

When running on Windows CE this function will rewrite the entire file, therefore its use is not recommended for lengthy files on Windows CE devices. The function will also add the following lines at the end of the file when on a Windows CE device:

```
[FileBackUpControl]
Valid=1
```

- **Parameters:**

strSelection	String tag containing the section name to be written.
strName	String tag containing the parameter name to be written.
strValue	String tag containing the value to be written.
strFileName	String tag containing the path and name of the .ini file to be written.

- **Returned Values:** The function returns 1 if the file was updated successfully.
- **Examples:**

Tag Name	Expression
Tag	WritePrivateProfileString(Section, Name, Value, FileName)
Tag	WritePrivateProfileString("Options", "ds1", "Value", "C:\viewer.ini")

Tags Database Functions

This section describes the InduSoft Web Studio Tags Database functions:

- `ExecuteAlarmAck(strTagName, optStrComment, optStrAlarmType)`
- `ForceTagChange(strTagName, numValue)`
- `GetTagValue(strTagName, optNumRefresh)`
- `SetTagValue(strTagName, tagValue)`

ExecuteAlarmAck (strTagName, optStrComment, optStrAlarmType)

Group	Tags Database
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Acknowledge an active alarm on the specified tag. The advantage of using this function is that if used from the Web Thin Client, the Alarm task will store the user name and station from which the alarm was acknowledged.
- **Parameters:**

strTagName	Name of the tag on which the alarm will be acknowledged.
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 Values:**

0	Successfully executed.
-1	Invalid number of parameters.
-2	Invalid tag name.

- Examples:

Tag Name	Expression
	ExecuteAlarmAck ("a", "Hi alarm in tag a", "Hi") // Returned value = 0 if successfully executed

ForceTagChange(strTagName, numValue)

Group	Tags Database
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- Description:** 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.

- Parameters:

strTagName	String tag containing the name of the target tag being forced to accept the new value.
numValue	Tag containing the new value to be written to the target tag.

- Returned Values:** No returned values.

- Examples:

Tag Name	Expression
	ForceTagChange("TagA", 5)


GetTagValue(*strTagName*, *optNumRefresh*)

Group	Tags Database
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Gets the value of the specified Tag from the application database.
- **Parameters:**

strTagName	String (or Tag of String type) containing the name of the Tag of which you want to get the value.
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 same way, such as when a <i>Pushbutton object</i> is clicked.) To execute the function at a regular interval, you can use one of application's <i>system tags</i> such as <i>Second</i> , <i>Minute</i> or <i>Hour</i> .

- **Returned Values:** 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	GetTagName("requiredTag", Second) // Return = 15
pointerTag = "Required Tag" requiredTag = 15 optNum = Second	GetTagName(pointerTag, optNum)

SetTagValue(*strTagName*, *tagValue*)

Group	Tags Database
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Sets the value of the specified Tag in the application database.
- **Parameters:**

strTagName	String (or Tag of String type) containing the name of the Tag that you want to set.
tagValue	Parameter containing the new value to be set to the specified Tag.

- **Returned Values:**

0	No error
-1	Invalid tag name

- **Examples:**

Tag Name	Expression
TagA	SetTagValue("TagA", "Hello") // Return = Hello
TagA	SetTagValue("TagA", 123) // Return = 123
TagA TagB = 15	SetTagValue("TagA", TagB) // Return = 15

Loop Function

This section describes the InduSoft Web Studio Loop function.

For(numInitialValue, numFinalValue, numStep) & Next

Group.	Loop
Execution	N/A
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	N/A

- **Description:** Implements a **For** loop within a script. The section of the script affected by the **For** loop begins with the **For ()** function and ends with the **Next** notation. The **Next** notation directs the script to the previous **For ()** function.

- **Parameters:**

numInitialValue	Numerical tag containing the initial step (increment) of the For loop.
numFinalValue	Numerical tag containing the final step (increment) of the For loop.
NumStep	Numerical tag containing the step (increment) of the For loop.

- **Returned Values:** Returns the step on which the loop is currently running.
- **Examples:**

Tag Name	Expression
Tag	For(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

This section describes the following InduSoft Web Studio ODBC functions:

- ODBCBeginTrans (numHandler)
- ODBCBindCol (numHandler, strColName, strColType, strTagName)
- ODDBCCanAppend (numHandler)
- ODDBCCanTransact (numHandler)
- ODDBCCanUpdate (numHandler)
- ODDBCclose (numHandler)
- ODDBCCommitTrans (numHandler)
- ODBCDelete (numHandler)
- ODBCExecuteSQL (numHandler, strSqlCommand)
- ODBCInsert (numHandler)
- ODBCIsBOF (numHandler)
- ODBCIsDeleted (numHandler)
- ODBCIsEOF (numHandler)
- ODBCIsFieldNULL (numHandler, strColName)
- ODBCIsFieldNullable (numHandler, strColName)
- ODBCMove (numHandler, numOffset)
- ODBCMoveFirst (numHandler)
- ODBCMoveLast (numHandler)
- ODBCMoveNext (numHandler)
- ODBCMovePrev (numHandler)
- ODBCOpen (strDsn, strUser, strPassw, strTable, strFilter, strSort)
- ODBCQuery (numHandler)
- ODBCROLLback (numHandler)
- ODBCSetFieldNull (numHandler, strColName, numValue)
- ODBCSetFilter (numHandler, strFilter)
- ODBCSetSort (numHandler, strSort)
- ODBCUnbindCol (numHandler, strColName)
- ODBCUpdate (numHandler)

ODBCBeginTrans (numHandler)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Begins a transaction with the connected data source.
- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
-------------------	--

- **Returned Values:**

0	Success
1	Invalid handler
2	Database not open
3	Error beginning transaction

- **Examples:**

Tag Name	Expression
Tag	ODBCBeginTrans(5)

ODBCBindCol (numHandler, strColName, strColType, strTagName)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Binds a column to a tag. Every time you finish binding columns, you must call the **ODBCQuery** function.
- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
strColName	String tag containing the Database column name.
strColType	String tag containing the SQL data type (one of the following):
	<ul style="list-style-type: none"> • 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	String tag containing the name of the tag to bind to the column.

- **Returned Values:**

0	Success
1	Invalid Handler
2	Invalid parameter type
3	One of the parameters has an empty string
4	ColType contains an invalid type

- **Examples:**

Tag Name	Expression
Tag	ODBCBindCol(5, "OrderDate", "SQL_DATE", "Order_Date")

ODBCCanAppend (numHandler)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Returns whether the database will allow you to add new records.
- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
-------------------	--

- **Returned Values:**

0	Database does not allow appending new records
Non-Zero	Database does allow appending new records

- **Examples:**

Tag Name	Expression
Tag	ODBCCanAppend(5)

ODBCCanTransact (numHandler)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Returns whether the database allows transactions.
- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
-------------------	--

- **Returned Values:**

0	Database does not allow transactions.
Non-Zero	Database does allow transactions.

- **Examples:**

Tag Name	Expression
Tag	ODBCCanTransact(2)

ODBCCanUpdate (numHandler)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Returns whether the database can be updated.
- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
-------------------	--

- **Returned Values:**

0	Database does not allow updates.
Non-Zero	Database does allow updates.

- **Examples:**

Tag Name	Expression
Tag	ODBCCanUpdate(6)

ODBCClose (numHandler)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Closes a connection to the database.
- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
-------------------	--

- **Returned Values:**

0	Success
1	Invalid Handler

- **Examples:**

Tag Name	Expression
Tag	ODBCClose(5)

ODBCCommitTrans (numHandler)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Commits a transaction. Call this function upon completing transactions.

- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
-------------------	--

- **Returned Values:**

0	Success
1	Invalid handler
2	Database not open
3	Error committing transaction

- **Examples:**

Tag Name	Expression
Tag	ODBCCommitTrans(1)

ODBCDelete (numHandler)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Deletes the current register. After a successful deletion, you must explicitly call one of the **Move** functions to move off the deleted record.

- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
-------------------	--

- **Returned Values:**

0	Success
1	Invalid handler
2	Database not open
3	Delete error

- **Examples:**

Tag Name	Expression
Tag	ODBCDelete(5)

ODBCExecuteSQL (numHandler, strSqlCommand)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Executes an SQL command directly. The **ODBCExecuteSQL** function does not return data records.
- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
strSqlCommand	String tag specifying a valid SQL command.

- **Returned Values:**

0	Success
1	Invalid handler
2	Database not open
3	Invalid parameter
4	Error executing SQL command

- **Examples:**

Tag Name	Expression
Tag	ODBCExecuteSQL(3, ")
Tag	ODBCExecuteSQL(4, ")

ODBCInsert (numHandler)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Inserts a new register to the database. InduSoft uses the values of the tags bound by the **ODBCbindCol** function to create the new register.

- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
-------------------	--

- **Returned Values:**

0	Success
1	Invalid handler
2	Database not open
3	Insert error

- **Examples:**

Tag Name	Expression
Tag	ODBCInsert(7)

ODBCIsBOF (numHandler)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Returns whether you have gone above the first record of the record set. (Call this function before scrolling from record to record.)

You also can use the **ODBCIsBOF** function with **ODBCIsEOF** to determine whether the record set contains any records or is empty. Immediately after calling **ODBCQuery**, and if the record set contains no records, **ODBCIsBOF** returns nonzero. 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**, the **ODBCIsBOF** function will subsequently return a nonzero.

- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
-------------------	---

- **Returned Values:**

0	Record found
Non-Zero	Record set contains no records or you scrolled backward, above the first record

- **Examples:**

Tag Name	Expression
Tag	ODBCIsBOF(1)

ODBCIsDeleted (numHandler)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Reports whether the current record was deleted. If you scroll to a record and **ODBCIsDeleted** returns a nonzero, then you must scroll to another record before you can perform any other operations.

- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
-------------------	---

- **Returned Values:**

0	Record set is not positioned on a deleted record.
Non-Zero	Record set is positioned on a deleted record.

- **Examples:**

Tag Name	Expression
Tag	ODBCIsDeleted(8)

ODBCIsEOF (numHandler)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Reports whether you have gone beyond the last record of the record set. (Call this function as you scroll from record to record.)

You also can use the **ODBCIsEOF** function to determine whether the record set contains any records or is empty. Immediately after calling **ODBCQuery** (and if the record set contains no records) **ODBCIsEOF** 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 when you call **ODBCMoveNext**, **ODBCIsEOF** will subsequently return a nonzero.

- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
-------------------	---

- **Returned Values:**

0	Record found.
Non-Zero	Record set contains no records or you scrolled backward, before the last record.

- **Examples:**

Tag Name	Expression
Tag	ODBCIsEOF(5)

ODBCIsFieldNULL(numHandler, strColName)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Reports whether a specified field in a record set was flagged as **Null**.
- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
strColName	String tag containing the column name.

- **Returned Values:**

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 (numHandler, strColName)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Reports whether a specified field is *nullable* (can be set to a **Null** value).
- **Parameters:**

NumHandler	Integer tag containing the handler returned by the ODBCOpen () function.
strColName	String tag containing the column name.

- **Returned Values:**

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 (numHandler, numOffset)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Moves the current record pointer within a record set, either forward or backward. If you pass a value of 0 for Offset, **ODBCMove** refreshes the current record.

- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
numOffset	Integer tag containing the number of rows to move forward or backward. <ul style="list-style-type: none"> • Positive values move forward, toward the end of the record set. • Negative values move backward, toward the beginning of the record set.

- **Returned Values:**

0	Success
1	Invalid handler
2	Database not open
3	Move error

- **Examples:**

Tag Name	Expression
Tag	ODBCMove(2, 3)
Tag	ODBCMove(8, 2)

ODBCMoveFirst (numHandler)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Moves to the first record within the record set.
- **Parameters:**

NumHandler	Integer tag containing the handler returned by the ODBCOpen () function.
-------------------	---

- **Returned Values:**

0	Success
1	Invalid handler
2	Database not open
3	Move error

- **Examples:**

Tag Name	Expression
Tag	ODBCMoveFirst(4)

ODBCMoveLast (numHandler)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Moves to the last record within the record set.
- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
-------------------	---

- **Returned Values:**

0	Success
1	Invalid handler
2	Database not open
3	Move error

- **Examples:**

Tag Name	Expression
Tag	ODBCMoveLast(7)

ODBCMoveNext (numHandler)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Moves to the next record within the record set.
- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
-------------------	---

- **Returned Values:**

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 (numHandler)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Moves to the next record within the record set.
- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
-------------------	---

- **Returned Values:**

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(strDsn, strUser, strPassw, strTable, strFilter, strSort)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Opens a connection to the database.

This function returns a handler to be used in subsequent calls to the ODBC functions.

After calling this function, no register has been read from the database yet. You must bind the columns and call the **ODBCQuery** function to retrieve the first register.

- **Parameters:**

strDsn	String tag containing the Data Source Name.
strUser	String tag containing the User name.
strPassw	String tag containing the Password.
strTable	String tag containing the Database table name.
strFilter	String tag containing the SQL WHERE clause.
strSort	String tag containing the SQL ORDER BY clause.

- **Returned Values:**

N	On success, returns the handler to identify the database
-1	Invalid parameter
-2	DSN or TableName contain an empty string

- **Examples:**

Tag Name	Expression
Tag	ODBCOpen("MyDSNFile", "Alex", "", "Table1", "Name='Mayer'", "Name ASC")
Tag	ODBCOpen("DSNFileName", "Robert", "Robot", "Table1", "", "")

 **Note:**

This function does not open the database itself; it simply creates a handle to manipulate the database. To open the database, you must bind the columns and call the **ODBCQuery** function.

ODBCQuery (numHandler)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Retrieves the first register after opening and binding the column. If you modify the column binding, or modify the filter and sort, you must call this function again.

- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
-------------------	---

- **Returned Values:**

0	Success
1	Invalid handler
2	No columns bound
3	Cannot open database
4	Cannot restart database
5	Query error

- **Examples:**

Tag Name	Expression
Tag	ODBCQuery(6)

ODBCRollback (numHandler)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Reverses the changes made during a transaction.
- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
-------------------	---

- **Returned Values:**

0	Success
1	Invalid handler
2	Database no open
3	Error rolling back transaction

- **Examples:**

Tag Name	Expression
Tag	ODBCRollback(4)

ODBCSetFieldNull(numHandler, strColName, numValue)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Flags a field data member in the record set as Null (specifically having no value) or as non-Null.
- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
strColName	String tag containing the column name.
numValue	Integer tag, which specifies the field data as Null if 0 and non-Null in non-zero.

- **Returned Values:**

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(numHandler, strFilter)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Constrains the records selected by IWS.
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.

- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
strFilter	String tag containing the SQL WHERE clause.

- **Returned Values:**

0	Success
1	Invalid handler
2	Invalid parameter

- **Examples:**

Tag Name	Expression
Tag	ODBCSetFilter(3, "Name='Morgan'")

ODBCSetSort (numHandler, strSort)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Sorts records selected by IWS.

You can use this feature to sort the records in one or more columns. Remember to call **ODBCQuery** after calling this function.

- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
strSort	String tag containing the SQL ORDER BY clause.

- **Returned Values:**

0	Success
1	Invalid handler
2	Invalid parameter type

- **Examples:**

Tag Name	Expression
Tag	ODBCSetSort(5,"Name DESC")

ODBCUnbindCol (numHandler, strColName)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Unbinds a column from a tag.
- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
strColName	String tag containing the column name.

