Introduction to InduSoft Web Studio

Training Course

IWS 6.1 SP2

Web-Ready InduSoft Tools for Automation

www.InduSoft.com
877.463.8763

PN: IND-TG/TG-001

Contents

Chapter 1. Welcome!

InduSoft Web Studio® (or IWS) is a powerful, integrated tool that exploits key features of Microsoft® Windows® NT/2000/XP and Windows® CE and enables you to build full-featured HMI (Human-Machine Interface) or SCADA (Supervisory Control and Data Acquisition) applications for your Industrial Automation business.

This introductory training course was designed to help you become familiar with InduSoft Web Studio and CEView. In this class, you will learn all the basic functionality of IWS, including an overview of the underlying architecture.

Course Objectives

Upon completion of this course, you will understand the basic features and functionality of InduSoft Web Studio, and you will be able to develop HMI and SCADA applications using the software.

Course Overview

This *Introduction to InduSoft Web Studio* course consists of the following modules:

DAY 1:

Welcome

Instructor introduction

Classroom/Building notes

Course Objectives and Overview

Additional Resources (online, manuals, technical support, and so forth)

What is InduSoft Web Studio?: Overview of InduSoft Web Studio features and functionality, the development and run-time environments, internal structure, and *Tags* database.

Installing InduSoft Web Studio: Discussion and hands-on exercise where students install and start IWS.

Creating Projects: Discussion and hands-on exercise where students create a new project, configure project settings, and specify project status.

Creating and Editing Tags: Discussion of tag types (class, array, pointer, internal, application, and shared tags) and hands-on exercise where students create the initial *Tags* database for their project.

Creating Screens: Discussion and hands-on exercise where students create the *Standard* and *Main* screens for their project (includes working with new 3-D graphics functionality).

DAY 2:

Working with IWS Expressions, Functions, and Scripting Language: Description of IWS math expressions, available functions, and scripting language syntax.

Configuring the Worksheets: Discussion and hands-on exercise where students configure *Math* and *Scheduler* worksheets.

Creating Recipes and Reports: Discussion and hands-on exercise where students create *Recipe* groups and reports for display in their applications or a World Wide Web (Web) browser.

Using the Translation Tool: Discussion and hands-on exercise where students use the IWS *Translation Tool* to create *Translation* worksheets and screens to enable automatic language translation at runtime.

Configuring the Security System: Discussion and hands-on exercise where students learn how to provide multi-level security for their Internet/intranet applications, and how to create alarms and send them to various utilities such as screens, e-mail, Web browsers, PDAs, and archives.

DAY 3:

Creating Trends: Discussion and hands-on exercise where students learn to keep track of process behaviors online or through historical trending and display the results on screens or a Web browser.

Communicating between Devices: Discussion and hands-on exercise where students configure IWS to communicate with PLC drivers, TCP/IP, OPC (OLE for process control), and Web devices.

Importing and Exporting Data in XML Format: Discussion and hands-on exercise where students learn how to import and export real-time data in XML format using the recipes module.

Working with ActiveX Objects: Discussion and hands-on exercise where students register an *ActiveX* object and configure it on the screen.

Running Web-Based Applications: Discussion and hands-on exercise where students learn how to configure their applications to run on the Web by saving them in HTML format and then exporting them to a browser.

Managing Applications Remotely: Discussion and hands-on exercise where students learn to configure and debug applications remotely using a TCP/IP link. Includes a discussion of the IWS development support tools (message register, error codes, event codes, *Database Spy*, and *LogWin*).

Additional Resources

If you require assistance while using InduSoft Web Studio, the following resources are available:

- **Related Publications**: The following publications provide additional information about InduSoft Web Studio and can be found in the *Documentation* folder on the IWS CD-ROM or can be downloaded from the InduSoft website at www.InduSoft.com:
- **InduSoft Web Studio Getting Started Guide.** Designed for first-time users, this publication contains information about the basic functions of InduSoft Web Studio.
- **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.
- **InduSoft Web Studio VBScript Reference Manual.** Describes the VBScript language and how VBScript can be used with InduSoft Web Studio.
- **Driver User Guides.** A Driver User Guide is provided for each InduSoft driver, explaining how to configure the driver according to the protocol characteristics/specification. The Driver User Guides are located 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.
- **Online Help.** Indusoft Web Studio provides a context-sensitive on-line help system that can be accessed in the Help menu located on the main menu bar.
- **Sample Applications.** The IWS CD-ROM includes two sample applications (NTDemo and CEDemo) that can be found in the DEMO subdirectory of the InduSoft Web Studio folder. Additional sample applications can be found on InduSoft's website at www.InduSoft.com.
- **Training Material Disk.** A disk that includes sample applications is provided with this Training Course.
- **IWS Advanced Training Class.**
- **InduSoft User Forum.** This is an on-line community forum that is located at http://indusoft.infopop.cc/eve. You will need to register and receive a password to access this forum.
- **Your Automation Supplier.** Depending on where you purchased InduSoft Web Studio (i.e. from an InduSoft distributor or OEM partner), they can provide additional technical support.
- **Customer/Technical Support**: Contact your InduSoft Technical Support Representative by email at support@indusoft.com or in the US or Canada by telephone at 877-INDUSOF (877-463-8763). Outside of the US or Canada, our support telephone number is (US) 513-349- 0334.
- **InduSoft Web Site**: Visit www.InduSoft.com.

Chapter 2. InduSoft Web Studio Product Overview

InduSoft Web Studio (IWS) is a powerful software product specifically designed to develop applications used in process supervision, automation, and control. IWS applications span the globe in a multitude of vertical markets including:

- Automotive
- Biomedical manufacturing
- Building automation
- Chemical
- Fabric manufacturing
- Food processing
- Glass manufacturing
- Machine manufacturers
- Oil and gas
- Pharmaceutical
- Pulp and paper
- Steel
- Transportation
- Utilities
- Water and wastewater
- And others ...

The flexibility built into InduSoft Web Studio allows you to design and implement applications for:

- Human-Machine Interfaces (HMI)
- Supervisory Control (Local and Remote) with Server and Database Redundancy capability
- Real-Time Performance Monitoring
- Process or Plant Floor data collection
- Data communications with corporate systems (databases)
- Data concentrators for distributed processes

InduSoft Web Studio is extremely scalable. Applications range from low-end Windows CE-based devices used for operator display, to higher end PC-based HMI's, up to high-end SCADA systems running on Windows Server 2003 and capable of supporting Server, Database and PLC redundancy. To develop these applications, InduSoft Web Studio provides **built-in** support for:

- One development platform that supports entire range of target platforms
- Efficient code generation to allow running on low-end Window CE-based devices
- Remote management
- Server-Client architecture (ability to support redundant IWS Servers)
- Redundant databases
- Run-time screen translation (Unicode-compliant)
- Web Thin Clients (up to 256 total Web Thin Clients, limited to 8 for Windows CE)
- Wide range of tools for Graphics generation, Alarms, Events, Security, Trending, Recipe Management, Report Generation, and Scheduler.
- Over 200 native drivers available, plus support for OPC Client/Server.
- ActiveX and .NET container (can add 3rd party programs/controls)
- Application importation for Rockwell PanelView and Eaton/Cutler-Hammer & Schneider Electric PanelMate products (separately licensed)

IWS provides a wide number of Tool and Connectivity options to let you build most any application. These Tools and Connectivity options are all based on Microsoft Windows standards (Windows CE, XP, 2000, NT, Server 2003 and Vista) and other industry standards, including:

- Microsoft standards: COM, DCOM, DDE, NetDDE, ADO, ADO.NET, ActiveX, and .NET
- Industry Standards: ODBC, XML, TCP/IP, OPC, SOAP, 21 CFR Part 11
- Scripting support: IWS built-in language and VBScript (Server, Thin Client and Windows CE)
- Application debugging tools: Database Spy, LogWin, Cross Reference
- Driver and Database toolkits (separately licensed)

The InduSoft Web Studio product consists of two parts:

A **development environment** that runs on a desktop, laptop, or industrial PC running Windows XP, 2000, Server 2003, Vista or NT.

The **run-time environment** that runs on an PC, operator interface workstation or other device running Windows NT/2000/XP or Windows CE.

Chapter 3. InduSoft Web Studio Configurations

InduSoft Web Studio (IWS) is a licensed software product that is available in a number of configurations to meet your application requirements. As shown in Figure 2-1, the two version of InduSoft software are InduSoft Web Studio and CEView, each available in various maximum tag count configurations. Depending on the tag count configuration, a specified maximum number of concurrent drivers will be supported.

InduSoft Web Studio

InduSoft Web Studio (IWS) is InduSoft's full-featured software product with the following features:

- All development is done on a Windows XP, 2000, Server 2003, Vista or Windows NT-based PC
- Applications can be developed to run on Windows XP, 2000, Server 2003, Vista or Windows NT-based PCs (or devices), or on Windows CE-based HMIs (or devices).
- IWS is a licensed product that uses a key-based protection system. The protection key can be either a HardKey (USB or Parallel Port) or a SoftKey (license file)
- IWS licenses can be purchased as development licenses or runtime-only licenses
  - A development license includes both the development (engineering) license and one runtime license.
  - The development (engineering) license and the runtime license can be placed on the same key (HardKey or SoftKey) or can be placed on separate keys.
  - Without a runtime license key, a runtime license will only work for 2 hours before automatically shutting down (demo mode).
- IWS is licensed based on the maximum number of tags supported. Based on the development license version purchased, a specifed number of concurrent drivers will be supported.
- Runtime applications can be developed using up to the maximum number of tags supported by the development (engineering) license. If the runtime can use a smaller tag count license version, then additional runtime licenses for the application can be purchased that use the smaller tag count size.
- With any runtime license, one Web Thin Client Session is supported. If additional concurrent Web Thin Client Sessions are required, they must be licensed separately.

CEView

CEView is a license-limited version of InduSoft Web Studio targeted for developers who will only use IWS to develop Windows CE-based applications. With CEView, the following features apply:

- All development is done on a Windows XP, 2000, Server 2003, Vista or Windows NT-based PC
- Applications run on Windows CE-based HMIs (or devices). CEView supports a wide range of Windows CE platforms, including x-86, XScale, ARM, MIPS and SH3/SH4 processor architectures.
- CEView licenses can be purchased as development licenses or runtime-only licenses
  - Development licenses include both the development (engineering) license and one runtime license.
  - The development (engineering) and runtime licenses must use separate license keys. The Windows CE runtime license must be SoftKey license-based only. The development (engineering) license can be either hardkey (USB or Parallel Port) or SoftKey based.
  - Without a runtime license key, a runtime license will only work for 2 hours before automatically shutting down (demo mode).
- CEView is licensed based on the maximum number of tags supported. (see Figure 2-1 for tag size versions). Based on the development license version purchased, a specifed number of concurrent drivers will be supported.
- Runtime applications can be developed using up to the maximum number of tags supported by the development (engineering) license. If the runtime can use a smaller tag count license version, then additional runtime licenses for the application can be purchased that use the smaller tag count size.
- With any runtime license, one Web Thin Client Session is supported. If additional concurrent Web Thin Client Sessions are required, they must be licensed separately.

CEView Differences and Limitations

CEView supports most of the functions of IWS. There are a few exceptions, as noted below:

- CEView will only generate runtime applications for Windows CE-based HMIs (devices)
- The maximum tag size license for CEView is 4,000 tags.
- The Picture Paste Link function is not supported in CEView
- Screen Groups (and HTML Screen Groups) are not supported in CEView
- ODBC and DDE are not supported in CEView (Microsoft does not support ODBC or DDE under Windows CE)
- Separate InduSoft Web Server and support files (ISSymbol) are provided for CEView (CPU architecture dependent).
- ActiveX Controls and .NET Controls developed for Windows XP may not work under Windows CE
- ADO.NET for Windows CE requires either the use of InduSoft's Database Gateway or ADOCE (Microsoft ADO.NET class library for Windows CE).
- Certain IWS built-in functions are not available for CEView.
- Certain Operating System support capabilities may not be available under Windows CE (Microsoft limitation) such as FileSystemObject (for VBScript), WMI (Windows Management Instrumentation), etc.
- **Important Note:** Effective use of the CEView runtime requires the Windows CE operating system to support key capabilities such as COM and DCOM. Additional items such as a registry save tool are useful and the implementation may vary from device to device depending on how the manufacturer implements the Windows CE operating system. Pre-installation by the manufacturer of software components such as VBScript.DLL (VBScript runtime library), CEServer (IWS remote management tool), ISSymbol.ocx (for Web Client Support) and a Registry save tool greatly simplify development of CEView/Windows CE application development. Check with your manufacturer to determine if these features are supported.

Choice of HardKey or SoftKey Protection

There is no right or wrong choice to the type of key (HardKey or SoftKey) used for licensing. Some customers prefer HardKey because of the portability, others prefer the SoftKey since it is embedded with the product. Some considerations for the type of key to use in your system include hardware swapping (in case of a hardware failure) and the ability to secure a HardKey.

License Upgrades

IWS and CEView licenses are field upgradeable, regardless of whether the license is SoftKey or HardKey based. Field upgrades are used to upgrade the license to a new version of software, support a larger tag count configuration or add additional Web Thin Clients. License files do not need to be upgraded to support Service Pack upgrades that use the same version of software (e.g IWS v6.1 SP1 → IWS v6.1 SP2).

Web Thin Clients

Web Thin Clients do not need to be licensed by InduSoft. Only an operating system (Windows XP (etc.) or Windows CE) with a browser such as Microsoft Internet Explorer is required. A runtime license from InduSoft is required for the Server to support the Web Thin Client. If additional Web Thin Clients operate concurrently, support for the additional Web Thin Clients must be licensed separately.

Product Types

InduSoft Product Types for Version 6.1 Service Pack 2								
	CEView Lite	*Lite Interface*	*CEView Standard*	*Local Interface*	*CEView Professional*	*Operator Workstation*	*Control Room*	*Advanced Server*
RunTime Operating System	CE	XP	CE	XP	CE	XP	XP	XP
Development System * includes 1 runtime + 1 web session	XP	XP	XP	XP	XP	XP		
Simultaneous Communication Drivers	1	1	3	3	5	5	8	Unlimited
Application tags (scripting + I/O points)	150	150	1,500	1,500	4,000	4,000	64,000	512,000
TranslationTool (unlimited languages)	✓	✓	✓	✓	✓	✓	✓	✓
Security System (unlimited groups/users)	✓	✓	✓	✓	✓	✓	✓	✓
Email interface (SMTP)	✓	✓	✓	✓	✓	✓	✓	✓
Dial-Up interface (RAS)	✓	✓	✓	✓	✓	✓	✓	✓
PC Based Control Integration (shared tags)	✓	✓	✓	✓	✓	✓	✓	✓
Alarm Online / History	✓	✓	✓	✓	✓	✓	✓	✓
Trend Online / History	✓	✓	✓	✓	✓	✓	✓	✓
Advanced Trending Tool (Online/Historical)	✓	✓	✓	✓	✓	✓	✓	✓
Recipe (ASCII / XML)	✓	✓	✓	✓	✓	✓	✓	✓
Report (ASCII / RTF)	✓	✓	✓	✓	✓	✓	✓	✓
ODBC	☒	✓	☒	✓	☒	✓	✓	✓
ADO.NET Relational DB support	✓	✓	✓	✓	✓	✓	✓	✓
DDE Client / Server	☒	✓	☒	✓	☒	✓	✓	✓
OPC Client / Server with OPC HDA	✓	✓	✓	✓	✓	✓	✓	✓
TCP/IP Client / Server	✓	✓	✓	✓	✓	✓	✓	✓
Scripting Module (Math)	✓	✓	✓	✓	✓	✓	✓	✓
VBScript Support	✓	✓	✓	✓	✓	✓	✓	✓
Event Module (Scheduler)	✓	✓	✓	✓	✓	✓	✓	✓
Group of screens	✓	✓	✓	✓	✓	✓	✓	✓
ActiveX & .NET Container	✓	✓	✓	✓	✓	✓	✓	✓
Static objects (shapes, button, text)	✓	✓	✓	✓	✓	✓	✓	✓
Bitmap objects	✓	✓	✓	✓	✓	✓	✓	✓
Picture (Paste Link)	✗	✓	✗	✓	✗	✓	✓	✓
Dynamics: Command; Text I/O (Input/Output); Bar graph; Color	✓	✓	✓	✓	✓	✓	✓	✓
Dynamics: Hyperlink; Position; Resizing; Show / Hide; Rotation	✓	✓	✓	✓	✓	✓	✓	✓
List box; Combo box	✓	✓	✓	✓	✓	✓	✓	✓
Smart Message; Pushbutton	✓	✓	✓	✓	✓	✓	✓	✓
Import Wizards	✓	✓	✓	✓	✓	✓	✓	✓
APIs Toolkits support	✓	✓	✓	✓	✓	✓	✓	✓