- **Returned Values:**

0	Success
1	Invalid handler
2	Invalid parameter type
3	Column not bound

- **Examples:**

Tag Name	Expression
Tag	ODBCUnbindCol(7, "Name")

ODBCUpdate (numHandler)

Group	ODBC
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Updates the current register. IWS uses the values of tags bound by the **ODBCBindCol** function to update the current register.

- **Parameters:**

numHandler	Integer tag containing the handler returned by the ODBCOpen () function.
-------------------	---

- **Returned Values:**

0	Success
1	Invalid handler
2	Database not open
3	Update error

- **Examples:**

Tag Name	Expression
Tag	ODBCUpdate(1)

Mail Functions

This section describes the following InduSoft Web Studio Email functions:

- `CNFEMail(strSMTP, strFrom, strPOP3, strUser, strPassword, optnumTimeOut)`
- `GetStatusSendEMailExt(optTagName)`
- `SendEMail(strSubject, strMessage, strTO)`
- `SendEMailExt(strSubject, strMessage, strTO, strCC, strBCC, strFile1, ..., strFileN)`
- `CnfEmail()`

CnfEmail(strSMTP, strFrom, strPOP3, strUser, strPassword, optnumTimeOut, optnumAuthType, optStrSMTPUser, optStrSMTPPassword)

Group	Mail
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Sets Simple Mail Transfer Protocol (SMTP) parameters. You must configure these parameters and execute this function before sending email with IWS (e.g. by the `SendEmail()` function, by the `SendEmailExt()` function, by the Alarm Email interface, etc.).
- **Parameters:**

strSMTP	String tag containing the SMTP server name or IP address. For CEView applications, you must specify the SMTP IP address instead of the server name.
strFrom	String tag containing the sender's address. This address must be a valid email address for the POP3 Server.
strPOP3	String tag containing the POP3 server name.
strUser	String tag containing the sender's user account name for the POP3 Server.
strPassword	String tag containing the password for the user account name for the POP3 Server.
optnumTimeOut	<i>Optional</i> numerical tag containing the timeout limit (in seconds) used when sending messages. If no answer is received from the Server after this period of time, the operation is aborted automatically.

optAuthType	<i>Optional</i> numerical tag containing the SMTP Authentication Type: 0 (default) = SMTP Server does not require authentication 1 = SMTP server requires authentication
optStrSMTPUser	<i>Optional</i> string tag containing the User Name used to log onto on the SMTP Server when it requires authentication. If this parameter is omitted and the optAuthType type is set to 1, the value configured for the strUser is automatically used as optStrSMTPUser.
optStrSMTPPassword	<i>Optional</i> string tag containing the Password used to log onto the SMTP Server that requires authentication. If this parameter is omitted and the optAuthType type is set to 1, the value configured for the strPassword is automatically used as optStrSMTPPassword too.

▪ **Returned Values:**

0	Success
1	Invalid format for parameter 1 (strSMTP)
2	Invalid format for parameter 2 (strFrom)
3	Invalid format for parameter 3 (strPOP3)
4	Invalid format for parameter 4 (strUser)
5	Invalid format for parameter 5 (strPassword)
6	Invalid format for parameter 6 (optnumTimeOut)
7	Wrong amount of parameters
8	Error getting host IP address (invalid POP3 server)
9	Error Connecting POP3 server
10	Error sending username
11	Error sending password
12	SMTP server does not support Login\Password Authentication mode
13	Invalid SMTP User Name
14	Authentication failed (Invalid User and/or Password)

▪ **Examples:**

Tag Name	Expression
Tag	CnfEmail("smtp.Studio.com", "Robert@Studio.com", "pop.Studio.com", "RobertH", "Shades556", 100)
Tag	CnfEmail("smtp.Studio.com", "Robert@Studio.com", "pop.Studio.com", "RobertH", "Shades556", 5 ,1)

Tag	CnfEmail("smtp.Studio.com", "Robert@Studio.com", "pop.Studio.com", "RobertH", "Shades556", 5, 1, "JohnS", "abcd1234")
-----	---

GetStatusSendEMailExt (optTagName)

Group	Mail
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns status of the last email sent using the **SendEMailExt ()** function.
- **Parameters:**

OptTagName	<i>Optional</i> tag that causes the function to update its return value. This parameter is optional but you must use it when configuring this function for any screen dynamic (such as Text I/O, Position, and so forth).
-------------------	---

- **Returned Values:**

-2	Incorrect version of the INDMail.DLL library.
-1	The INDMail.DLL 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(strSubject, strMessage, strTO)

Group	Mail
Execution	Synchronous
Windows 2K/XP/Vista	Supported

Windows CE	Supported
Web Thin Client	Supported

- **Description:** Sends e-mail messages. Before executing this function, you must set some parameters using the **CnfEmail()** function
- **Parameters:**

strSubject	String tag containing the e-mail subject.
strMessage	String tag containing the e-mail message (up to 255 characters).
strTo	String tag containing the recipient's address (target).

- Returned Values:

0	Success
1	Invalid format for parameter 1 (strSubject)
2	Invalid format for parameter 1 (strMessage)
3	Invalid format for parameter 3 (strTo)
4	Wrong amount of parameters
5	Start Socket error
6	Error getting host IP Address (invalid SMTP server)
7	Error Connecting SMTP server
8	Error sending HELO command (initialization)
9	Error sending MAIL command (sending FROM address)
10	Error sending RCPT command (sending TO address)
11	Error sending DATA (sending message)

- Examples:

Tag Name	Expression
Tag	SendEmail("Subject", "Message", "Wrogers@pnd.net")

SendEmailExt(strSubject, strMessage, strTO, strCC, strBCC, strFile1, ..., strFileN)

Group	Mail
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- Description:** Sends e-mail messages with attached files. Before executing this function, you must set some parameters using the **CnfEmail()** function.

- Parameters:

strSubject	String tag containing the e-mail subject (up to 255 characters).
strMessage	String tag containing the e-mail message (up to 255 characters).
strTO	String tag containing the recipient's address.
strCC	String tag containing the recipients' addresses to be CCed.
strBCC	String tag containing the recipients' addresses to be BCCed.
strFile (1-N)	String tags, each containing file name and path of a file to send with e-mail.

- Returned Values:

-4	Some of the attached files were not found.
-3	Wrong number of parameters (at least three parameters are required).
-2	The INDMail.DLL library version is incorrect.
-1	The INDMail.DLL library is corrupted.
0	Success
1	Cannot execute the function because the last e-mail has not been sent yet.
2	Internal error

- Examples:

Tag Name	Expression
Tag	SendEmailExt("Subject", "Message", "Sam@universe.com", "", "", "C:\Projects\report.txt")
Tag	SendEmailExt("Subject", "Message", "David@Ohio.net", "Ted@Austin.com", "Bart@Springfield.gov", "C:\TechRef51.doc")

- Notes:

- You **must** configure the **strSubject**, **strMessage**, and **strTO** parameters. All of the other parameters are *optional*.
- You can use the null string value ("") for the **strTO**, **strCC**, or **strBCC** parameters if you will not be using them.
- You can assign more than one recipient in the **strTO**, **strCC**, or **strBCC** parameters, using the semicolon (;) char to share the addresses.

Dial-Up Functions

This section describes the following InduSoft Web Studio Dial-Up functions:

- `DialError(numType, strPhonebookEntryOrModem, optError, optRefresh)`
- `DialGetClientIP(numType, strPhoneBookOrDevice, "tagClientIP", optRefresh)`
- `DialGetServerIP(numType, strPhoneBookOrDevice, "tagServerIP", optRefresh)`
- `DialStatus(numType, strPhonebookEntryOrModem, optStatus, optRefresh)`
- `DialUp(numType, strPhonebookEntryOrModem, strUserName, strPassword, optStrDomain, strPhoneNumber)`
- `DialUpToCE(numModem, strDialPhone, strMyNumber, strUser, strPassword, optStrDomain, optAutoDial, optAutoClose)`
- `FindAllDevices("tagArray")`
- `FindModem("tagArray")`
- `HangUp(numType, strPhonebookEntryOrModem)`
- `PhoneDialUp(strPhoneNumber, OptStrModemName)`
- `PhoneDisableListen(optStrModemName)`
- `PhoneEnableListen(optStrModemName)`
- `PhoneHangUp(optStrModemName)`
- `PhoneStatus("strStatus", optStrModemName)`

DialError (numType, strPhonebookEntryOrModem, optError, optRefresh)

Group	Dial-Up
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the error codes regarding each connection.
- **Parameters:**

numType	Numerical tag specifying the content of the strPhonebookEntryOrModem tag. <ul style="list-style-type: none"> • 0: Phonebook Name • 1: Modem Name • 2: Direct Connection Name
strPhonebookEntryOrModem	String tag containing a Phonebook Name, Modem Name, or Direct Connection Name used to make the connection. The numType tag specifies which of these methods is used.
optError	<i>Optional</i> string tag containing the name of the string tag receiving the Error Message.
optRefresh	<i>Optional</i> tag, which causes the function to update its return value. This parameter is optional but you must use it when configuring this function for any screen dynamic (such as Text I/O, Position, and so forth).

- **Returned Values:**

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.
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.
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 . INF file section.
654	A command or response in the device . INF file section refers to an undefined acro.
655	The <message macro was not found in the device . INF file section.
656	The <defaultoff macro in the device . INF file section contains an undefined macro.
657	The device . INF file could not be opened.
658	The device name in the device . INF or media . INI file is too long.
659	The media . INI file refers to an unknown device name.
660	The device . INF file contains no responses for the command.
661	The device . INF file is missing a command.
662	Attempted to set a macro not listed in device . INF file section.

663	The media .INI 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 .INI file.
668	The connection dropped.
669	The usage parameter in the media .INI file is invalid.
670	Cannot read the section name from the media .INI file.
671	Cannot read the device type from the media .INI file.
672	Cannot read the device name from the media .INI file.
673	Cannot read the usage from the media .INI file.
674	Cannot read the maximum connection BPS rate from the media .INI file.
675	Cannot read the maximum carrier BPS rate from the media .INI 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
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 . INF file is not in the expected format.
699	The device response caused buffer overflow.
700	The expanded command in the device . INF 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 TCPCFG.DLL .
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.
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.

▪ **Examples:**

Tag Name	Expression
Tag	DialError(0, "Office DialUp")
Tag	DialError(1, "USRobotics_SportsterFaxModem", "StatusMessage", second)
Tag	DialError(2, "DirectDial", "DialupError")

DialGetClientIP (numType, strPhoneBookOrDevice, "tagClientIP", optRefresh)

Group	Dial-Up
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Obtains the IP Address for the RAS Client station of a Dial-Up connection. You must execute this function on the RAS Client station (*not* on the RAS Server station) or the function will not work properly.
- **Parameters:**

numType	Numerical tag specifying the content of the strPhonebookEntryOrModem tag. <ul style="list-style-type: none"> • 0: Phonebook Name • 1: Modem Name • 2: Direct Connection Name
strPhonebookEntryOrModem	String tag containing the Phonebook Name, Modem Name, or Direct Connection Name used to make the connection. The numType tag specifies which of these methods is used.
"tagClientIP"	Name of the string tag that will receive the IP address.

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 any screen dynamic (such as Text I/O, Position, and so forth).
-------------------	---

Returned Values:

-5	GetClientIP function was not found in IndRAS . DLL .
-4	Invalid type (0: Phonebook, 1: Modem, 2: Direct Connection)
-3	Invalid number of parameters
-2	DialStatus function was not found in IndRAS . DLL
-1	IndRAS . DLL was not loaded
N	N: Status code returned by the DialStatus () function

Examples:

Tag Name	Expression
Tag	DialGetClientIP(0, "Office DialUp", "ClientIPTag")
Tag	DialError(1, "USRobotics_SportsterFaxModem", "ClientIPAddress", second)
Tag	DialError(2, "DirectDial", "IPAdd")

DialGetServerIP (numType, strPhoneBookOrDevice, "tagServerIP", optRefresh)

Group	Dial-Up
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported

- **Description:** Obtains the IP Address of the RAS Server station for a Dial-Up connection. You must execute this function on the RAS Client station (*not* on the RAS Server station) for this function to work properly.

Parameters:

numType	Numerical tag specifying the content of the strPhonebookEntryorModem tag. <ul style="list-style-type: none"> • 0: Phonebook Name • 1: Modem Name • 2: Direct Connection Name
----------------	--

strPhonebookEntryOrModem	String tag containing the Phonebook Name, Modem Name, or Direct Connection Name used to make the connection. The numType tag specifies which of these methods is used.
"tagServerIP"	Name of string tag receiving the IP address.
optRefresh	<i>Optional</i> tag that causes the function to update its return value. This parameter is optional, but you must use it when configuring this function for any screen dynamic (such as Text I/O, Position, and so forth).

- Returned Values:

-5	GetServerIP function was not found in IndRAS . DLL .
-4	Invalid type (0: Phonebook, 1: Modem, 2: Direct Connection)
-3	Invalid number of parameters
-2	DialStatus function was not found in IndRAS . DLL
-1	IndRAS . DLL was not loaded
N	N: Status code returned by the DialStatus () function

- Examples:

Tag Name	Expression
Tag	DialGetClinetIP(0, "Office DialUp", "ClientIPtag")
Tag	DialError(1, "USRobotics_SportsterFaxModem", "ClientIPAddress", second)
Tag	DialError(2, "DirectDial", "IPAdd")

DialStatus(numType, strPhonebookEntryOrModem, optStatus, optRefresh)

Group	Dial-Up
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- Description:** Returns the status of each connection
- Parameters:**

NumType	Numerical tag specifying the content of the strPhonebookEntryOrModem tag. <ul style="list-style-type: none"> • 0: Phonebook Name • 1: Modem Name • 2: Direct Connection Name
strPhonebookEntryOrModem	String tag containing the Phonebook Name, Modem Name, or Direct Connection Name used to make the connection. The numType tag specifies which of these methods is used.
optStatus	<i>Optional</i> string tag containing the name of the string tag receiving the status message.
optRefresh	<i>Optional</i> tag that causes the function to update its return value. This parameter is optional, but you must use it when configuring this function for any screen dynamic (such as Text I/O, Position, and so forth).

▪ **Returned Values:**

-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: INDRAS . DLL damaged
-1	Error: INDRAS . DLL 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
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 RASPHONE . exe .
25	Retry authentication state supported by RASPHONE . exe .
26	Callback state supported by RASPHONE . exe .
27	Change password state supported by RASPHONE . exe .
8192	Connected to remote server successfully!
8193	Disconnected.

- **Examples:**


Tag Name	Expression
Tag	DialError(0, "Office DialUp")
Tag	DialError(1, "USRobotics_SportsterFaxModem", "StatusMessage", second)
Tag	DialError(2, "DirectDial", "DialupError")

DialUp(numType, strPhonebookEntryOrModem, strUserName, strPassword, optStrDomain, strPhoneNumber)

Group	Dial-Up
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Triggers a dial-up connection
- **Parameters:**

numType	Numerical tag specifying the content of the strPhonebookEntryOrModem tag. <ul style="list-style-type: none"> • 0: Phonebook Name • 1: Modem Name • 2: Direct Connection Name
strPhonebookEntryOrModem	String tag containing the Phonebook Name, Modem Name, or Direct Connection Name used to make the connection. The numType tag specifies which of these methods is used.
strUserName	String tag containing the Username to use for logging on.
strPassword	String tag containing the Password to use for logging on.
optStrDomain	<i>Optional</i> string tag containing the domain name to specify when logging on.
strPhoneNumber	String tag containing the phone number to dial (used only when the parameter numType=1).

 **Note for Windows NT/2000 Users:**
Due to limitations in the Microsoft API, the **DialUp()** function may not be able to create temporary phonebooks on Windows NT/2000.