Web Sessions
- 1 included with runtime
- Max of 8 for WinCE
- Max of 256 for WinXP

✓=Supported ✗=NOT supported ☒=NOT supported by Windows CE

Windows XP designation includes Windows NT, 2000, Server 2003 and Vista

Recent IWS Versions and Service Packs

- IWS 6.0 SP3 Oct. '04
  - Added ADO.Net Support
  - Enhanced Database Redundancy (Web Server, automated switchover, automated recovery)
  - Added RSLogix Import Wizard

- IWS 6.0 SP4 March '05
  - Added OPC HDA for Win2K/XP/NT (Historical data access for OPC clients)
  - Added encapsulation of serial protocols over TCP/IP or UDP, modem dialing to remote sites
  - Added new Import Wizards : TwinCat, Omron CX-Programmer (available from Omron only)

- IWS 6.0 SP5 April '05
  - Added the Grid Object
  - Added PanelBuilder Import Wizard

- IWS 6.1 December '05
  - Added New Trend Object
  - Added VBScript support
  - Added OEM Protection Function

- IWS 6.1 SP1 May '06
  - Added Linked Symbols support
  - Added PanelMate Import

- IWS 6.1 SP2 December '06
  - Made IWS a .NET container
  - Updated the Symbol Library
  - Added custom Properties for IWS Objects
  - SNMP Agent
  - See **MLIST.PDF** found in the IWS 6.1 SP2 Program Directory

Notes

Chapter 4. InduSoft Web Studio Configurations

This Chapter covers the installation of IWS. Before beginning the installation, you should make sure:

1. Your PC meets the Minimum System Requirements listed below
2. You have logged on to the PC as a user with administrative rights

If a previous version of IWS (or CEView) is installed, you should either uninstall the previous version or install the new software version in a different path. The default installation path is C:\Program Files\InduSoft Web Studio v6.1.

Important Notes:

If you have installed a previous version of IWS or CEView and intend to continue to develop applications using the previous version, it is **HIGHLY** recommended to do the following:

Be sure you make backup copies of all your application files

If you intent to keep older versions of IWS, install the new version of IWS (or CEView) into a different path that clearly designates the new version and Service Pack (e.g. C:\Program Files\InduSoft Web Studio v6.1SP2)

Create a new folder to store applications developed using the new version of software

Be sure to start the proper version of IWS and then select the application you want to develop. Don't double click on the application file (*.app) to start IWS (CEView).

Don't use the new version of IWS (CEView) software to develop an existing application (previous version or Service Pack) that you want to maintain at a previous version.

If you accidentally open up an application (that you want maintain at a previous version) with the new software, **UNDER NO CIRCUMSTANCES** should you select the "Upgrade to New Version" option if prompted to. There is no recovery.

Minimum Development System Requirements

IBM-Compatible PC with Pentium 4 Processor (or better) with 512MB RAM (or more)

512MB (1GB preferred) of disk drive space for the development software and application program storage. Note that if historical trend, alarm or event files are stored, additional storage space may be required.

100% IBM-compatible VGA or SVGA display adapter with 64MB Video RAM (VRAM) or higher

Ethernet port (wired or wireless)

Microsoft-compatible pointing device (such as a mouse, trackball or touch-screen)

Windows XP, 2000, Server 2003 or Vista (Windows XP Pro, Server 2003 or Vista recommended)

Standard keyboard with Function Keys F1-F12 and Numeric Keypad (or USB Keypad)

CD-ROM Drive (if installing IWS from the InduSoft CD)

USB port or Parallel Port if a HardKey license is used

Microsoft Internet

IWS (CEView) can be installed from the InduSoft CD or downloaded from InduSoft's web site at www.InduSoft.com.

Example: Uninstalling a previous version of IWS

Use the following procedure to uninstall a previous version of IWS:

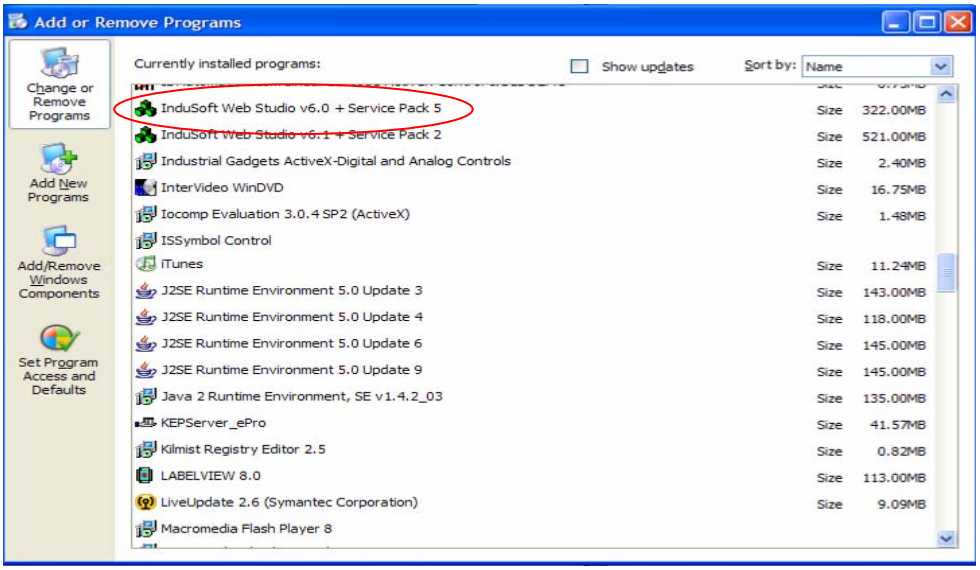
Turn on the power to your development computer, log on as a user with Administrative Rights and be sure that no other programs are running. Be sure that any application files are backed up.

Open the Windows Control Panel.

E.g. With Windows XP, click on the **Start** button in the lower left corner. Click on the Control Panel icon.

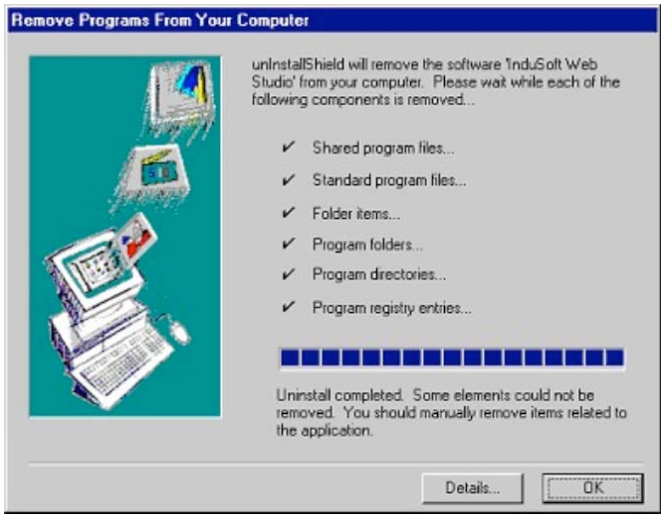
Click on the **Add or Remove Programs** icon

Select the previous version of IWS that you want uninstalled.



Click on the **Change/Remove** button that appears.

Follow the instructions from the Uninstall Shield Wizard to uninstall the older version of InduSoft Web Studio.