To work around this problem:

1. Configure **DialUp()** using the **Modem Name** parameter instead of **Phonebook Name**.
2. Create a phonebook manually in the *Control Panel* and save it as *IWS Temporary 1*.

IWS will use the *IWS Temporary 1* phonebook with the parameters you configured using the **DialUp()** function (**user name**, **password**, **domain**, and **telephone**). You then can change these parameters at runtime.

- **Returned Values:**

0	OK: dialing started
-1	Error: INDRAS . DLL not found
-2	Error: INDRAS . DLL damaged
-3	Error: invalid number of parameters (minimum=5)
-4	Invalid value for the numType parameter (0 or 1)
-5	Invalid value for the parameter strPhonebookEntryOrModem (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"> • If numType = 0: Could not read properties from PhoneBook. • If numType = 1: More than 1000 connections are enabled at same time.
-9	Unable to create a temporary PhoneBook.

- **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")

- **Note:**

The operating system's RAS Server executes the dial-in for Windows 2K/XP/Vista computers automatically.

DialUpToCE(numModem, strDialPhone, strMyNumber, strUser, strPassword, optStrDomain, optAutoDial, optAutoClose)

Group	Dial-Up
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported

- **Description:** Executes the **DialUpToCE** program, which sends the information necessary to **CERasSvr.exe** calls back to the Windows 2K/XP/Vista computer.
- **Parameters:**

numModem	Numerical tag containing the modem number used to dial to the WinCE computer.
strDialPhone	Numerical tag containing the telephone number of the WinCE remote station.
strMyNumber	Numerical tag containing the telephone number sent to the WinCE remote station. CERasSvr.exe will call back to this phone number.
strUser	String tag containing the user name to be sent to WinCE remote station. CERasSvr.exe will use this name to connect to the Windows 2K/XP/Vista computer after calling back to it.


strPassword	String tag containing the password to be sent to WinCE remote station. CERasSvr.exe will use this password to connect to the Windows 2K/XP/Vista computer after calling back to it
optStrDomain	<i>Optional</i> string tag containing the domain name to specify when logging on.
optAutoDial	<i>Optional</i> tag, which can be set to one of the following: <ul style="list-style-type: none"> • 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	<i>Optional</i> tag, which can be set to one of the followign: 1: Closes the <i>DialUpToCE</i> dialog automatically after dialing the WinCE remote station 0: Leaves the <i>DialUpToCE</i> dialog open

▪ **Returned Values:**

0	Fail, unable to call DialUpToCE .
1	Success, DialUpToCE executed.

▪ **Examples:**

Tag Name	Expression
Tag	DialUpToCE(0,"12344321","98765432", "Administrator", "MyPass")
Tag	DialUpToCE(0,"12344321","98765432", "Administrator", "MyPass", "", 1,1)

 **Note:**
 The **DialUpToCE** program was developed to dial a remote WinCE station. Because Windows CE v3.00 does not provide a RAS Server, you must be running the **CERasSvr.exe** program on the WinCE device to answer a call, and call back to a Windows 2K/XP/Vista computer using parameters sent by the **DialUpToCE ()** function. You must configure the RAS Server service on the Windows 2K/XP/Vista computer to answer the call back from the WinCE device and set the TCP/IP connection.

FindAllDevices ("tagArray")

Group	Dial-Up
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported

Web Thin Client	Supported
------------------------	-----------

- **Description:** Returns the list of all the available modems and direct connection interfaces (COM ports) in the local station.
- **Parameters:**

"tagArray"	Name of a string array tag receiving the list of available modems and direct connection interfaces.
-------------------	---

- **Returned Values:** Returns the number of modems and/or interfaces found.
- **Examples:**

Tag Name	Expression
Tag	FindAllDevices("SerialConnections[1]")

FindModem("tagArray")

Group	Dial-Up
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Returns the list of all available modems in the local station.
- **Parameters:**

"tagArray"	Name of a string array tag receiving the list of available modems.
-------------------	--

- **Returned Values:** Returns the number of modems found.
- **Examples:**

Tag Name	Expression
Tag	FindModem("Modems[1]")

⇒ **Tip:**

You can use the **FindModem()** function to get the serial interface name for a dial-up connection via modem, and use this information to fill the **strPhonebookEntryOrModem** parameter for the **DialError()**, **DialStatus()**, **DialUp()**, and **HangUp()** functions.

HangUp(numType, strPhonebookEntryOrModem)

Group	Dial-Up
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Hangs-up a dial-up connection.
- **Parameters:**

numType	Numerical tag specifying the content of the strPhonebookEntryOrModem tag. <ul style="list-style-type: none"> • 0: Phonebook Name • 1: Modem Name • 2: Direct Connection Name
strPhonebookEntryOrModem	String tag containing the Phonebook Name, Modem Name, or Direct Connection Name used to make the connection. The numType tag specifies which of these methods is used.

- **Returned Values:** This function returns the following values:

0	OK.
-1	Error: INDRAS . DLL not found
-2	Error: INDRAS . DLL damaged
-3	Invalid value for the numType parameter (0 or 1)
-4	PhoneBook or modem does not exist
-5	No configured modems exist

- **Examples:**

Tag Name	Expression
Tag	HangUp(0, "OfficeDialup")
Tag	HangUp(1, "USRobotics_SportsterFaxModem")
Tag	HangUp(2, "DirectDial", "Rberton", "MyPassword")

PhoneDialUp(strPhoneNumber, optStrModemName)

Group	Dial-Up
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported

- **Description:** Dials to a phone number using *TAPI* (Telephony Application Program Interface).
- **Parameters:**

strPhoneNumber	Telephone number the function will call.
OptStrModemName	Name of the modem used to dial. If you do not specify a modem, IWS will use the first modem found on the operating system.

- **Returned Values:** This function returns the following values:

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
	PhoneDialUp ("512-123-4567")
	PhoneDialUp (StringPhoneNumberTag)
	PhoneDialUp (StringPhoneNumberTag, StringModemNameTag)

PhoneDisableListen (optStrModemName)

Group	Dial-Up
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported

- **Description:** Disables IWS from listening to the modem for incoming calls.
- **Parameters:**

OptStrModemName	Name of the modem used to dial. If you do not specify a modem, IWS will use the first modem found on the operating system.
------------------------	--

- **Returned Values:** This function returns the following values:

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
	PhoneDisableListen ()
	PhoneDisableListen ("Hayes Compatible Modem on COM1")
	PhoneDisableListen (StringModemNameTag)

PhoneEnableListen (optStrModemName)

Group	Dial-Up
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported

- **Description:** Enables IWS to listen to the modem for incoming calls.
- **Parameters:**

OptStrModemName	Name of the modem used to dial. If you do not specify a modem, IWS will use the first modem found on the operating system.
------------------------	--

- **Returned Values:** This function returns the following values:

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
	PhoneEnableListen ()
	PhoneEnableListen ("Hayes Compatible Modem on COM1")
	PhoneEnableListen (StringModemNameTag)

PhoneHangUp (Opt StrModemName)

Group	Dial-Up
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported

- **Description:** Hangs up a dial-up connection triggered with the **PhoneDialUp ()** function.
- **Parameters:**

Opt StrModemName	Name of the modem used to dial. If you do not specify a modem, IWS will use the first modem found on the operating system.
-------------------------	--

- **Returned Values:** This function returns the following values:

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
	PhoneHangUp()
	PhoneHangUp ("Hayes Compatible Modem on COM1")
	PhoneHangUp (StringModemNameTag)

PhoneStatus ("strStatus", optStrModemName)

Group	Dial-Up
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Not Supported
Web Thin Client	Supported

- **Description:** Checks the status of the current connections.
- **Parameters:**

"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, IWS will use the first modem found on the operating system

- **Returned Values:** This function returns the following values:

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	PhoneStatus ("String Tag Status")
Tag Status Code	PhoneStatus ("String Tag Status", "Hayes Compatible Modem on COM1")
Tag Status Code	PhoneStatus ("String Tag Status", "StringModemNameTag")

ActiveX Functions

This section describes the following InduSoft Web Studio ActiveX functions:

- `XGet(strName, strProperties)`
- `XRun(strName, strMethod, Parameter1, Parameter2, ..., ParameterN)`
- `XSet(strName, strProperties, Value)`

XGet(strName, strProperties)

Group	ActiveX
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

 **Note:**

This function cannot be used with Tasks (page 8–1) or in the Global Procedures script (page 15–8).

- **Description:** Executes a (**PropGet**) on the specified ActiveX object.
- **Parameters:**

strName	String tag containing the unique name given to the target ActiveX object.
strProperties	String tag containing the specific method to be used.

- **Returned Values:** Returns the value of the property as reported by the ActiveX object.
- **Examples:**

Tag Name	Expression
Tag	<code>XGet("ActXRec", "Color")</code>

XRun(strName, strMethod, Parameter1, Parameter2, ..., ParameterN)

Group	ActiveX
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

 **Note:**

This function cannot be used with Tasks (page 8–1) or in the Global Procedures script (page 15–8).

- **Description:** Calls a method on the specified ActiveX control object.
- **Parameters:**

strName	String containing the unique name of the ActiveX object, as configured in the Name field of the <i>Object Properties</i> dialog.
strMethod	String containing the specific method to be called.
Parameter (1–N)	Data of various types that are required by the method to run. The number of parameters can vary from 0 to 255 and depends on the specified method and ActiveX object. All original data values, including strings, must be configured <i>without</i> quotes. All referring Tags must be configured with quotes (" "), and the Tag types must match the parameter types (e.g. Boolean, Integer, Real or String) on the method.

- **Returned Values:** Returns the method result as reported by the ActiveX object. (Not all methods return results.)
- **Examples:**

Tag Name	Expression
Tag	XRun("ActXCir", "XPos", 12, 4.6, 0.2, 1) // Without referring Tags
Tag	XRun("ActXCir", "XPos", <i>Parameter1</i> , <i>Parameter2</i>) // Without referring Tags
Tag	XRun("ActXCir", "XPos", "Parameter1", "Parameter2") // With referring Tags

XSet(strName, strProperties, Value)

Group	ActiveX
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

Note:

This function cannot be used with Tasks (page 8–1) or in the Global Procedures script (page 15–8).

- **Description:** Executes a (**PropPut**) on the specified ActiveX object.

- **Parameters:**

strName	String tag containing the unique name given to the target ActiveX object.
strProperties	String tag containing the specific method to use.
Value	Tag of any type containing the value to put in the ActiveX object

- **Returned Values:** No returned values.
- **Examples:**

Tag Name	Expression
Tag	XSet("ActXDisplay", "Display", "Status Normal")

Event Logger Functions

This section describes the InduSoft Web Studio Event Logger function:

`SendEvent (strEvent) .`

SendEvent (strEvent, optBooFlag, optStrComment)

Group	Event Logger
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Use to send an event to the **Event Log** file.

This function has an option that allows you to create a comment. When this option is enabled, the user is prompted to enter a comment after executing the `SendEvent()` function. This comment will be saved in the Event Logger file.

- **Parameters:**

strEvent	String value or tag containing the text to be saved in the Event Log file.
optBooFlag	If omitted or 0 (zero), the event does not have a comment. Otherwise, there is a comment associated to the event.
optStrComment	String value of tag containing the text of the comment for the event saved in the database. If omitted, the user is prompted with a standard dialog where the comment can be typed.

- **Returned Values:**

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.

▪ **Examples:**

Tag Name	Expression
Tag	SendEvent("Valve Open") // Saves the event message.
Tag	SendEvent("Valve Open Oven No." + OvenID) // Saves the event message concatenated with the value of the OvenID tag
Tag	SendEvent("Valve Open", 1) // Displays the dialog where the operator can type his comments.
Tag	SendEvent("Valve Open", 1, TagComment) // Saves the event message with the comment configured in the TagComment tag.

⚠ Caution:

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.

FTP Functions

This section describes the following IWS FTP functions:

- CNFFtp (strServer , strUser , strPwd , numPassiveMode , numPort)
- ftpGet (strRemoteFile , strLocalFile , numOverWrite , numTransferType)
- ftpPut (strLocalFile , strRemoteFile, numTransferType)
- ftpStatus ("strStatusTag")

CNFFtp (strServer , strUser , strPwn , numPassiveMode , numPort)


Group	FTP
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** Sets FTP (File Transfer Protocol) parameters. You must configure these parameters and execute this function before transferring files with the FtpGet or FtpPut functions.
- **Parameters:**

strServer	FTP server name
strUser	User name Default is <i>Anonymous</i> .
strPwn	User password Default is blank.
numPassiveMode	0 – Passive mode disabled 1 – Passive mode enabled Default is 0.
numPort	TCP/IP port number Default is 21.

- **Returned Values:**

0	Success
-1	Invalid number of parameters
-2	Invalid server name
-3	Invalid user name

 **Note:**
This function does not try to establish a connection with the FTP server, but it must be called before using any other FTP function.

▪ **Examples:**

Tag Name	Expression
Tag	CNFFtp("ftp.mycompany.com", "admin", "12345", 1) // Configures the ftp server using passive mode

ftpGet (strRemoteFile , strLocalFile , numOverWrite , numTransferType)

Group	FTP
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** This function retrieves a file from the FTP server and stores it under the specified file name, creating a new local file. Before executing this function you must set some parameters using CNFFtp. This function is executed asynchronously; therefore you need to call FtpStatus in order to determine if the transfer has been completed successfully.

▪ **Parameters:**

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 with the correct capitalization.
strLocalFile	Full qualified name of the local file (e.g. "C:\file.extension")
numOverWrite	0 – Error if the Local File already exists 1 – Overwrite Default is 0.
numTransferType	0 – Unknown 1 – ASCII 2 – Binary Default is 0.

▪ **Returned Values:**

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

- **Examples:**

Tag Name	Expression
Tag	FtpGet (“\Reports\040303.txt”, “C:\Report.txt”) // Retrieves the file 040303 from the folder Reports in the FTP server and stores it in the C:\Report.txt file

ftpPut (strLocalFile , strRemoteFile, numTransferType)

Group	FTP
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** This function stores a file on the FTP server. Before executing this function you must set some parameters using CNFFtp. This function is executed asynchronously; therefore you need to call FtpStatus in order to determine if the transfer has been completed successfully.

- **Parameters:**

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 with the correct capitalization.
optnumTransferType	0 – Unknown 1 – ASCII 2 – Binary Default is 0.

- **Returned Values:**

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

▪ **Examples:**

Tag Name	Expression
Tag	FtpPut ("C:\Report.txt", "\Reports\040303.txt") // Retrieves the file "C:\Report.txt" in the with the name 040303.txt in the folder Reports in the FTP Server.

ftpStatus ("strStatusTag")

Group	FTP
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

- **Description:** This function returns the current status of a transaction started with FtpGet or FtpPut. The return value indicates the status as described below.

▪ **Parameters:**

"strStatusTag"	String with status description.
----------------	---------------------------------

▪ **Returned Values:**

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 opttagErrorDescription
-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

- **Examples:**

Tag Name	Expression
Tag	FtpStatus("StatusDescription") // Retrieves the status of a current transfer. The return code is stored in the StatusCode tag and the description in the StatusDescription tag.

DB/ERP Functions

This section describes the InduSoft Web Studio DB/ERP functions:

- `DBCursorClose(numCur, optStrErrorTag)`
- `DBCursorCurrentRow(numCur, optStrErrorTag)`
- `DBCursorGetValue(numCur, strColumn, optStrErrorTag)`
- `DBCursorMoveTo(numCur, numRows, optStrErrorTag)`
- `DBCursorNext(numCur, optStrErrorTag)`
- `DBCursorOpen(strConn, strTable, optStrCondition, optStrColumns, optStrTags, optStrOrder, optStrErrorTag)`
- `DBCursorOpenSQL(strConn, strSQL, optStrTags, optStrErrorTag)`
- `DBCursorPrevious(numCur, optStrErrorTag)`
- `DBCursorRowCount(numCur, optStrErrorTag)`
- `DBDelete(strConn, strTable, strCondition, optStrErrorTag)`
- `DBExecute(strConn, strSQL, optStrTags, optNumMaxRows, optStrErrorTag)`
- `DBInsert(strConn, strTable, strValues, optStrColumns, optStrErrorTag)`
- `DBSelect(strConn, strTable, strTags, optStrColumns, optStrCondition, optStrOrder, optNumMaxRows, optStrErrorTag)`
- `DBUpdate(strConn, strTable, strValues, strColumns, optStrCondition, optStrErrorTag)`
- `SyncAlarm(optStrStartDate, optStrEndDate)`
- `SyncEvent(optStrStartDate, optStrEndDate)`
- `SyncTrend(numGroup, optStrStartDate, optStrEndDate)`
- `SyncAlarmStatus()`
- `SyncEventStatus()`
- `SyncTrendStatus(numGroup)`

DBCursorClose(numCur, optStrErrorTag)

Group	DB/ERP
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

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.

- **Description:** Closes the cursor and releases the dataset. After the cursor is closed, it is destroyed and it cannot be opened again; you must create a new cursor using **DBCursorOpen ()** or **DBCursorOpenSQL ()**.

- **Parameters:**

numCur	A numeric value or tag; the cursor handle created by DBCursorOpen () or DBCursorOpenSQL () .
optStrErrorTag	A tag of String type that will receive detailed error messages, if errors occur during runtime. This is an optional parameter.

- **Returned Values:** Returns an error code. In case of error, returns a negative number. When no error occurs, it returns zero.
- **Examples:**

Tag Name	Expression
Tag	DBCursorClose(nCursor)

DBCursorCurrentRow(numCur, optStrErrorTag)

Group	DB/ERP
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

 **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.

- **Description:** Returns the number of the current row (i.e. the position of the cursor).
- **Parameters:**


numCur	A numeric value or tag; the cursor handle created by DBCursorOpen () or DBCursorOpenSQL () .
optStrErrorTag	A tag of String type that will receive detailed error messages, if errors occur during runtime. This is an optional parameter.

- **Returned Values:** Returns the number of the current row. In case of error, returns a negative number.
- **Examples:**

Tag Name	Expression
Tag	DBCursorCurrentRow(nCursor)

DBCursorGetValue(numCur, strColumn, optStrErrorTag)

Group	DB/ERP
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

 **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.

- **Description:** Gets the value in the specified column of the current row (i.e. the cursor position).
- **Parameters:**

numCur	A numeric value or tag; the cursor handle created by DBCursorOpen () or DBCursorOpenSQL () .
strColumn	A string or a tag of String type; the name of the specific column.
optStrErrorTag	A tag of String type that will receive detailed error messages, if errors occur during runtime. This is an optional parameter.

- **Returned Values:** Returns the value in the specified column. If the value is NULL or the cursor is invalid, then it returns an empty string with quality BAD.
- **Examples:**

Tag Name	Expression
Tag	DBCursorGetValue(nCursor, "Column1")

DBCursorMoveTo(numCur, numRows, optStrErrorTag)

Group	DB/ERP
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

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.

- **Description:** Moves the cursor to the specified row of the dataset and copies those values to the mapped tags. If the specified row does not exist — that is, if it's outside the range of the dataset — then the function returns an error code and does not change the mapped tags.

▪ **Parameters:**

numCur	A numeric value or tag; the cursor handle created by DBCursorOpen () or DBCursorOpenSQL () .
numRow	A numeric value or tag; the row of the dataset to which the cursor will be moved.
optStrErrorTag	A tag of String type that will receive detailed error messages, if errors occur during runtime. This is an optional parameter.

- **Returned Values:** Returns an error code. In case of error, returns a negative number. When no error occurs, it returns zero.

▪ **Examples:**

Tag Name	Expression
Tag	DBCursorMoveTo(nCursor, 4) // Moves to the fourth row of the dataset and copies those values.

DBCursorNext (numCur, optStrErrorTag)

Group	DB/ERP
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

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.

- **Description:** Moves the cursor to the next row of the dataset and copies those values to the mapped tags. If there is no next row — that is, if the current row is the last one — then the function returns an error code and does not change the mapped tags.

▪ **Parameters:**

numCur	A numeric value or tag; the cursor handle created by DBCursorOpen () or DBCursorOpenSQL () .
optStrErrorTag	A tag of String type that will receive detailed error messages, if errors occur during runtime. This is an optional parameter.

- **Returned Values:** Returns an error code. In case of error, returns a negative number. When no error occurs, it returns zero.

▪ **Examples:**

Tag Name	Expression
Tag	DBCursorNext (nCursor)

DBCursorOpen(strConn, strTable, optStrCondition, optStrColumns, optStrTags, optStrOrder, optStrErrorTag)

Group	DB/ERP
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

 **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.

- **Description:** Creates a cursor handle for a selected dataset.
- **Parameters:**

strConn	A string or a tag of String type; the name of the database connection. Connections are configured in the <i>External Databases</i> folder. For more information, see page 8–35.
strTable	A string or a tag of String type; the name of the table in the database.
optStrCondition	A string or a tag of String type; the condition statement to filter the rows of the table. (This is equivalent to the SQL “Where” clause.) This is an optional parameter; if no condition is given, then all rows of the table will be selected.
optStrColumns	A string or a tag of String type; a list of the columns, separated by commas, to be included in the selection. This is an optional parameter; if no columns are specified, then all columns will be included.
optStrTags	A string or a tag of String type; a list of application tags, separated by commas, to map to the columns. As the cursor is moved through the dataset, the values in the current row will be written to these tags. This is an optional parameter; if no tags are specified, then no values will be written.
optStrOrder	A string or a tag of String type; the order in which the rows will be sorted. (This is equivalent to the SQL “Order By” clause.) This is an optional parameter; if no order is specified, then the rows will be left in the default order of the table.
optStrErrorTag	A tag of String type that will receive detailed error messages, if errors occur during runtime. This is an optional parameter.


- **Returned Values:** Returns a numeric value that represents the cursor handle. In case of error, returns a negative number.
- **Examples:**

Tag Name	Expression
----------	------------

nCursor	<pre>DBCursorOpen("DB1", "Table1", "Column1>3", "Column1,Column2", "Tag1,Tag2", "Column1, Column2 DESC", "TagError") // Opens Table1 of DB1 and selects all rows where Column1 has a value greater than 3. Column1 is mapped to Tag1, and Column2 is mapped to Tag2. Rows are ordered first by Column1, then by Column2, in descending order. Error messages are written to TagError.</pre>
---------	---

DBCursorOpenSQL(strConn, strSQL, optStrTags, optStrErrorTag)

Group	DB/ERP
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

 **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.

- **Description:** Creates a cursor handle for a selected dataset, as specified by a SQL “Select” statement.
- **Parameters:**

strConn	A string or a tag of String type; the name of the database connection. Connections are configured in the <i>External Databases</i> folder. For more information, see page 8–35.
strSQL	A string or a tag of String type; a custom SQL “Select” statement.
optStrTags	A string or a tag of String type; a list of application tags, separated by commas, to map to the columns. As the cursor is moved through the dataset, the values in the current row will be written to these tags. This is an optional parameter; if no tags are specified, then no values will be written.
optStrErrorTag	A tag of String type that will receive detailed error messages, if errors occur during runtime. This is an optional parameter.

- **Returned Values:** Returns a numeric value that represents the cursor handle. In case of error, returns a negative number.
- **Examples:**

Tag Name	Expression
nCursor	DBCursorOpenSQL("DB1", "SELECT Column1, Column2 FROM Table1 WHERE Column1>3 ORDER BY Column1, Column2 DESC", "Tag1,Tag2")

DBCursorPrevious (numCur, optStrErrorTag)

Group	DB/ERP
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

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.

- **Description:** Moves the cursor to the previous row of the dataset and copies those values to the mapped tags. If there is no previous row — that is, if the current row is the first one — then the function returns an error code and does not change the mapped tags.

- **Parameters:**

numCur	A numeric value or tag; the cursor handle created by DBCursorOpen () or DBCursorOpenSQL () .
optStrErrorTag	A tag of String type that will receive detailed error messages, if errors occur during runtime. This is an optional parameter.

- **Returned Values:** Returns an error code. In case of error, returns a negative number. When no error occurs, it returns zero.
- **Examples:**

Tag Name	Expression
Tag	DBCursorPrevious(nCursor)

DBCursorRowCount (numCur, optStrErrorTag)

Group	DB/ERP
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

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.

- **Description:** Return the total number of rows in the selected dataset.
- **Parameters:**


numCur	A numeric value or tag; the cursor handle created by DBCursorOpen () or DBCursorOpenSQL () .
optStrErrorTag	A tag of String type that will receive detailed error messages, if errors occur during runtime. This is an optional parameter.

- **Returned Values:** Returns the number of rows. In case of error, returns a negative number.
- **Examples:**

Tag Name	Expression
Tag	DBCursorRowCount (nCursor)

DBDelete(strConn, strTable, strCondition, optStrErrorTag)

Group	DB/ERP
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

 **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.

- **Description:** Deletes all rows in the selected dataset.
- **Parameters:**

strConn	A string or a tag of String type; the name of the database connection. Connections are configured in the <i>External Databases</i> folder. For more information, see page 8–35.
strTable	A string or a tag of String type; the name of the table in the database.
strCondition	A string or a tag of String type; the condition statement to filter the rows of the table. (This is equivalent to the SQL “Where” clause.) This is an optional parameter; if it is left blank, then all rows of the table will be deleted.
optStrErrorTag	A tag of String type that will receive detailed error messages, if errors occur during runtime. This is an optional parameter.

- **Returned Values:** Returns the number of rows deleted from the table. In case of error, returns a negative number.
- **Examples:**

Tag Name	Expression
Tag	DBDelete("DB1", "Table1", "Column1>1000", "TagError")

DBExecute(strConn, strSQL, optStrTags, optNumMaxRows, optStrErrorTag)

Group	DB/ERP
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

 **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.

- **Description:** Executes a custom SQL command. If the command is a query (SELECT), then the values are copied to specified array tags.
- **Parameters:**


strConn	A string or a tag of String type; the name of the database connection. Connections are configured in the <i>External Databases</i> folder. For more information, see page 8–35.
strSQL	A string or a tag of String type; a custom SQL command.
optStrTags	A string or a tag of String type; a list of array tags, separated by commas, to map to the columns. The values in the columns will be copied to these tags. This is an optional parameter.
optNumMaxRows	A numeric value or tag; the number of rows to be read. This is an optional parameter; if no number is specified, then only the first row of the selected dataset will be read.
optStrErrorTag	A tag of String type that will receive detailed error messages, if errors occur during runtime. This is an optional parameter.

- **Returned Values:** Returns the number of records affected by the command. In case of error, returns a negative number.
- **Examples:**

Tag Name	Expression
Tag	DBExecute("DB1", "INSERT INTO Table1(Column1,Column2) values(1,1)")
Tag	DBExecute("DB1", "SELECT max(Column1),max(Column2) FROM Table1", "Tag1,Tag2", 1, "TagError")

DBInsert(strConn, strTable, strValues, optStrColumns, optStrErrorTag)

Group	DB/ERP
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

 **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.

- **Description:** Inserts a new row into the database table.
- **Parameters:**


strConn	A string or a tag of String type; the name of the database connection. Connections are configured in the <i>External Databases</i> folder. For more information, see page 8–35.
strTable	A string or a tag of String type; the name of the table in the database.
strValues	A string or a tag of String type; a list of values, separated by commas, to be written to the specified columns. String values must be enclosed in single quotes.
optStrColumns	A string or a tag of String type; a list of column names, separated by commas.
optStrErrorTag	A tag of String type that will receive detailed error messages, if errors occur during runtime. This is an optional parameter.

- **Returned Values:** Returns the number of rows inserted in the table (one). In case of error, returns a negative number.
- **Examples:**

Tag Name	Expression
Tag	DBInsert("DB1", "Table1", "1, 'one'", "Column1, Column2")

```
DBSelect( strConn, strTable, strTags, optStrColumns,
optStrCondition, optStrOrder, optNumMaxRows, optStrErrorTag
)
```

Group	DB/ERP
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

 **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.

- **Description:** Selects a dataset and copies all values to array tags.
- **Parameters:**

strConn	A string or a tag of String type; the name of the database connection. Connections are configured in the <i>External Databases</i> folder. For more information, see page 8–35.
strTable	A string or a tag of String type; the name of the table in the database.
strTags	A string or a tag of String type; a list of array tags, separated by commas, to map to the columns. The values in the columns will be written to these tags.
optStrColumns	A string or a tag of String type; a list of the columns, separated by commas, to be included in the selection. This is an optional parameter; if no columns are specified, then all columns will be included.
optStrCondition	A string or a tag of String type; the condition statement to filter the rows of the table. (This is equivalent to the SQL “Where” clause.) This is an optional parameter; if it is left blank, then all rows of the table will be selected.
optStrOrder	A string or a tag of String type; the order in which the rows will be sorted. (This is equivalent to the SQL “Order By” clause.) This is an optional parameter; if no order is specified, then the rows will be left in the default order of the table.
optNumMaxRows	A numeric value or tag; the number of rows to be read. This is an optional parameter; if no number is specified, then only the first row of the selected dataset will be read.
optStrErrorTag	A tag of String type that will receive detailed error messages, if errors occur during runtime. This is an optional parameter.


- **Returned Values:** Returns the total number of rows in the query. In case of error, returns a negative number.

- **Examples:**

Tag Name	Expression
Tag	DBSelect("DB1", "Table1", "Array1,Array2", "Column1,Column2")
Tag	DBSelect("DB1", "Table1", "Array1,Array2", "Column1,Column2", "Column2<Column1", "Column1", 4, "TagError")

DBUpdate(strConn, strTable, strValues, strColumns, optStrCondition, optStrErrorTag)

Group	DB/ERP
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported

 **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.

- **Description:** Updates all rows in the selected dataset with new values.
- **Parameters:**

strConn	A string or a tag of String type; the name of the database connection. Connections are configured in the <i>External Databases</i> folder. For more information, see page 8–35.
strTable	A string or a tag of String type; the name of the table in the database.
strValues	A string or a tag of String type; a list of values, separated by commas, to be written to the specified columns. String values must be enclosed in single quotes.
strColumns	A string or a tag of String type; a list of column names, separated by commas.
optStrCondition	A string or a tag of String type; the condition statement to filter the rows of the table. (This is equivalent to the SQL “Where” clause.) This is an optional parameter; if it is left blank, then all rows of the table will be updated.
optStrErrorTag	A tag of String type that will receive detailed error messages, if errors occur during runtime. This is an optional parameter.

- **Returned Values:** Returns the number of rows updated. In case of error, returns a negative number.
- **Examples:**

Tag Name	Expression
Tag	<pre>DBUpdate("DB1", "Table1", "'X'", "Column2", "Column1=1", "TagError") // In Table1 of DB1, for all rows where Column1 equals 1, writes "X" to Column2.</pre>

SyncAlarm(optStrStartDate, optStrEndDate)

Group	DB/ERP
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Synchronizes the alarm [Event, Trend] database.
- **Parameters:**

strStartDate	String with the start date. If this parameter is not specified, the current date is used.
optStrEndDate	String with the end date, if this parameter is not specified the functions uses the same as the start date.