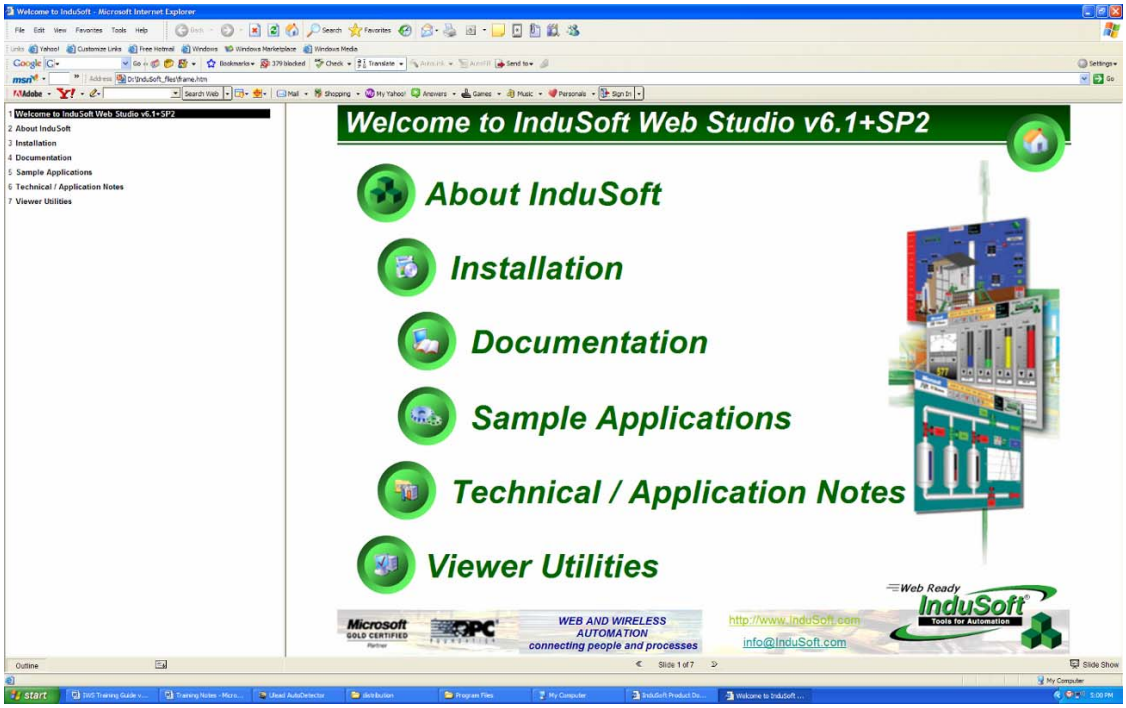
Example: Installing IWS v6.1 SP2 from the InduSoft CD

Use the following procedure to install IWS Version 6.1 and/or Service Pack 2 from the CD-ROM:

Turn on the power to your development computer, log on as a user with Administrative Rights and be sure that no other programs are running.

Insert the InduSoft CD-ROM labeled “InduSoft Web Studio v6.1 SP2” into the computer’s CD-ROM drive. An Internet Explorer Browser window should display automatically.

Note: You may be prompted to allow installation of an ActiveX control. Allow this ActiveX Control to be installed.



Internet Explorer Browser Window

If the Browser window does not display, you can manually start the program from the Windows Explorer. Navigate to the **<*D*>:\Installation** directory (where **<*D*>** is your CD-ROM drive), and click on the **Indusoft** html document.

To open Windows Explorer from Windows XP:

Click **Start,** point to **All Programs**, point to **Accessories**, then click **Windows Explorer**

The Browser Window should now display a welcome screen that allows you to navigate as follows:

- **About InduSoft**
  Contains several Powerpoint files including a brief introduction to InduSoft, InduSoft products and features, and sample case studies.

- **Installation**
  Installs IWS 6.1 and the Service Pack 2 upgrade.

- **Documentation**
  Contains various IWS documentation in PDF format including the InduSoft Web Studio Users Manual.

- **Sample Applications**
  Contains IWS application and communication examples that you can use.
- **Technical/Application Notes**
  Contains various Technical and Application Notes.
- **Viewer Utilities**
  Contains the Microsoft Powerpoint Viewer, WinZip and Adobe Acrobat Reader PDF Viewer utilities.

Notes:

- If you do not have Microsoft Powerpoint or Powerpoint Viewer installed on your PC, you need to install Powerpoint Viewer to view certain Powerpoint files contained on this CD. Powerpoint Viewer is available under "Viewer Utilities". Likewise, you will need the Adobe Acrobat PDF Viewer to view certain
- A demo version of Symbol Factory from Software Toolbox (*www.softwaretoolbox.com*) is available in the \AddOns folder on the InduSoft CD. Click on the \SymbolFactory sub-directory and then double click on Symbol Factory ActiveX to install. Follow the instructions.

In the Browser window, double-click the **Installation** button and then double-click the **InduSoft Web Studio v6.1** icon to start the InduSoft Web Studio Installation Wizard. A **Setup** dialog displays to inform you that the InstallShield Setup Wizard is loading. Click on **Run.**

Note:

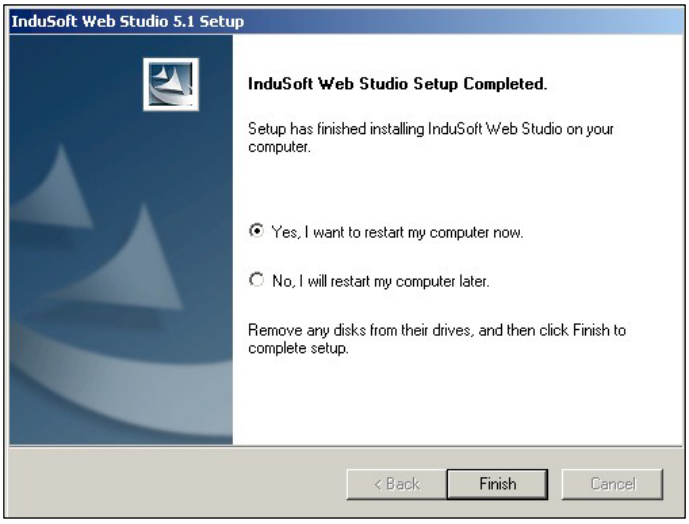
- If you are have a existing installation of IWS Version 6.1 (SP1 or earlier) on your PC, you may skip this step (and the next few steps). Proceed to the last step "Once your computer has restarted, ….." to install the Service Pack 2 update only.

Follow the instructions provided by the Setup 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.
- *A*ccept the default if you have uninstalled the previous version of IWS.
- If you have a previous version of IWS installed, select a different (unique) path name to install IWS.
- 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:

You must restart your computer to continue. Click the **Yes, I want to restart my computer now** radio button, and then click **Finish**.



Setup Complete Dialog

Once your computer has restarted, repeat the above steps but this time, double-click on the install **Service Pack 2 for InduSoft Web Studio v6.1** button after double clicking on the **Installation** button. Click on **Run** to begin the installation process for Service Pack 2.

After the installation is complete, IWS Service Pack 2 will be installed on your hard drive at C:\Program Files\InduSoft Web Studio v6.1 (or another directory if you had specified it). An InduSoft Web Studio icon will be added to your desktop.

Notes:

- Microsoft *.NET Framework 2.0* is automatically installed with as part of the Service Pack 2 installation process. The *.NET Framework 2.0* is required to enable IWS to become a .NET container, and to provide ADO.NET (Microsoft ActiveX Data Objects) support.
- During the installation process, you will be given the option to install files for Windows CE-based applications. If you are not developing Windows CE-based applications, you can omit these files. They can be added at a later time if required.
- If you are upgrading from InduSoft Web Studio Version 6.0 or earlier, your license will need upgrading.
- IWS Service Pack 2 development software allows you to **UPGRADE** an existing application from a previous version to IWS Version 6.1 Service Pack 2. However, you cannot develop an application using IWS 6.1 SP2 and save the application in a format compatible with a previous version of IWS.
- If you currently have IWS Version 6.1 installed and are simply looking to upgrade to Service Pack 2, then you do not need to uninstall the current IWS Version 6.1. You need to install the Service Pack 2 upgrade only

Example: Installing IWS v6.1 and SP2 from InduSoft's Website

Use the following procedure to download and install IWS Service Pack 2 from InduSoft's website:

Be sure your PC has a utility to unzip .ZIP files. If not, download a copy from WinZip' website at www.winzip.com.