- **Returned Values:**

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.

- **Examples:**

Tag Name	Expression
Tag	SyncAlarm() //Synchronizes the database using the current date
Tag	SyncAlarm("10/20/2004") //Synchronizes the database only for the day 10/20/2004
Tag	SyncAlarm("10/20/2004", "10/28/2004") //Synchronizes the database from 10/20/2004 to 10/28/2004

SyncEvent (optStrStartDate, optStrEndDate)

Group	DB/ERP
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Synchronizes the alarm [Event, Trend] database.
- **Parameters:**

optStrStartDate	String with the start date. If this parameter is not specified, the current date is used.
optStrEndDate	String with the end date, if this parameter is not specified the functions uses the same as the start date.

- **Returned Values:**

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.

- **Examples:**

Tag Name	Expression
Tag	SyncEvent() //Synchronizes the database using the current date
Tag	SyncEvent("10/20/2004") //Synchronizes the database only for the day 10/20/2004
Tag	SyncEvent("10/20/2004", "10/28/2004") //Synchronizes the database from 10/20/2004 to 10/28/2004

SyncTrend(numGroup, optStrStartDate, optStrEndDate)

Group	DB/ERP
Execution	Asynchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Synchronizes the alarm [Event, Trend] database.
- **Parameters:**

numGroup	Trend group number
optStrStartDate	String with the start date. If this parameter is not specified, the current date is used.
optStrEndDate	String with the end date, if this parameter is not specified the functions uses the same as the start date.

- **Returned Values:**

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.

- **Examples:**

Tag Name	Expression
Tag	SyncTrend(1) //Synchronizes the group 1 database using the current date
Tag	SyncTrend(1, "10/20/2004") //Synchronizes the group 1 database only for the day 10/20/2004
Tag	SyncTrend("10/20/2004", "10/28/2004") //Synchronizes the group 1 database from 10/20/2004 to 10/28/2004

SyncAlarmStatus ()

Group	DB/ERP
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Returns the synchronization status.

- **Returned Values:**

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	SyncAlarmStatus()

SyncEventStatus ()

Group	DB/ERP
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Returns the synchronization status.

- **Returned Values:**

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	SyncEventStatus()

SyncTrendStatus (numGroup)

Group	DB/ERP
Execution	Synchronous
Windows 2K/XP/Vista	Supported
Windows CE	Supported
Web Thin Client	Supported / Executed on Server

- **Description:** Returns the synchronization status.
- **Parameters:**

numGroup	Trend group number
-----------------	--------------------

- **Returned Values:**

3	Synchronization has finished.. volume
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)

Index

- .NET
 - Control button, 3–21, 7–100
- A**
- ABS function, A-14
- ABS(numValue), A-191
- accessing
 - information about active screens, 3–26
 - Least Significant Bit (LSB), A-23
 - Most Significant Bit (MSB), A-23
 - tags/tag fields during runtime*, 5–20
 - tags/tag fields, indirectly*, 5–3
- Ack tag field, description/purpose**, 5–11
- acknowledging alarms
 - Ack field, 5–11
 - remotely, 5–21
- ACos function, A-81
- Active Objects toolbar, 3–20, 7–44
- active screens/worksheets
 - enabling/disabling dragging, 3–26
 - using the status bar, 3–26
- ActiveSync, 2–5, 2–22
- ActiveX
 - Control button, 3–20, 7–93
 - creating/configuring objects, 3–3
 - functions, A-10**, A-271
 - Program IDs/File Names, A-195
- ActiveX Events interface, 15–17
- Add/Remove Programs icon, 2–7
- Address text box, 3–17
 - Web toolbar, 7–18
- addresses, specifying limits, 10–13
- Administrator privileges, 2–1
- ADO.NET, 8–35, 17–1
- Advanced dialog
 - Trend Control Object, 7–64
- Advanced dialog, alarms, 7–49
- Alarm button, 7–110
- Alarm Worksheet
 - restrictions, 5–12
- Alarm/Event Control Object button, 3–20, 7–44
- alarms
 - AlrStatus tag field, 5–10
 - configuring message strings, 5–22
 - configuring tag properties**, 5–19
 - enabling for tags, 5–11
 - filtering strings, 5–22
 - mnemonics, 5–23
 - properties, 5–21
 - responding remotely (enabling the Server option for tags), 5–7
 - saving/storing history files, 6–13
- alarms, tag fields
 - Ack, 5–11
 - AlrDisable**, 5–11
 - DevM**, 5–11
 - DevMLimit**, 5–12
 - DevP**, 5–11
 - DevPLimit**, 5–12
 - DevSetpoint**, 5–12
 - Hi**, 5–11
 - HiHi**, 5–11
 - HiHiLimit**, 5–11
 - HiLimit**, 5–11
 - Lo**, 5–11
 - LoLimit**, 5–11
 - LoLo**, 5–11
 - LoLoLimit**, 5–12
 - Rate**, 5–11
 - RateLimit**, 5–12
- Align and Distribute toolbar, 3–17, 7–10
- Align bottom button, 3–18, 7–12
- Align left button, 3–18, 7–11
- Align right button, 3–18, 7–11
- Align top button, 3–18, 7–11
- AlrDisable tag field**
 - description/purpose**, 5–11
- AlrStatus tag field**
 - description/purpose**, 5–10
- AppActivate function, A-105
- AppIsRunning function, A-106
- application database tags, 5–13
- Application Tags**
 - adding to database, 5–13
 - datasheet, closing, 5–14
 - datasheet, opening, 5–13
 - datasheet, specifying maximum indexes*, 5–4
 - deleting, 5–26
 - description/purpose**, 5–1
 - folder*, 5–14
- Application Tags folder, 5–1
- applications
 - associating tags with objects, 5–24
 - description/purpose, 1–1
 - downloading CEView, 2–1
 - downloading to a runtime workstation, 1–1
 - downloading to remote workstations, 14–1
 - exporting screens, 1–2, 7–16
 - icons, A-136
 - interfacing with other systems/applications, 1–1
 - license levels, 6–3
 - managing remotely, 5–7, 14–1
 - merging, 1–3
 - running, 3–52
 - running on CEView, 1–1
 - setting resolution, 2–5
 - tags, 3–8
 - tags, description/purpose**, 3–8
 - testing/debugging, 12–1
 - translating, 1–2, 6–11
 - using sample template, 6–4
 - verifying, 3–51, 18–4
- AppPostMessage function, A-107
- AppSendKeys function, A-108
- Arc Cosine
 - calculating, A-81
 - returned values, A-81
- Arc Sine, calculating, A-82
- Arc Tangent, calculating, A-83
- Arithmetic functions, A-1**, A-14
- arithmetic operations

- using for array tags, 5–4
- Arithmetic Operators, 15–2
- array indexes
 - description/purpose, 5–4
 - formatting, 5–4
 - specifying maximums, 5–4
- array tags**, 5–4
 - description/purpose**, 5–3, 5–4
 - examples, 5–5
 - formatting, 5–4
 - specifying maximum index*, 5–14
 - using arithmetic operations, 5–4
 - using constants, 5–4
- Asc2Str function, A-45
- ASCII TABLES, A-43
- ASin function, A-82
- ATan function, A-83
- AutoFormat function, A-162**
- Avg function, A-31
- Axes dialog
 - Trend Control Object, 7–59
- B**
- B0 to B31 tag field**
 - description/purpose**, 5–7
- Back button, 3–16, 7–18
- Background Color
 - button, 3–19, 7–9
 - screen, 7–7
- Background Task, description/purpose**, 4–1
- backing-up files, 2–6
- Bargraph button, 3–21, 7–36
- basic tags**
 - description/purpose**, 5–3
 - formatting, 5–3
- batch history file, 8–19, 8–22
- Bitmap Editor button, 3–19, 7–8
- Bitmap toolbar, 3–21, 7–22
- bitmaps, Windows CE
 - 16-color only, 7–4
- BlockUser function, A-94
- Boolean data type
 - using in class tags, 5–3
- Boolean data type
 - description/purpose, 5–6
- Boolean data type
 - using in tag fields, 5–7
- Boolean data type
 - selecting, 5–14
- borders, resizing**, 6–21
- Button button, 3–20, 7–28
- buttons**
 - .NET Control, 3–21, 7–100
 - ActiveX Control, 3–20, 7–93
 - Alarm, 7–110
 - Alarm/Event Control Object, 3–20
 - Alarm/Event Control Object button, 7–44
 - Align bottom, 3–18, 7–12
 - Align left, 3–18, 7–11
 - Align right, 3–18, 7–11
 - Align top, 3–18, 7–11
 - Back, 3–16, 7–18
 - Background Color, 3–19, 7–9
 - Bargraph, 3–21, 7–36
 - Bitmap Editor, 3–19, 7–8
 - Button, 3–20, 7–28
 - Center horizontally, 3–18, 7–12
 - Center vertically, 3–18, 7–12
 - Change colors, 7–22
 - Change Colors, 3–22
 - Close, 3–2
 - Closed Polygon, 3–19, 7–24
 - Colors, 3–21, 7–38
 - Combo-Box, 3–20, 7–106
 - Command, 3–21
 - common, 3–39
 - Context Sensitive Help, 3–15
 - Contract/Expand, 3–23
 - Contract/Expand, Output window, 3–24
 - Copy, 3–13
 - Cross Reference, 3–15, 3–48
 - Cut, 3–13
 - Database Spy Window, 3–14
 - Delete, 3–13
 - Dynamic, 3–21
 - Dynamic Properties, 3–21
 - Dynamic Rotation, 7–42
 - Ellipse, 3–20, 7–26
 - Erase Area, 3–22, 7–22
 - Evenly space horizontally, 3–18, 7–13
 - Evenly space vertically, 3–18, 7–13
 - Execution Environment, 3–16, 12–9
 - Fill Color, 3–19, 7–8
 - Flip horizontally, 3–18, 7–13
 - Flip vertically, 3–18, 7–14
 - Flood fill, 7–22
 - Flood Fill, 3–22
 - Fonts, 3–19, 7–8
 - Forward, 3–17, 7–18
 - Global Tags Replace, 3–16
 - Go, 3–17, 7–18
 - Grid, 3–19, 3–20, 7–9
 - Group, 3–18, 7–16
 - Hide Docked Window, 3–23, 3–24
 - Home, 3–17, 7–18
 - Hyperlink, 3–21, 7–35
 - IP Security, 7–20
 - Library, 3–14
 - Line, 3–19, 7–25
 - Line Color, 3–19, 7–9
 - List Box Object, 3–20
 - Main Password, 11–1
 - Maximize**, 3–2
 - Minimize**, 3–2
 - Move backward, 7–15
 - Move forward, 7–15
 - Move to back, 3–18, 7–15
 - Move to front, 3–18, 7–15
 - New, 3–12, 7–2
 - Object Finder, 3–15, **3–47**, 5–26
 - Open Polygon, 3–19, 7–24
 - Open Project, 3–12
 - Output Window, 3–14
 - Paste, 3–13
 - Pixel Editing, 3–22, 7–22
 - Position, 3–21, 7–40
 - Print, 3–14
 - property, 5–24
 - Pushbuttons, 3–20, 7–89
 - Rectangle, 3–20
 - Refresh, 3–17, 7–18
 - Resize, 3–21, 7–41
 - Resize height, 3–17, 7–10
 - Resize width, 3–18, 7–10
 - resizing windows**, 3–2
 - Rotate, 3–18, 7–14
 - Rounded Rectangle, 3–20, 7–26, 7–27
 - Run Application, 3–16, 12–9

- Run Application, 13–14
 - Save, 3–12, 3–52
 - Save All, 3–13, 3–52
 - Select Area, 3–22, 7–22
 - Select Transparent Color, 3–22, 7–22
 - Selection, 3–19, 7–8
 - Send project to target, 3–16, 12–9
 - Smart Message, 3–20
 - Smart Message Objects, 7–84
 - Stop, 3–17, 7–18
 - Stop Application, 3–16, 12–9
 - Stop display test, 3–16
 - Stop Test Display, 12–9
 - Tag Properties, 3–16
 - Tags Properties, 5–19
 - Test Display, 3–16, 12–9
 - Text, 3–20, 7–29
 - Text I/O, 3–21, 7–37
 - Toggle Transparent Color, 3–22, 7–23
 - Trend, 7–75
 - Undo, 3–14
 - Ungroup, 3–18
 - Workspace Window, 3–14
 - XRef, 3–48
 - X-Ref, 3–15
 - Zoom, 3–15
 - Zoom combo-box, 3–15
- C**
- Caution notes
 - application development and license levels, 6–3
 - configuring worksheets, 5–12
 - deleting tags, 5–26
 - description/purpose, xiii
 - hard disk performance, 5–20
 - making changes in screens saved as HTML, 7–18
 - modifying default settings, 4–5, 4–9**
 - reading saved .bmp file, 7–4
 - resizing a window, 7–4
 - using the Math() function, 4–10**
 - Center horizontally button, 3–18, 7–12
 - Center vertically button, 3–18, 7–12
 - CEServer.exe
 - default directory path, 2–21**
 - downloading, 2–21
 - running Remote Agent, 2–22, 14–1
 - CEView
 - downloading, 1–1, 2–1, 2–5, 14–4
 - installing, 2–1, 2–5
 - running applications, 1–1
 - runtime files, storing, 2–5, 2–8
 - sending email from, 1–3
 - updating, 1–1
 - Change Colors button, 3–22, 7–22
 - change the screen resolution, 3–42
 - changing object layers buttons, 7–14
 - CharToValue function, A-46
 - CharToValueW function, A-47
 - CheckESign function, A-95
 - class members
 - naming guidelines, 5–13
 - class tags, 5–17**
 - description/purpose, 3–8, 5–1, 5–3, 5–5**
 - Class worksheets*
 - configuring, 5–17*
 - Classes folder, 5–1, 5–17
 - ClassMembersToStrVector function, A-48
 - CleanReadQueue function, A-109
 - ClockGetDate function, A-73
 - ClockGetDayOfTheWeek function, A-74
 - ClockGetTime function, A-75
 - Close button, 3–2, 6–20
 - Close function, A-89, A-93
 - Closed Polygon button, 3–19, 7–24
 - CloseSplashWindow function, A-110
 - closing files, 3–39
 - closing projects, 3–38
 - closing Translation Editor, 16–12
 - CnfEmail function, A-242
 - CNFFtp function, A-276
 - color interface, 3–33
 - color selection, recommendation, 2–4
 - Colors button, 3–21, 7–38
 - Combo-Box button, 3–20, 7–106
 - Comm tab, 10–1
 - Command button, 3–21
 - command windows
 - accessing tag fields, 5–12
 - common buttons, 3–39
 - common errors, 18–5
 - communication tab
 - project settings, 6–23
 - communication, configuring, 10–1
 - communication, specifying for drivers/OPC, 6–13**
 - Communications tab, 3–10
 - Configure *dialog*, 6–5
 - configuring
 - alarm messages, 5–22
 - DDE, 10–27
 - Development Station, 14–3
 - event settings, 9–2
 - OPC, 10–20
 - projects, 6–1
 - security system, 11–1
 - tags, 5–26
 - TCP/IP, 10–25
 - unit properties, 5–19
 - worksheets, Class*, 5–17
 - worksheets, using tags, 5–1
 - constants
 - using for array tags, 5–4
 - contacting technical support, 18–2
 - Context Sensitive Help button, 3–15
 - Contract/Expand button, 3–23
 - controlling who can delete Alarm messages, 7–50
 - conventions
 - mouse/selection, xiii
 - text, xii
 - Windows, xiii
 - converting Trend History files
 - from binary to text, 8–21
 - from text to binary, 8–22
 - Copy button, 3–13
 - copying files to WinCE devices, 2–5
 - Cos function, A-84
 - Cot function, A-85
 - CreateUser function, A-96
 - creating
 - ActiveX objects, 3–3
 - basic tags, 5–3
 - creating class tags, 5–5
 - creating tags, 5–1
 - Cross Reference button, 3–15, 3–48
 - CSV Databases
 - importing from, 3–63
 - cursor display properties, 6–21
 - custom parameters, 10–25
 - customer support
 - contacting, xv
 - customer information form, xvi

Cut button, 3–13

D

data flow

runtime module, 4–1

Data Sources dialog

Trend Control Object, 7–53

data types

choosing, 5–6, 5–14

class tags, 5–3

using in tag fields, 5–7

database

batch history file, 8–22

database interface, 17–1

configuring a default database, 6–18, 17–6

configuring primary and secondary databases, 17–5

database providers, 17–2

Database Spy, 3–23

accessing tag fields, 5–12

description/purpose, 1–2, 4–1

Remote, 12–10

using for debugging, 12–2

Database Spy Window button, 3–14

Database tab, 3–8

database tags

deleting, 5–26

description/purpose, 4–1, 5–1

internal, 3–8, 5–2

read-only, 3–8, 5–2

sharing with PC-based control software

programs, 3–8, 5–1

using in applications, 5–24

using indirect tags, 5–6

database troubleshooting, 17–25

database, importing, 3–56

databases, 8–35

Datasheet View icon, 5–13, 5–14

datasheets

Application Tags, closing, 5–14

Application Tags, opening, 5–13

Application Tags, specifying maximum indexes, 5–4

using Tab key, 5–13

Date and Time functions, A-3, A-73

DateTime2Clock function, A-76

DB tabs, Database Spy window, 3–23

DB/ERP function, A-281

DB/ERP functions, A-11

DBCursorClose function, A-282

DBCursorCurrentRow function, A-283

DBCursorGetValue function, A-284

DBCursorMoveTo function, A-285

DBCursorNext function, A-286

DBCursorOpen function, A-287

DBCursorOpenSQL function, A-289

DBCursorPrevious function, A-290

DBCursorRowCount function, A-291

DBDelete function, A-292

DBExecute function, A-293

DBInsert function, A-294

DBSelect function, A-295

DBUpdate function, A-297

DbVersion function, A-176

DCOM Tutorial, 10–23

DDE

Client, configuring, 10–28

Client, description/purpose, 4–1

configuring, 10–27

protocol, 1–1

server, configuring, 10–29

Server, description/purpose, 4–1

Dead Band value, specifying, 5–20, 5–21

defaults

caution notes, 4–5, 4–9

changing how modules start, 6–31

development language, 6–11

screen resolution, 6–4

server name, 6–6

title bar text, 6–20

translation files, 6–11

defining users, 11–7

Delete button, 3–13

DeleteOlderFiles(strPath, strMask, strDate), A-139

Demo Mode

IWS, 2–11

development environment, description/purpose, 3–1

Development Station, configuring, 14–3

development system, description/purpose, 1–1

deviations

specifying

for tag fields, 5–22

specifying for tag fields, 5–10

DevM tag field, description/purpose, 5–11

DevMLimit tag field, description/purpose, 5–12

DevP tag field, description/purpose, 5–11

DevPLimit tag field, description/purpose, 5–12

DevSetLimit tag field, description/purpose, 5–12

DialError function, A-249

DialGetClientIP function, A-255

DialGetServerIP function, A-256

dialogs

Configure, 6–5

Hardkey Settings, 6–2

Insert Class, 5–17

New, 6–1

New Tag, 5–4

Object Properties, adding tags, 5–13, 5–24

Project Wizard, 6–3

Protection Manager, 6–2

Runtime menu options, 6–21

Softkey Settings, 6–2

Tag Properties, 5–19

DialStatus function, A-257

DialUp function, A-259

Dial-Up functions, A-9, A-248

DialUpToCE function, A-261

DirCreate(strDirectory, optBooFullPath), A-140

DirDelete(strDirectory, optBooEmptyOnly), A-141

DirLength(strPath), A-142

DirRename(strPath, strDirectoryFrom, strDirectoryTo), A-143

displaying a grid, 7–44

displays, testing, 3–50

Div function, A-15

downloading

applications, 1–1, 14–1

applications to target system, 6–29

CEView, 2–1, 14–4

considerations, losing tag properties, 5–12, 5–24

dragging

enabling/disabling, 3–26

Driver Runtime task

description/purpose, 4–1

Driver Users Guides, xv

Driver Worksheets, 10–10

drivers

available for IWS, 1–3

communicating with OPC, 6–13

- starting modules, 6–33
- stopping modules, 6–33
- drivers folder, 10–2
- debugging tools, 1–2
- Dynamic button, 3–21
- Dynamic Data Exchange. *See* DDE
- Dynamic Properties
 - button, 3–21
 - toolbar, 7–31
- Dynamic Rotation button, 7–42
- E**
- editing tags, 5–13
- Ellipse button, 3–20, 7–26
- email
 - functions, A-9**, A-242
 - sending from CEView, 1–3
- Enable Fire Compression check-box, 7–20
- enabling
 - alarms, specifying tag properties, 5–11, 5–19, 5–21
 - history logging, 5–19, 5–23
 - tags to display on HTML pages, 5–7
 - translation, alarm messages, 5–21
- EndTask function, A-111
- entering constants in tag/numeric fields, 7–36
- Erase Area button, 3–22, 7–22
- Evaluation Mode
 - IWS, 2–10
- Evenly space horizontally button, 3–18, 7–13
- Evenly space vertically button, 3–18, 7–13
- Event Logger functions, A-10**, A-274
- Event Settings, 9–1
- executables
 - CERasSvr.exe, A-262
 - CEServer.exe, 2–21, 2–23, 14–1
 - hst2txt.exe, 8–21
 - netdde.exe, 18–9
 - RunStartup.exe, xii
 - Setup.exe, 2–3
 - SKEYUPG.exe, 2–18
 - Studio Manager.exe**, 4–4
 - txt2hst.exe, 8–22
- ExecuteAlarmAck function, A-208
- executing
 - driver, 10–19
 - runtime module, 4–1
 - translation functions, 16–9
- Execution Control toolbar, 3–16, 12–9
- Execution Environment
 - button, 3–16, 12–9
 - dialog, 14–3
- execution fields, 5–12
 - accessing tag fields, 5–12
- Execution Modes for IWS, 2–10
- Execution Tasks tab, 6–31
- ExitWindows function, A-112
- Exp function, A-35
- experience requirements
 - working in a Windows™ environment, xii
- exporting
 - application screens, 1–2, 7–16
- Ext function, A-172
- Ext(strText), 16–11
- External Databases task, 8–35
- F**
- False function, A-38
- features, product, 1–2
- fields
 - accessible during runtime, 5–7
 - execution, 5–12
 - tag, saving changes, 5–20
- File functions, A-5**
- File menu**
 - description/purpose**, 3–3
 - options**, 3–3
- FileCopy(strSourceFile, strTargetFile, optnumTimeout), A-144
- FileDelete(strFile), A-145
- FileLength (strFile), A-146
- FileRename(strOldName, strNewName), A-146
- files
 - .app, 6–20
 - backing-up, 2–6
 - CEServer.exe, 2–21, 14–1
 - closing, 3–39
 - configuration, storing, 6–1
 - history (.hst), saving, 5–23, 6–13
 - opening, 3–38
 - SKEYUPG.exe, 2–18
 - specifying default translation, 6–11
 - specifying file lifetime**, 6–13
 - storing CEView runtime files, 2–1
 - storing, CEView runtime files, 2–5, 2–8
 - storing, Remote Agent program file, 2–21
 - Studio Manager.exe**, 4–4
 - translation, saving, 6–11
- Fill Color, 7–8
 - button, 3–19
- filtering alarm messages during runtime, 7–47
- filtering alarms
 - Dead Band value, 5–21
 - strings, 5–22
- FindAllDevices function, A-262
- FindFile(strFile, opttagFilesFound, optnumTimeout), A-147
- finding tags, screen objects, 3–45
- FindModem function, A-264
- FindPath(strPathName), A-149
- Flip horizontally button, 3–18, 7–13
- Flip vertically button, 3–18, 7–14
- Flood Fill button, 3–22, 7–22
- focus
 - screen editor, object properties window, 3–55
- folders
 - Application Tags, 5–1, 5–14
 - Classes, 5–1, 5–17
 - Shared Database, 5–1
 - System Tags, 5–1
 - Tag List*, 5–14
- font dialog, 7–45
- fonts, 3–31
- Fonts button, 3–19, 7–8, 7–83
- For & Next loop, A-212
- ForceTagChange function, A-209
- Format function, A-16
- formatting
 - formatting conventions, xii
- forms
 - customer information, xvi
- Forward button, 3–17, 7–18
- Frequently Asked Questions (FAQ), xv
- FTP functions, A-10**
- ftpGet function, A-277
- ftpPut (strLocalFile, strRemoteFile, numTransferType), A-278
- ftpStatus function, A-279
- function names syntax, A-12
- functions, A-1

- ActiveX, A-10**, A-271
- alarm, enabling, 5–21
- Arithmetic, A-1**, A-14
- Date and Time, A-3**, A-73
- DB/ERP, A-11**, A-281
- Dial-Up, A-9**, A-248
- Email, A-9**, A-242
- Event Logger, A-10**, A-274
- File, A-5**
- FTP, A-10**
- Graphic, A-6**, A-162
- IWS supervisory tasks, 3–8, 5–2
- Log Message, A-1**, A-13
- Logarithmic, A-35
- Logarithmic, A-1**
- Logical, A-2**, A-38
- Loop, A-8**, A-212
- Module Activity, A-4**, A-104
- Multimedia, A-6**, A-174
- ODBC, A-8**, A-213
- Opening and Closing Windows, A-3**
- Screen, A-89
- Security, A-4**, A-94
- Statistical, A-1**, A-30
- String, A-2**, A-42
- System Info., A-7**, A-175
- tag name syntax, 15–4
- Tags Database, A-8**, A-208
- Translation, A-6**, A-172
- Trigonometric, A-3**, A-81
- UNICODE conversion restrictions, A-46
- functions executed during runtime, 4–9
- File functions, A-138
- FTP functions, A-276
- G**
- GetAppHorizontalResolution function, A-176
- GetAppPath function, A-177
- GetAppVerticalResolution function, A-177
- GetBit function, A-20
- GetClock function, A-77
- GetComputerIP function, A-178
- GetComputerName function, A-178
- GetCursorX function, A-179
- GetCursorY function, A-179
- GetDisplayHorizontalResolution function, A-180
- GetDisplayVerticalResolution function, A-180
- GetFileAttributes(strFile), A-150
- GetFileTime(strFileName, numFormat), A-151
- GetFreeMemoryCE function, A-181
- GetHardKeyModel function, A-182
- GetHardkeySN function, A-183
- GetHstInfo(strFileName, numInfoType), A-152
- GetIPAll function, A-184
- GetLine(strFileName, strSeqChar, tagStore, optnumCase, optOverflowTag), **A-152, A-153**
- GetMemoryCE function, A-185
- GetNetMACID function, A-186
- GetOS function, A-187
- GetPrivateProfileString function, A-188
- GetProductPath function, A-187
- GetRegValue function, A-189
- GetRegValueType function, A-190
- GetScrInfo function, A-163**
- GetStatusSendEmailExt function, A-244
- GetTagValue function, A-210
- GetTickCount(), A-191
- Getting Started Guide, xv
- GetUserNames function, A-98
- GetUserPwdAging function, A-99
- GetUserState function, A-100
- Global Tags Replace, 3–48
- Global Tags Replace button, 3–16
- Go button, 3–17, 7–18
- go to option, 3–46
- Graphic functions, A-6**, A-162
- graphics, 3–75
- Graphics Script interface, 15–11
- Grid button, 3–19, 3–20, 7–9
- grid, displaying, 7–44
- Group Account dialog, 11–2, 11–4
- Group button, 3–18, 7–16
- group numbers, 5–22, 5–23
- Group, inserting, 11–3
- grouping tags, 5–5
- H**
- HangUp function, A-265
- hard disk cautions, 5–20
- Hardkey license, upgrading, 2–14
- Hardkey Settings *dialog*, 6–2
- hardkeys
 - installing, 2–12, 2–14
- Help, accessing, xii
- Hi tag field**, 5–22
 - description/purpose**, 5–11
- Hide check-box
 - Screen Attributes, 7–4
- Hide Docked Window button, 3–23, 3–24
- HiHi tag field**, 5–21
 - description/purpose**, 5–11
- HiHiLimit tag field**
 - description/purpose**, 5–11
- HiLimit tag field**
 - description/purpose**, 5–11
- Hint field, 3–26
- history
 - batch, 8–22
 - configuring properties, 5–23
 - enabling logging, 5–23
- history files (.hst)
 - saving, 5–23
 - saving/storing, 6–13
- HMI
 - downloading CEView, 1–1, 2–1, 2–5
 - pre-loaded software, 1–1, 2–5
- Home button, 3–17, 7–18
- Horizontal Scale button, 7–77
- Hour2Clock function, A-78
- HST2TXT function, 8–21
- HST2TXT(strStartDate, strStartTime, numDuration, numGroupNumber, optStrTargetFile, optStrSeparator, optnumMilliseconds, optStrFormat), A-155
- HST2TXTIsRunning(), A-157
- HTML format, saving screens, 7–18
- HTML pages
 - displaying tag values, 5–7
 - enabling tag display, 5–7
- Human Machine Interface. *See* HMI
- Hyperlink button, 3–21, 7–35
- I**
- icons
 - Add/Remove Programs, 2–7
 - applications, A-136
 - Datasheet View**, 5–13, 5–14