Use your Browser to connect to InduSoft's website at www.InduSoft.com.

In the **Downloads** section on the home page (lower left corner), click on the link **Click here** in the Try before you Buy paragraph.

Click on **InduSoft Web Studio v6.1** link in the Download Products page.
Note: The Download Products page also provides other downloads, such as older versions of IWS, Reference Manuals, Sample Applications, and 3rd party demo products.

Complete the form and click the **Send** button.

Before downloading, create a directory on your PC's hard drive. The location on the directory is not critical but the directory name must be unique (e.g. **C:\InduSoft Downloads**). Create two subdirectories, **\IWS6.1** and **\IWS6.1SP2**.

If you need to install InduSoft Web Studio v6.1, locate **Download InduSoft Web Studio, Version 6.1** and click on the link **iws61.zip.** When the download starts, save this file into the subdirectory **\IWS6.1.** (This is a big file, approx. 200 MB. Download time depends on your network connection speed).
Note: You will need to download and install InduSoft Web Studio v6.1 if either you are

- upgrading from Version 6.0 or earlier, or
- if you want to install Version 6.1 Service Pack 2 separately from other previous versions of Version 6.1 (e.g. Service Pack 1). Some developers need to maintain older versions of InduSoft Web Studio to support existing installations.

To download InduSoft Web Studio v6.1 Service Pack 2, locate **Download InduSoft Web Studio, Version 6.1, Service Pack 2** and click on the link **iws61sp2.zip**. When the download starts, save this file into the subdirectory **\IWS6.1SP2.** (This is a big file, approx. 200 MB. Download time depends on your network connection speed).

Use the ZIP utility to UnZip (extract) the .Zip files you downloaded. Extract each .ZIP file to the same subdirectory where the .Zip file is currently stored. A new subdirectory **\Disk1** will be created.

If you need to install InduSoft Web Studio Version 6.1, open the folder **\IWS6.1\Disk1** and run **Setup.exe**. An installation wizard will help you install InduSoft Version 6.1 on your system. Follow the Wizard's instructions.
Note: If you intend to keep an older version of InduSoft Web Studio Version 6.1 (e.g. Service Pack 1), then during the installation process, specify a different destination folder to install the files for InduSoft Web Studio Version 6.1 (e.g. c:\Program Files\InduSoft Web Studio v6.1SP2).

To install Service Pack 2, open the folder and subdirectory where you stored the UnZipped Service Pack 2 files. Run **Setup.exe.** An installation wizard will help you install Service Pack 2 for InduSoft Web Studio Version 6.1. Choose the destination folder for IWS Version 6.1 that you want to upgrade.

Note:

- Microsoft *.NET Framework 2.0* is automatically installed with as part of the Service Pack 2 installation process. The *.NET Framework 2.0* is required to enable IWS to become a .NET container, and to provide ADO.NET (Microsoft ActiveX Data Objects) support.

The Licensing Process for IWS Software

All InduSoft Web Studio software is licensed. As noted in Chapter 3, there are Development licenses (development licenses include one engineering and one runtime license) and Runtime-only licenses. The Engineering license (Development) can be HardKey (USB or Parallel Port) or SoftKey based, as can the Runtime licenses. Any Runtime licenses used with Windows-CE based systems **must** be SoftKey based. Engineering (Development) licenses can run on any Windows OS platform **except** Windows CE.

Licenses on HardKeys will be preprogrammed by InduSoft before being sent out for a customer order. SoftKey licenses require a valid site code from the PC or device where the Runtime license will be installed. After receiving this site code, InduSoft will generate a site key that enables the license on the PC or device. This site key is unique for the PC or device and will not be valid on any other PC or device. Both HardKeys and SoftKeys can be field upgraded (e.g. to support a new version of IWS).

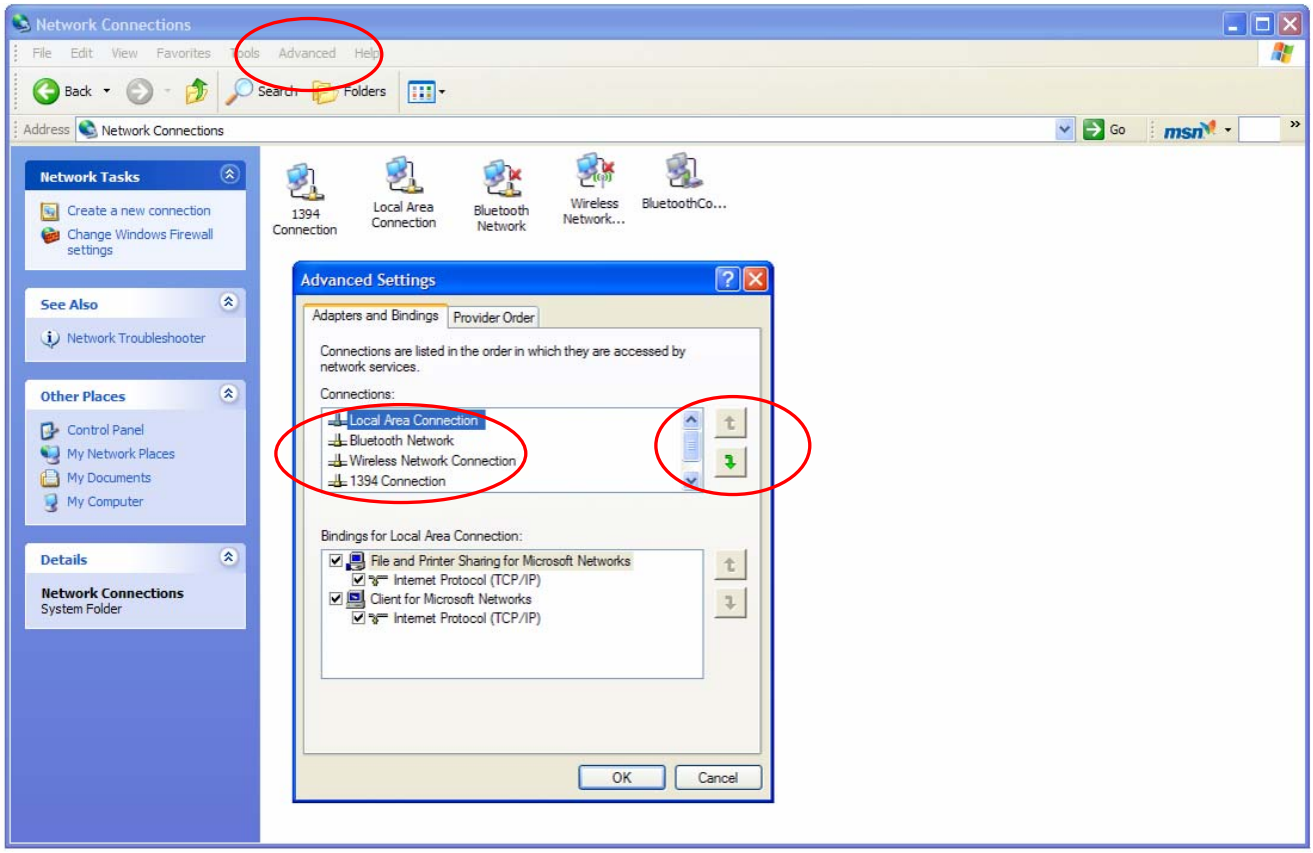
Installing or Upgrading a SoftKey License (for PC-based Platforms)

Be sure InduSoft Web Studio plus Service Pack 2 is installed on your development PC.

Make sure your HardKey is plugged into a USB port or Parallel port on your development PC.

Before starting the licensing procedure, be sure your built-in network controller (e.g. Ethernet Controller) is selected and placed on top of the Network Services Access List.

Select **Start → Control Panel → Network Connections → Advanced** (Menu Item) **→ Advanced Settings.** A dialog box will appear as show. Make sure **Local Area Controller** is at the top of the list (use up/down buttons to set)

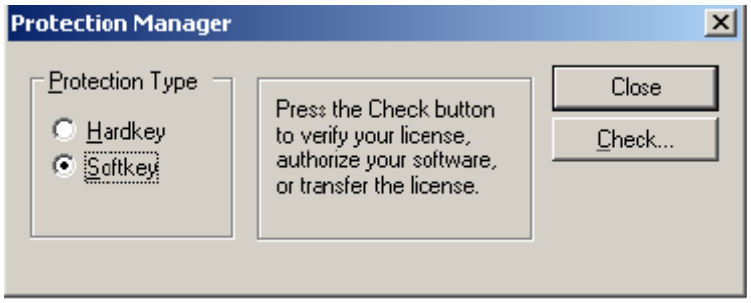


After completion of the licensing procedure, the order of the network connections can be altered. The reason for this step is that the licensing algorithm uses (as one if its components) the MAC ID address of a **Network Adapter**. The first **Network Adapter** in the list is selected. If the first item in the list is a **Network Adapter** that is a removable device (e.g. a PCMCIA, CF, USB, etc. device with a network adapter), if that device is removed, the licensing software will no longer recognize the PC as a validly licensed device. However, once the initial licensing procedure is completed, the licensing algorithm will examine all **Network Adapters** in the list in an attempt to find the correct **Network Adapter** for the license.

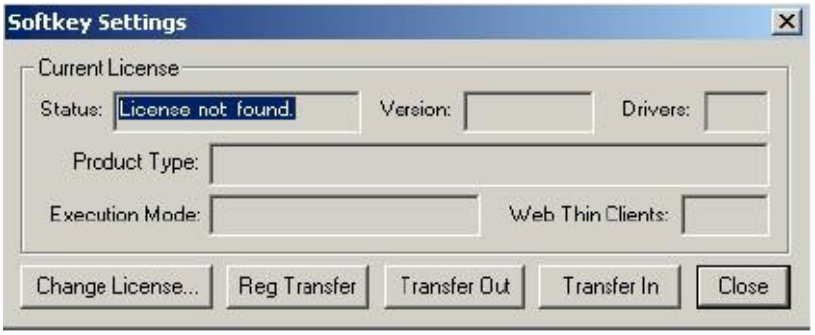
Select **Start → Programs → InduSoft Web Studio v6.1 → Register**

Note: Depending how you installed IWS 6.1 SP2, the above Program Group may be named differently.

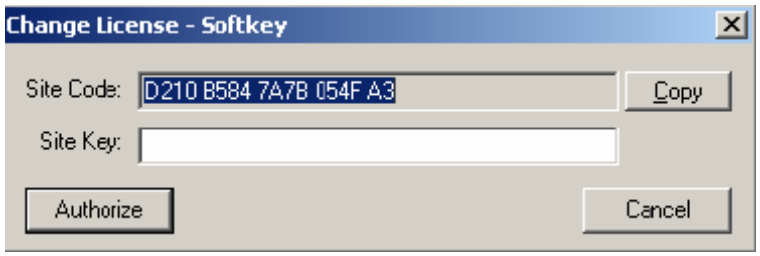
The IWS Protection Manager will be started. A dialog box will be displayed. Select **Softkey** and then click the **Check** button.



If a valid SoftKey is not found, you will get a dialog box similar to that shown below. If there is a valid SoftKey license currently installed on the PC, the Site Code for this license will be displayed in the Status field (e.g. when you are upgrading your SoftKey license). Click on **Change License**.



When the **Change License** button is pressed, you will get a new dialog box that displays the Site Code.



This Site Code will need to be sent to InduSoft for an updated Site Key to reflect a license change. You can copy and paste the Site Code into an email, along with a reference to the Purchase Order Number, and send it to *support@indusoft.com*.

When the new Site Key is returned, enter it in the Site Key field and click on **Authorize**. If the new Site Key is correctly entered, you will see the following message:

If you have problems, make sure the Site Key you entered is correct. If it still will not work correctly, contact InduSoft support at *support@indusoft.com*.

Note:

- The Site Code is generated from an algorithm using the PC's primary MAC ID address, along with various information from the registry. Take care not to make changes to your PC (e.g. install new programs) from the time you send the Site Code to InduSoft until you enter the Site Key, otherwise the Site Code may change and the Site Key will be invalid.
- Once you have put a valid license on a HardKey, you can move this HardKey to another PC. The license now travels with the HardKey.
- You can reference Technical Note **ProdLicense.PDF** for additional information on licensing **procedures. This document can be found on InduSoft's website at *www.indusoft.com*** or on the InduSoft CD in the \Documentation\TechNotes folder.

Installing or Upgrading a SoftKey License (for Windows CE-based Devices)

If you are using a Windows CE-based device (e.g. HMI, Controller, etc.) and intend to run an application developed using IWS or CEView on this device, you must have a valid InduSoft license installed, otherwise the application will only run in a demo mode (automatically shuts down after 2 hours of runtime).

The method of installing a SoftKey license on a Windows CE-based device is different than installing a SoftKey license on a PC. You can install a SoftKey license on the Windows CE-based device either locally or remotely and both methods are equally effective.

Notes:

- If your Windows CE-based device has an LCD or other type of screen display (e.g. an HMI), you can use both license installation methods.
- If your Windows CE-based device does not have a screen display (e.g. a Controller), you will need to use the remote license installation method.
- Some Windows CE-based devices have InduSoft licenses pre-installed. Check with your vendor.

Installing or Upgrading a Softkey License Locally

Be sure InduSoft Web Studio plus Service Pack 2 is installed on your development PC.

You need to install the **InduSoft Remote Agent** program **CEServer.exe** into your Windows CE device's non-volatile folder. The non-volatile folder can be different with every Windows CE-device. Check with your vendor for specific information.

Where to find CEServer.exe:

The **CEServer.exe** file is located on your Development PC in a subfolder contained in a folder where you installed the Program files. The file is located in the **\Redist** folder, where <platform> is a sub-folder name that specifies your processor type.

E.g. C:\Program Files\InduSoft Web Studio v6.1\Redist\

There may be different subfolders based on what options you selected during the IWS installation process. The **\WinCE 4.0** subfolder is for platforms using Windows CE 4.0 and later, while the **\CEView** subfolder is used for platforms using Windows CE 3.0

Notes:

- The non-volatile folder retains data after a power-up or reboot of the Windows CE device.
- The non-volatile folder can be different with every Windows CE-device. Check with your vendor for specific information on which folder is the non-volatile folder.
- Depending on the Windows CE device, the non-volatile folder can be located on the on-board flash memory or on a removable memory (e.g. Compact Flash) storage card. If the InduSoft license is stored on a removable memory storage card, the Windows CE-based device will not be licensed if the removable memory storage card is not properly installed in (or removed from) the device.

How to install the CEServer.exe file:

- You can copy the **CEServer.exe** file using ActiveSync (requires USB port or Serial Connection). ActiveSync can be downloaded free of charge from Microsoft's web site at no cost.
- Create a Shared Folder on the WinCE device and use the following command sequence:
  - Run
  - CMD
  - net use [<Local Name>] [Remote Name] [/user:<UserName>]
  - copy **\Redist\WinCE 4.0\<platform>\Bin\CEServer.exe CEServer.exe**

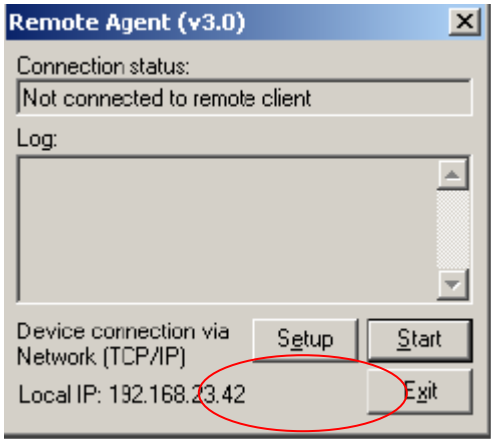
Alternatively, you can use a Compact Flash (CF) adapter and write to the CF

Configure CEServer.exe to automatically start up when you power on the Windows CE device.