- IWS, 3–2
- IWS desktop, 2–1, 2–3
- IWS logo, 3–2
- IWS, desktop, 2–4
- launching InstallShield Wizard, 2–4
- Symbol Library, 1–2
- Tasks tab, 3–6
- TCP/IP Client module, 10–26
- TCP/IP Server module, 10–26
- Identification tab, description/purpose, 6–8
- If function, A-39
- Import Wizard, 3–56
- indexes, array
 - description/purpose, 5–4
 - specifying maximums, 5–4
- indirect tags**
 - description/purpose**, 5–3, 5–6
- InduSoft Web Studio. *See* IWS
- InduSoft Web Studio *Getting Started Guide*, xv
- InfoAppAlrDir(), A-192
- InfoAppHstDir(), A-192
- InfoDiskFree(strDisk), A-193
- InfoResources function, A-194
- Insert Class dialog*, 5–17
- insert screen, 7–3
- Insert Screen Group, 7–16
- inserting a Group, 11–3
- installing
 - CEView, 2–1, 2–5
 - IWS, 2–1
- InstallShield Wizard
 - launching, 2–4
- integer data type
 - description/purpose**, 5–6
 - selecting, 5–14
 - using in class tags, 5–3
 - using in tag fields, 5–7
- IntelliSense
 - VBScript, 15–33
- internal structure
 - description/purpose, 4–1
- invalid license
 - warning message, 2–10
- IP Security button, 7–20
- IsActiveXReg function, A-195
- IsScreenOpen function, A-113
- IsTaskRunning function, A-114
- IsViewerInFocus function, A-115
- IWS
 - available drivers, 1–3
 - closing**, 3–2
 - description/purpose, 1–1
 - development environment, description/purpose*, 3–1
 - development system, description/purpose, 1–1
 - Driver Users Guides*, xv
 - Execution Modes, 2–10
 - exiting**, 3–2
 - features, 1–2
 - Frequently Asked Questions (FAQ), xv
 - functions, A-1
 - Getting Started Guide*, xv
 - icons, desktop, 2–1, 2–3, 2–4, 3–2
 - icons, logo, 3–2
 - installing, 2–1
 - overview, 1–2
 - product limitations, 1–3
 - Protection Types, 2–12
 - removing program, 2–6
 - resizing screens**, 3–2
 - runtime, description/purpose, 1–1
 - uninstalling, 2–6
 - using the title bar**, 3–2
 - viewing status**, 3–2
- IWS Databases
 - importing from, 3–60
- IWS Scripting Language, 15–1
- K**
- keyboard shortcuts
 - cutting/pasting objects, 5–24
 - using Tab key, 5–13
- KeyPad function, A-116
- L**
- language reference**
- VBScript, 15–22**
- languages
 - default development, 6–11
- Legend dialog
 - Trend Control Object, 7–62
- Library
 - symbols, 7–130
- license levels
 - important notes, 6–3
 - specifying, 6–2
 - verifying, 6–2
- Licensed for Engineering + Runtime Mode, 2–11
- Licensed for Engineering Only Mode, 2–11
- Licensed for Runtime Only Mode, 2–11
- limitations, product, 1–3
- Line button, 3–19, 7–25
- Line Color button, 3–19, 7–9
- List Box Object button, 3–20, 7–81
- Lo tag field**, 5–22
 - description/purpose**, 5–11
- Log function, A-36
- Log Message functions, A-1**, A-13
- Log On dialog, 11–17
- Log10 function, A-37
- Logarithmic functions, A-1**, A-35
- logging
 - history, 5–23
 - on/off, 11–17
 - specifying Dead Band variation, 5–23
- Logic Operators, 15–3
- Logical functions, A-2**, A-38
- LogOff function, A-118
- Logon function, A-118
- LogWin
 - description/purpose**, 4–1
 - module, 12–7
 - module, description/purpose, 1–2
 - Remote, 12–11
- LoLimit tag field**
 - description/purpose**, 5–11
- LoLo tag field**, 5–22
 - description/purpose**, 5–11
- LoLoLimit tag field**
 - description/purpose**, 5–11
- Loop function, A-212
- Loop functions, A-8**
- LSB, accessing, A-25
- M**
- Main Driver Sheet, 10–17
- Main Password button, 11–1
- managing alarms remotely, 5–7, 5–21
- managing applications remotely, 5–7, 14–1

- manuals
 - Getting Started Guide*, xv
 - mouse/selection, xiii
 - Windows conventions, xiii
 - Math function, A-119
 - Math task, 8-31
 - Math Worksheet, 8-31
 - Math() function, caution note, 4-10**
 - Max function, A-31
 - Max tag field, 5-12**
 - description/purpose, 5-7**
 - exceeding specified value, 5-20
 - specifying values, 5-20
 - Maximize button, 3-2, 6-20**
 - maximum characters
 - tag and class member names, 5-13
 - menu
 - bar, using, 3-3
 - description/purpose, File, 3-3**
 - enabling/disabling, 6-21**
 - options, File, 3-3**
 - merging applications, 1-3
 - messages
 - alarm, creating, 5-22
 - button, 7-83
 - warning, enabling, 6-29
 - warning, values exceed Min/Max, 5-12, 5-20
 - written to Output window, 5-12, 5-20
 - Microsoft ActiveSync, 2-24
 - Min function, A-33
 - Min tag field, 5-12**
 - description/purpose, 5-7**
 - exceeding specified value, 5-20
 - specifying values, 5-20
 - Minimize button, 3-2, 6-20**
 - Minimum/Maximum
 - array tag sizes, 5-4
 - specifying number of characters, A-16
 - specifying tag values, 5-7
 - specifying worksheet values, 10-13
 - mnemonics
 - configuring for alarm states, 5-23
 - Mod function, A-21
 - Mode toolbar, 3-19, 7-8
 - modem
 - not supported for Pocket PC v3.00, 10-3, 10-8
 - modem connections
 - and drivers, 10-7
 - Module Activity functions, A-4, A-104**
 - modules, changing default start-up, 6-31
 - modules, runtime
 - data flow, 4-1
 - starting, 6-31
 - mouse/selection conventions, xiii
 - Move backward button, 7-15
 - Move forward button, 7-15
 - Move to back button, 3-18, 7-15
 - Move to front button, 3-18, 7-15
 - MSB, accessing, A-25
 - Multimedia functions, A-6, A-174**
- N**
- naming projects, 6-1
 - naming tags, 5-13, 5-14
 - NCopy function, A-49
 - NET Framework 1.1, 2-5, 17-13, 17-29*
 - NetDDE
 - protocol, 1-1
 - Network Dynamic Data Exchange. See NetDDE
 - New button, 3-12, 7-2
 - New dialog, 6-1
 - New Tag dialog, 5-4
 - New Value dialog, 12-3
 - Notes, description/purpose, xiii
 - Num function, A-50
- O**
- object alignment buttons, 7-11
 - Object Finder button, 3-15, 3-47, 5-26
 - object grouping and ungrouping buttons, 7-16
 - Object ID, 3-26
 - object library, description/purpose, 1-2
 - Object Properties dialogs*
 - adding tags, 5-13, 5-24*
 - object properties window
 - focus, 3-55
 - Object Size, 3-26
 - objects, screen
 - dragging, 3-26
 - specifying display properties, 6-21**
 - ODBC
 - database, and .NET Framework 1.1, 2-5, 17-29
 - database, importing from, 3-64
 - driver interface with IWS, 17-3
 - error codes in IWS, 8-30
 - functions, A-8, A-213**
 - protocol, 1-1
 - runtime task, description/purpose, 4-1**
 - runtime, description/purpose, 4-1**
 - runtime, TimeSlice settings, 4-8
 - task, 8-28
 - worksheet, 8-28
 - ODBC worksheet
 - changing data during runtime, 8-29
 - ODBCBeginTrans(numHandler), A-214
 - ODBCBindCol(numHandler, strColName, strColType, strTagName), A-215
 - ODBCCanAppend(numHandler), A-216
 - ODBCCanTransact(numHandler), A-217
 - ODBCCanUpdate(numHandler), A-218
 - ODBCClose(numHandler), A-219
 - ODBCCommitTrans(numHandler), A-220
 - ODBCDelete(numHandler), A-221
 - ODBCExecuteSQL(numHandler, strSqlCommand), A-222
 - ODBCInsert(numHandler), A-223
 - ODBCIsBOF(numHandler), A-224
 - ODBCIsDeleted(numHandler), A-225
 - ODBCIsEOF(numHandler), A-226
 - ODBCIsFieldNULL(numHandler, strColName), A-227
 - ODBCIsFieldNullable(numHandler, strColName), A-228
 - ODBCMove(numHandler, numOffset), A-229
 - ODBCMoveFirst(numHandler), A-230
 - ODBCMoveLast(numHandler), A-231
 - ODBCMoveNext(numHandler), A-232
 - ODBCMovePrev(numHandler), A-233
 - ODBCOpen(strDsn, strUser, strPassw, strTable, strFilter, strSort), A-234
 - ODBCQuery(numHandler), A-235
 - ODBCRollback(numHandler), A-236
 - ODBCSetFieldNull(numHandler, strColName, numValue), A-237
 - ODBCSetFilter(numHandler, strFilter), A-238
 - ODBCSetSort(numHandler, strSort), A-239
 - ODBCUnbindCol(numHandler, strColName), A-240
 - ODBCUpdate function, A-241

- OLE DB provider, 17–3
- OMRON CX Programmer Databases
 - importing from, 3–69
- OPC
 - Client, configuring, 10–20
 - Clients, description/purpose, 4–1**
 - communicating with target devices, 6–13
 - configuring, 10–20
 - protocol, 1–1
 - Server, configuring, 10–24
 - Servers, description/purpose, 4–1**
 - troubleshooting, 10–23
- OPC Server Databases
 - importing from, 3–62
- Open Connectivity. *See* OPC
- Open Database Connectivity. *See* ODBC
- Open function, A-90
- Open Polygon button, 3–19, 7–24
- Open Project button, 3–12
- opening
 - files, 3–38
 - projects, 3–38
- Opening and Closing Windows functions, A-3**
- operating systems
 - running IWS, 1–1
- option tab
 - project settings, 6–9
- Options dialog
 - Trend Control Object, 7–57
- Options tab
 - description/purpose, 6–9
- Output Window, 3–24
 - button, 3–14
 - messages, 5–12, 5–20
 - using for debugging, 12–4
 - XRef tab, 3–24
- P**
- PanelBuilder32™ Databases
 - importing from, 3–65
- PanelMate
 - importing from, 3–70
- Paste button, 3–13
- Paste Link, 7–131
- PC-based control programs
 - customized interface, 6–6
 - IWS integration, 1–3
 - modifying shared tags, 5–2
 - storing shared tags, 3–8
 - viewing shared tags, 5–1
- PDFCreate function, A-158
- Pen Style dialog
 - Trend Control Object, 7–56
- Pens button, 7–80
- performance considerations, Caution notes, 5–20
- PhoneDialUp function, A-266
- PhoneDisableListen function, A-267
- PhoneEnableListen function, A-268
- PhoneHangUp function, A-269
- PhoneStatus function, A-270
- Pi function, A-86
- pin button, 3–55
- Pixel Editing button, 3–22, 7–22
- Play(strFileName), A-174
- Points dialog
 - Trend Control Object, 7–55
- ports**
 - configuring, 2–6, 2–24**
 - installing hardkeys, 2–14
 - installing, hardkeys, 2–12
 - installing, softkeys, 2–24
 - removing hardkeys, 2–13
 - using TCP, 6–14
- Position button, 3–21, 7–40
- PostKey(numKeyDownOrKeyUp, numwParam, numlParam), A-120
- Pow function, A-22
- predefined tags, 3–8, 5–2
- Preferences tab, Project Settings menu
 - display warning message option, 6–29
- Print, A-160
- Print button, 3–14
- print preview, 3–54
- printing project screens, 3–53
- PrintSetup function, A-164
- PrintWindow function, A-165
- priority, specifying group priority, 5–22
- processor platforms
 - runtime provided, 2–5
 - supported, 2–1, 2–5
 - supported by CEView runtime, 2–1
- product
 - features, 1–2
 - limitations, 1–3
- ProductVersion function, A-196
- ProductVersion(), A-196
- programmable logic controller. *See* PLC programmable logic controller
- project settings
 - communication tab, 6–23
 - option tab, 6–9
- Project Status dialog, 10–19
- projects
 - closing, 3–38
 - configuring, 6–1
 - default location, 6–2
 - naming, 6–1
 - opening, 3–38
 - providing identifying information, 6–8
 - specifying screen resolution, 6–4
 - starting drivers, 6–33
 - starting tasks, 6–32
 - storing, 6–1
- properties
 - alarm, 5–21
 - tag, 5–12
 - tag, applying to system tags/downloading considerations, 5–24
 - tag, associating to objects, 5–24
 - tag, editing, 5–24
 - tag, specifying, 5–7
- property buttons, 5–24
- Protection Manager *dialog*, 6–2
- Protection Types, 2–12
- protocols
 - DDE, 1–1
 - NetDDE, 1–1
 - ODBC, 1–1
 - OPC, 1–1
 - TCP/IP, 1–1
- publications
 - related, xv
- Pushbuttons
 - button, 3–20, 7–89
 - styles, 7–92
- Q**
- quality tag, 5–8**
- field, description/purpose, 5–7**

R

- Rand function, A-34
 - rate of change, 5-20, 5-22
 - Rate tag field**, 5-22
 - description/purpose**, 5-11
 - RateLimit tag field**
 - description/purpose**, 5-12
 - RDFileN(tagSelectedFile, strSearchPath, strMask, numChangeDir), A-161
 - read-only
 - tags, 3-8, 5-2
 - real data type
 - description/purpose**, 5-6
 - selecting, 5-14
 - using in class tags, 5-3
 - using in tag fields, 5-7
 - Real numbers, storing, 5-6
 - Recipe Worksheet, 8-24
 - Recipe(strFunction), A-121
 - Recipes
 - task, 8-24
 - UNICODE support, 1-3
 - recommendations
 - linking WinCE devices, 2-6
 - video settings, 2-4
 - Rectangle button, 3-20, 7-27
 - referencing tags, 5-6
 - Refresh button, 3-17, 7-18
 - RegSaveCE function, A-197
 - related IWS publications, 18-3
 - related publications, xv
 - relational database
 - interface with IWS, 17-1
 - relational databases
 - SQL, 17-2
 - Remote Agent
 - dialog, 14-1
 - storing, program file, 2-21
 - remote alarm management
 - configuring tags, 5-7, 5-21
 - remote application management, 5-7
 - Remote Database Spy, 12-10
 - Remote LogWin, 12-11
 - Remote Security System, 11-14
 - Remote tools, 12-9
 - RemoveUser function, A-101
 - removing IWS, 2-6
 - replacing tags, 3-49
 - global, 3-48
 - Report function, A-122, A-125
 - Reports
 - task, 8-26
 - UNICODE support, 1-3
 - worksheet, 8-26
 - Reset Tags Database**, 5-15
 - ResetBit function, A-23
 - ResetDecimalPointsTable function, A-166
 - Resize**
 - button**, 3-2, 3-21, 7-41
 - height button, 3-17, 7-10
 - width button, 3-18, 7-10
 - resizing**
 - application screens, 6-21
 - borders**, 6-21
 - windows**, 3-2
 - resources, xv
 - restore defaults, 3-52
 - restrictions
 - naming tags, 5-13
 - tags, 5-12
 - worksheets, 5-12
 - Retentive Parameters, 5-20
 - Retentive Value, 5-20
 - RGBColor function**, A-167
 - RGBComponent function**, A-168
 - right-clicking*
 - Application Tags folder*, 5-14
 - tag icons, 5-25
 - tag names, 5-26
 - tag properties, 5-24
 - Rotate button, 3-18, 7-14
 - Round function, A-23
 - Rounded Rectangle button, 3-20, 7-26
 - RSLogix™ 5000 CSV Databases
 - importing from, 3-67
 - Run Application button, 3-16, 12-9
 - RunGlobalProcedureOnServer function, A-124
 - running applications, 3-52, 12-1
 - RunStartup.exe, xii
 - runtime
 - CEView, processor platforms supported, 2-1
 - data flow, 4-1
 - description/purpose, 1-1
 - Driver Runtime task, description/purpose**, 4-1
 - executing, 4-1
 - fields, accessible during, 5-7
 - functions executed during, 4-9
 - initial address limits, 10-13
 - menu options dialog*, 6-21
 - modules, starting, 6-31
 - ODBC, description/purpose**, 4-1
 - ODBC, TimeSlice settings, 4-8
 - processor platforms, 2-1, 2-5
 - properties, Screen Attributes, 7-4
 - tasks, descriptions/purposes**, 4-1
 - Runtime Desktop tab, description/purpose, 6-19
- S**
- sample applications, 6-4
 - Save All button, 3-13, 3-52
 - Save button, 3-12, 3-52
 - saving
 - history files, 17-1, 17-4
 - screens in HTML format, 7-18
 - translation files, 6-11
 - Translation worksheets, 16-8
 - Scheduler task, 8-33
 - Scheduler Worksheet, 8-33
 - Screen Attributes dialog, 7-3
 - screen background color, 7-7
 - Screen Coordinates field, 3-26
 - screen editor
 - focus, 3-55
 - Screen functions, A-89
 - screen objects
 - associating tags, 5-24
 - screen resolution, changing, 3-42
 - screens, application
 - enabling/disabling dragging, 3-26
 - exporting, 1-2, 7-16
 - resizing**, 3-2
 - scaling, 6-21
 - specifying startup screen**, 6-21
 - status bar, 3-26
 - viewing coordinates, 3-26
 - Scroll bars
 - Database Spy window, 3-23
 - Output window, 3-24