Notes:

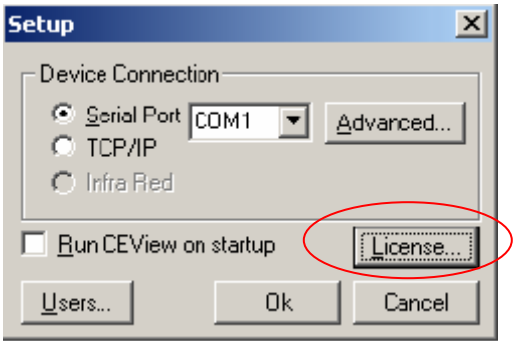
- This step is not mandatory. However, it is advised to do this step, so that CEServer.exe can automatically start your application after power up.
- Check with your vendor for specific information on how to select a program (**CEServer.**exe) to automatically execute after power up.

After the Remote Agent **CEServer.exe** starts, you will get the following dialog box on your Windows CE device. If **CEServer.exe** does not start automatically after power up of the Windows CE device, you can manually launch the file from the <non-volatile folder> on the Windows CE device.



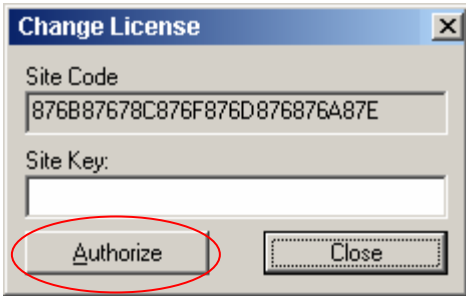
In the Remote Agent dialog box, click on **Setup**. The following dialog box will appear.

Click on the **License** button



When the **License** dialog box opens, click on the **Change License** button.

A **Change License** dialog box will now be displayed. The current site code will be displayed. This site code will need to be sent to InduSoft, along with a reference to the Purchase order for the runtime license.



When you receive your Site Key from InduSoft, enter this Site Key into the **Site Key** field, then click on **Authorize**.

If the new Site Key is correctly entered, you will see the following message:

If you have problems, make sure the Site Key you entered is correct. If it still will not work correctly, contact InduSoft support at support@indusoft.com.

Installing or Upgrading a Softkey License Remotely

Be sure InduSoft Web Studio plus Service Pack 2 is installed on your development PC.

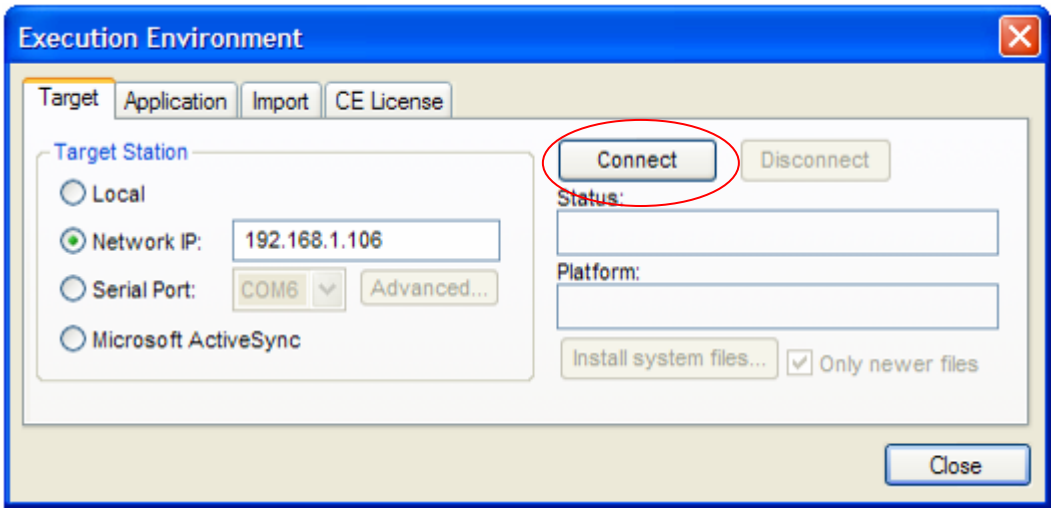
Be sure **CEServer.exe** is installed on the remotely connected (Serial or Ethernet) Windows CE device. (see the previous example)

Verify a serial or Ethernet connection between your development PC and the remote Windows CE-based device is installed correctly. The Ethernet connection must be a through a cross-over cable, hub, router, switch or other type of connection.

Note:
- You can verify the Ethernet connection is established by running the Ping command on your development PC:
  Start → Run → CMD → Ping <CE Device IP address>
  E.g. Run → CMD → Ping 10.28.64.100

Start IWS on your development PC (if it is not already started).

On the IWS Main Menu Bar, select **Project → Execution Environment**. The following dialog box will appear:

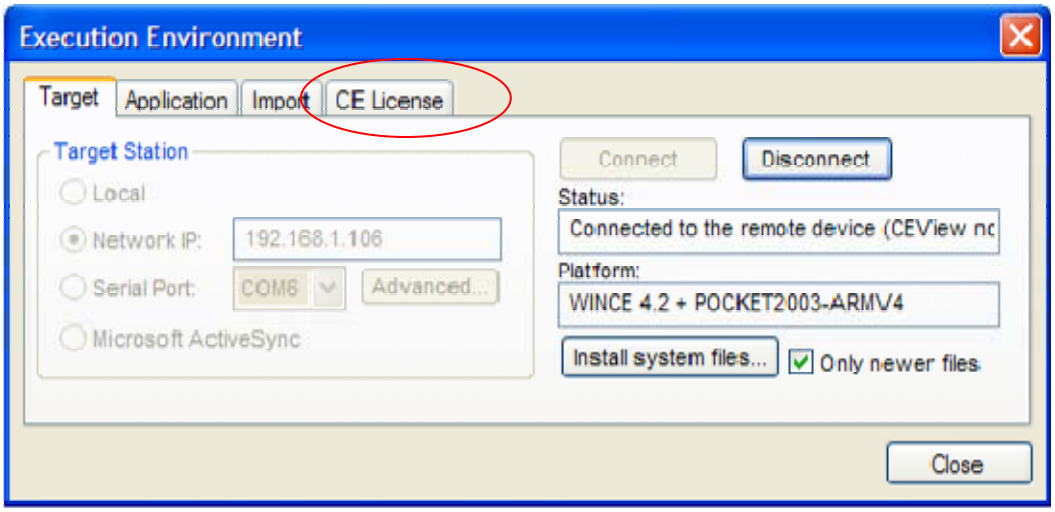


Select either **Network IP** (for an Ethernet connection) or **Serial Port** (for a Serial connection) for the Target (remote) Station connection type.

Note: If you selected **Network IP**, you must enter the IP address of the Target Station.
If you selected **Serial Port**, you must set the Serial Port number and any advanced settings

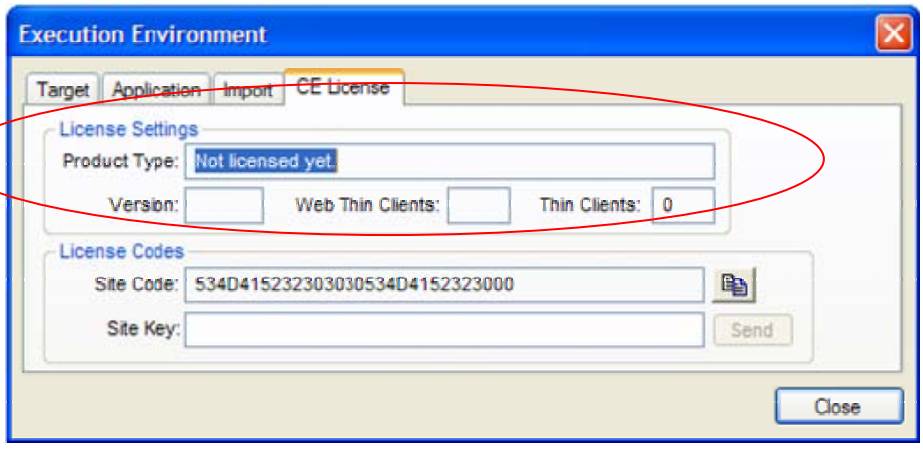
When the **Connect** button becomes active, click on the button to connect to the Windows CE device. When the development station is properly connected to the remote Windows CE device (Target Station), the **Status** Field will display "Connected".