- searching Translation worksheets, 16–8
- secondary data server
 - settings, A-204
- secondary URL
 - settings, A-204
- security
 - field, Screen Attributes, 7–4
 - functions, A-4**, A-94
 - providing, 1–3
 - system access level, 11–16
- Security System dialog, 11–1
- Select Area button, 3–22, 7–22
- Select Transparent Color button, 3–22, 7–22
- Selection button, 3–19, 7–8
- Send project to target button, 3–16, 12–9
- SendEMail(strSubject, strMessage, strTO), A-244
- SendEMailExt function, A-246
- SendEvent function, A-274
- SendKeyObject function, A-126
- Serial Advanced Settings dialog, 14–2
- serial encapsulation tests
 - and drivers, 10–9
- serial links, downloading software to HMI, 2–1
- servers
 - changing default, 6–6
- SetAppAlarmPath function, A-199
- SetAppAlarmPath(strPath), A-198
- SetAppHSTPath function, A-200
- SetAppPath function, A-128
- SetBit function, A-25
- SetDateFormat(strSeparator, strFomat), A-201
- SetDecimalPoints function, A-169**
- SetDisplayUnit function, A-170**
- SetPassword function, A-102
- SetRegValue function, A-202
- SetsystemDate function, A-79
- SetSystemTime function, A-80
- SetTagDisplayUnit function, A-171**
- SetTagValue function, A-211
- setting tag properties, 5–19
- settings, Event, 9–1
- SetTranslationFile function, A-173
- SetTranslationFile() function, 16–9
- SetViewerInFocus function, A-129, A-130
- SetViewerPos function, A-131
- SetWebConfig function, A-176, A-204
- Shared Database**
 - tags, description/purpose, 3–8**, 5–1
 - tags, modifying, 5–2, 6–5
- Shared Database folder, 5–1
- Shared Image check-box, 7–3
- sharing tags, 5–7, 5–14, 6–4
- ShutDown function, A-132
- Sin function, A-87
- Size tag field, description/purpose, 5–7**
- SKEYUPG.exe, 2–18
- Smart Message button, 3–20
- Smart Message Objects button, 7–84
- smoothing, specifying for tags, 5–20
- SNMPGet function, A-206
- Softkey Settings *dialog*, 6–2
- softkeys, installing, 2–24
- SPC button, 7–81
- specifying
 - default translation files, 6–11
- specifying
 - license levels, 6–2
 - project settings, 6–7
 - screen resolution, 6–4
- specifying**
 - driver/OPC communication, 6–13**
- specifying**
 - properties, screen objects, 6–21**
- SQL
 - relational databases, 17–2
- SQRT function, A-26
- Standard Driver Worksheet, 10–10
- Standard toolbar, 3–12
- starting modules, 6–31
- StartTask function, A-133
- startup screens, specifying, 6–21**
- Startup Script interface, 15–19
- Startup value, 5–20
- Static Objects toolbar, 3–19, 7–24
- Statistical functions, A-1**, A-30
- status bar
 - controlling display, 6–21**
 - tips, 3–5
 - using, 3–26
- stop application, 12–1
- Stop Application button, 3–16, 12–9
- Stop button, 3–17, 7–18
- stop display test, 12–1
- Stop display test button, 3–16
- Stop Test Display button, 12–9
- storing
 - alarm history files, 6–13
 - CEView runtime files, 2–1, 2–5, 2–8
 - configuration files, 6–1
 - data, 1–1
 - projects, 6–1
 - Real numbers, 5–6
 - Remote Agent program file, 2–21
 - shared tags, 3–8
 - symbols, 7–130
 - tag values, 5–1
 - tag values, 4–1
 - trend curves, 3–9
- Str function, A-51
- Str2Asc function, A-52
- StrCompare function, A-53
- StrCompareNoCase function, A-54
- StrFromInt function, A-55
- StrFromReal function, A-56
- StrFromTime function, A-57
- string data type
 - description/purpose, 5–6**
 - selecting, 5–14
 - using in class tags, 5–3
 - using in tag fields, 5–7
- String functions, A-2**, A-42
- string, creating messages, 5–22
- StrLeft function, A-59, A-60
- StrLen function, A-61
- StrLower function, A-62
- StrRChr function, A-63
- StrRight function, A-64
- StrStr function, A-65, A-66
- StrStrPos function, A-67
- StrTrim function, A-68
- StrTrimAll function, A-69
- structure, internal, description/purpose, 4–1
- StrUpper function, A-70
- Studio Manager.exe, 4–4**
- Supervisory Control And Data Acquisition. *See* SCADA
- supervisory tasks, 3–8, 5–2
- support

- contacting, xv
- customer information form, xvi
- supporting
 - UNICODE conversion restrictions, A-46
 - UNICODE tag data types, 5-6
 - UNICODE, Reports, 1-3
- Swap16 function, A-27
- Swap32 function, A-28
- Symbol Library, 7-130
 - icon, 1-2
- SyncAlarm function, A-298
- SyncAlarmStatus function, A-301
- SyncEvent function, A-299
- SyncEventStatus function, A-302
- SyncTrend function, A-300
- SyncTrendStatus function, A-303
- syntax
 - function names, A-12
 - function tag names, 15-4
 - samples, tag fields, 5-12
- System Info. functions, A-7**, A-175
- system information, 3-44
- system load, specifying Startup Value, 5-20
- system tags**
 - description/purpose**, 3-8, 5-2
 - downloading considerations, 5-12, 5-24
 - viewing, 5-2
- System Tags folder, 5-1
- T**
- Tab key shortcut, 5-13
- tabs
 - icons, 3-6
 - Identification, description/purpose, 6-8
 - Options, description/purpose, 6-9
 - Runtime Desktop, description/purpose, 6-19
- tag fields
 - Retentive Parameters, 5-20
 - saving changes, 5-20
 - specifying deviations +/-, 5-10
 - using tag data types, 5-7
- Tag List folder*, 5-14
- tag names
 - maximum characters, 5-13
 - syntax, 15-4
- Tag Properties
 - button, 3-16
 - dialog*, 5-19
 - toolbar, 3-15, 3-46
- tag types**, 5-3
- Tagname text box, 3-15
- tags**
 - adding Application Tags datasheet, 5-13
 - adding on-the-fly, 5-14
 - application, description/purpose**, 3-8
 - array, specifying maximum index*, 5-14
 - class, description/purpose**, 3-8
 - data types, Boolean, integer, real, string, 5-6
 - data types, selecting, 5-14
 - deleting, 5-26
 - description/purpose*, 5-1, 15-1
 - displaying on HTML pages, 5-7
 - editing, 5-13
 - enabling alarms, 5-21
 - finding, 3-45
 - grouping into classes, 5-5
 - history properties, 5-23
 - internal, description/purpose**, 3-8
 - internal, downloading considerations, 5-12, 5-24
 - Local and Server parameters, 5-7
 - naming, 5-13, 5-14
 - properties, description/purpose, 5-7
 - quality, 6-21
 - read-only, 3-8
 - replace, 3-49
 - saving Retentive Value parameter, 5-20
 - sharing with Web Thin Clients, 5-7
 - specifying parameters, 5-20
 - specifying quality, 5-8, 12-2
 - storing values, 4-1, 5-1
 - using, 15-1
 - using *Object Properties* dialogs, associating tags, 5-24
- Tags Database*
 - accessing, 15-2
 - description/purpose*, 4-1, **5-1**
 - editing tags, 5-24
 - saving, 5-14
 - tag categories, description/purpose, 5-1
 - tags, adding/creating, 5-1
- Tags Database function, A-208
- Tags Database functions, A-8**
- Tags Properties button, 5-19
- Tan function, A-88
- target devices, specifying, 2-24
- Task Worksheets, configuring, 8-1
- taskbars, display properties, 6-21
- tasks
 - supervisory, 5-2
- tasks, 3-76
 - supervisory, 3-8
- tasks
 - External Databases, 8-35
- Tasks tab, 3-9
 - icons, 3-6
- TCP/IP
 - Client module icon, 10-26
 - Client module, description/purpose**, 4-1
 - Clients, specifying TCP port, 6-14
 - configuring, 10-25
 - links, downloading software to HMI, 2-1
 - ports, specifying, 6-14
 - protocol, 1-1
 - Server module icon, 10-26
 - Servers, specifying TCP port, 6-14
- TCP/IP and UDP/IP encapsulation
 - and drivers, 10-5
- TCP/IP Server module
 - description/purpose, 4-1
- technical support, 18-2
 - contacting, xv
 - customer information form, xvi
- templates, project
 - description/purpose, 6-4
- Test Display button, 3-16, 12-9
- test displays, 3-50, 12-1
- testing your application in a Web browser, 13-14
- testing/debugging your application, 12-1
- text box
 - Address, 3-17
 - Tagname, 3-15
- Text button, 3-20, 7-29
- text conventions, xii
 - formatting, xii
- Text I/O button, 3-21, 7-37
- TimeSlice settings, 4-8
- TimeStamp tag field**
 - description/purpose**, 5-7
- Tips

- description/purpose, xiii
 - using Status bar, 3–5
 - video settings, 2–4, 3–8, 3–31, 6–29**
 - title bar
 - changing default text, 6–20
 - enabling/disabling, 6–20**
 - using, 3–2
 - Toggle function, A-40
 - Toggle Transparent Color button, 3–22, 7–23
 - Toolbar dialog
 - Trend Control Object, 7–63
 - toolbars
 - Active Objects, 3–20, 7–44
 - Align and Distribute, 3–17, 7–10
 - Bitmap, 3–21, 7–22
 - description/purpose, 3–5
 - docking, 3–5
 - Dynamic Properties, 7–31
 - Execution Control, 3–16, 12–9
 - Mode, 3–19, 7–8
 - Standard, 3–12
 - Static Objects, 3–19, 7–24
 - Tag Properties, 3–15, 3–46
 - Web, 3–16, 7–18
 - ToolTips, 6–21
 - Trace function, A-13
 - translating
 - alarm messages, 5–21
 - applications, 1–2, 6–11
 - translation
 - files, specifying default, 6–11
 - functions, A-6**
 - functions, A-172
 - functions, executing, 16–9
 - Translation Editor, 16–4
 - Translation Editor utility
 - enabling translation, 6–11
 - Translation Tool, 16–1
 - Transmission Control Protocol/Internet Protocol. *See* TCP/IP
 - Trend
 - button, 7–75
 - worksheet, 8–14
 - worksheet and batch history file, 8–19
 - worksheets, configuring, 5–23*
 - worksheets, restrictions, 5–12
 - Trend Control Development Interfaces, 7–52
 - Trend Control object, 7–51
 - Trend Control Runtime Interface, 7–67
 - Trend task, 8–14
 - trends, storing trend curves, 3–9
 - trigger field, 7–76
 - Trigonometric functions, A-3, A-81**
 - troubleshooting, 18–1
 - database, 17–25
 - True function, A-41
 - Trunc function, A-29
 - TwinCAT PLC Databases
 - importing from, 3–71
- U**
- UnblockUser function, A-103
 - Undo button, 3–14
 - Ungroup button, 3–18
 - UNICODE support
 - IWS functions, conversion restrictions, A-46
 - Recipes, 1–3
 - Reports, 1–3
 - tag data types, 5–6
 - uninstalling IWS, 2–6
- Unit tag field**
- configuring, 5–20
 - description/purpose, 5–7**
 - updating
 - CEView, 1–1
 - User Account dialog, 11–7
 - User Guides
 - Driver, xv
 - user is blocked, 11–14
 - users, defining, 11–7
 - Using the Screen/Worksheet Editor, 3–11
- V**
- value
 - display previous value, 7–38, 7–46
 - ValueToChar function, A-71
 - ValueWToChar function, A-72
 - VBA
 - compared to VBScript, 15–35
 - VBScript, 15–5
 - and CEView, 15–39
 - and conditional statements, 15–42
 - and InputBox(), 15–37
 - and MsgBox(), 15–37
 - and NOT, 15–37
 - and True/False, 15–38
 - compared to VBA, 15–35
 - constants, 15–41
 - Editor, 15–33
 - Graphics Script interface, 15–12
 - language reference, 15–22**
 - looping statements, 15–44
 - operators, 15–41
 - procedures, 15–47
 - screen events, 15–37
 - support for ActiveX objects, 15–37
 - variables, 15–40
 - verifying
 - application, 3–51, 18–4
 - license levels, 6–2
 - TCP/IP and background tasks are running, 13–15
 - Vertical Scale button, 7–79
 - video settings**
 - recommendations, 2–4
 - tips, 2–4, 3–8, 3–31, 6–29**
 - view screen attributes, 7–3
 - Viewer window**
 - description/purpose, 4–1**
 - enabling/disabling features, 6–20*
 - ViewerPostMessage function, A-134
 - viewing
 - alarm properties, 5–19, 5–21
 - applications remotely, 5–7
 - history properties, 5–23
 - tag lists, 5–2, 5–25
 - tag values, 5–14
 - virtual keyboard, 3–30
 - Virtual Keyboard, 7–20, 7–38, 7–51, 11–1, A-129
 - Visual Script Language (VBScript), 15–5
- W**
- Wait(numMillisec), A-135
 - warning messages, enabling, 6–29
 - Web client, debugging, 12–8
 - Web site, InduSoft, xv
 - Web solution, 13–1
 - Web Thin Clients
 - parameters, A-204
 - tags, 5–7, 5–14
 - Web toolbar, 3–16, 7–18
 - web tunneling

- parameters, A-204
 - web tunnelling, 13-1
 - WinCE
 - copying files, 2-5
 - windows
 - CE applications, downloading CEView, 2-1
 - commands
 - Cut/Copy/Paste, 5-24
 - conventions, xiii
 - dockable, xiv
 - experience requirements, xii
 - Help, xii
 - resizing**, 3-2
 - running IWS, 1-1, 4-1
 - types, xiii
 - Viewer, enabling/disabling features*, 6-20
 - Winexec(strCommand, optnumState), A-136, A-137
 - worksheets
 - Class, configuring*, 5-17
 - Driver, 10-10
 - group numbers, 5-22
 - Main Driver, 10-17
 - Math, 8-31
 - ODBC, 8-28
 - Recipe, 8-24
 - Report, 8-26
 - restrictions, 5-12
 - Scheduler, 8-33
 - Standard Driver, 10-10
 - Trend, 8-14
 - Trend, configuring*, 5-23
 - using Tab key, 5-13
 - using tags, 5-1, 15-1
 - Workspace Window button, 3-14
 - WritePrivateProfileString function, A-207
- X**
- XGet function, A-271
 - X-Ref button, 3-15
 - XRef tab, Output window, 3-24
 - XRun function, A-272
 - XSet function, A-273
- Z**
- Zoom button, 3-15
 - Zoom combo-box button, 3-15