Note: **CEServer.exe** must be properly installed and running on the Target Station in order to properly connect.

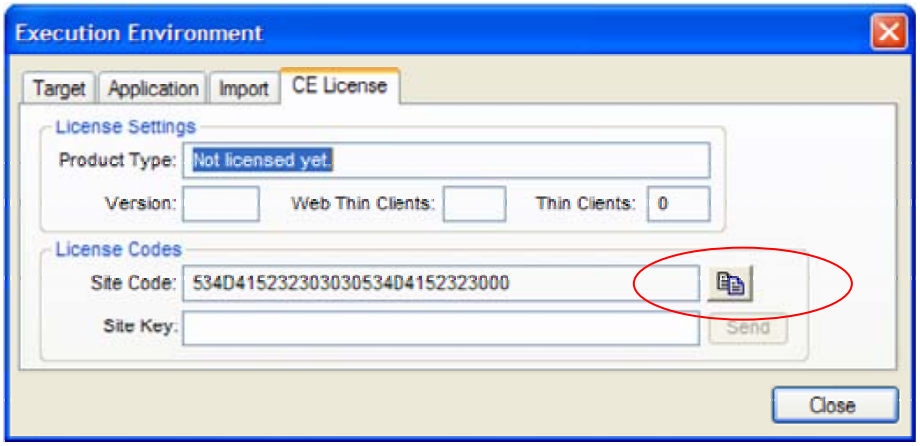


Click on the **CE License** tab

If the Windows CE device has previously been licensed, the **License Settings** Fields will be populated with information about the type of license installed in the Windows CE device. If the Windows CE device has not previously been licensed, the **Execution Environment** dialog box will be shown as below.



If you need to install a new InduSoft runtime license or upgrade the runtime license, click the **Copy** icon that is to the right of the **Site Code** field. By clicking on this icon, the **Site Code** information is automatically copied and can then be pasted in an E-mail to send to InduSoft at support@indusoft.com along with the Purchase Order information. InduSoft will return a Site Key which needs to be entered into the **Site Key** Field. The **Send** button will become active ("lit up"). Press the **Send** button to send the new Site Key to the Windows CE device. When this happens, the **Remote Agent** (**CEServer.exe**) will properly store this new Site Key in the Windows CE device.



If the new Site Key is correctly entered, you will see the following message:

If you have problems, make sure the Site Key you entered is correct. If it still will not work correctly, contact InduSoft at support@indusoft.com.

Notes:

- After the **Site Key** is sent to the Windows CE device, be sure to save the Registry settings on the Windows CE device. Otherwise, the **Site Key** may be lost after the device is rebooted.
- The method of saving the Registry is determined by the hardware vendor. Check with your hardware vendor to determine how to save the Registry.
- Windows XP/2000/Server 2003/Vista/NT automatically save their Registry settings after a change, but Windows CE does not. Since the Windows CE image must be built using the Microsoft Platform Builder and a Registry Save tool must be added, this is why different Windows CE devices can implement the Registry Save tool differently.

Upgrading a HardKey License

Be sure InduSoft Web Studio Service Pack 2 is installed on your development PC.

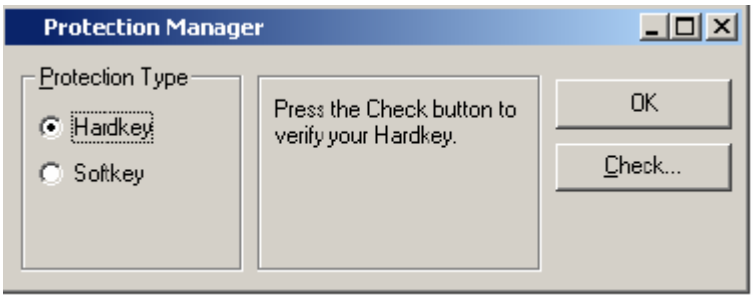
Make sure your HardKey is plugged into a USB port or Parallel port on your development PC.

Select **Start → Programs → InduSoft Web Studio v6.1 → Register**

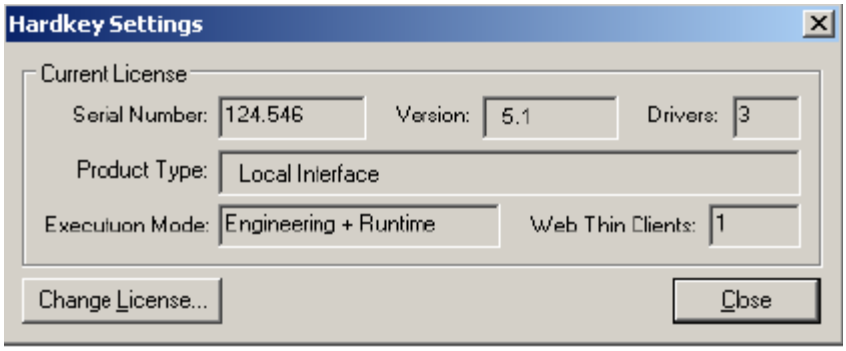
HardKey Types

Note: Depending how you installed IWS 6.1 SP2, the above Program Group may be named differently.

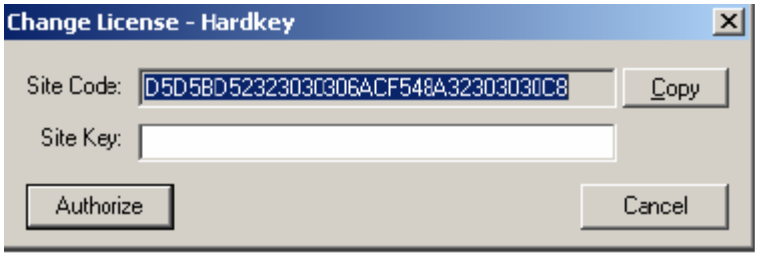
The IWS Protection Manager will be started. A dialog box will be displayed. Select **Hardkey** and then click the **Check** button.



If a valid HardKey is found, you will get a dialog box similar to that shown below:



Click on the **Change License** button.



The Site Code for this PC is displayed in the Dialog box. This Site Code will need to be sent to InduSoft for an updated Site Key to reflect a license change. You can copy and paste the Site Code into an email, along with a reference to the Purchase Order Number, and send it to *support@indusoft.com*.

When the new Site Key is returned, enter it in the Site Key field and click on **Authorize**. If the new Site Key is correctly entered, you will see the following message:

If you have problems, make sure the Site Key you entered is correct. If it still will not work correctly, contact InduSoft support at *support@indusoft.com*.

Notes:

- The Site Code is generated from an algorithm using the PC's primary MAC ID address, along with various information from the registry. Take care not to make changes to your PC (e.g. install new programs) from the time you send the Site Code to InduSoft until you enter the Site Key, otherwise the Site Code may change and the Site Key will be invalid.
- Once you have put a valid license on a HardKey, you can move this HardKey to another PC. The license now travels with the HardKey.
- You can reference Technical Note **ProdLicense.PDF** for additional information on licensing **procedures. This document can be found on InduSoft's website at *www.indusoft.com*** or on the InduSoft CD in the \Documentation\TechNotes folder.

Installing Runtime System Software on Remote Devices

InduSoft provides a common development platform allowing you to develop applications for any platform (or device) using a Microsoft Operating System (e.g. Windows XP/2000/Vista/Server 2003/NT and Windows CE). If the target (runtime) platform is also the development platform, the required IWS runtime system files are already installed when IWS 6.1 plus Service Pack 2 are installed. However, if the target platform does not have these IWS runtime system files or if the target platform is a Windows CE device (which does not support IWS development), the IWS (or CEView) runtime system files must be installed in the target platform. CEView system files can be downloaded to the CE runtime platform from the development environment, while a Windows XP/2000/Vista/Server 2003/NT system must install IWS.

Notes

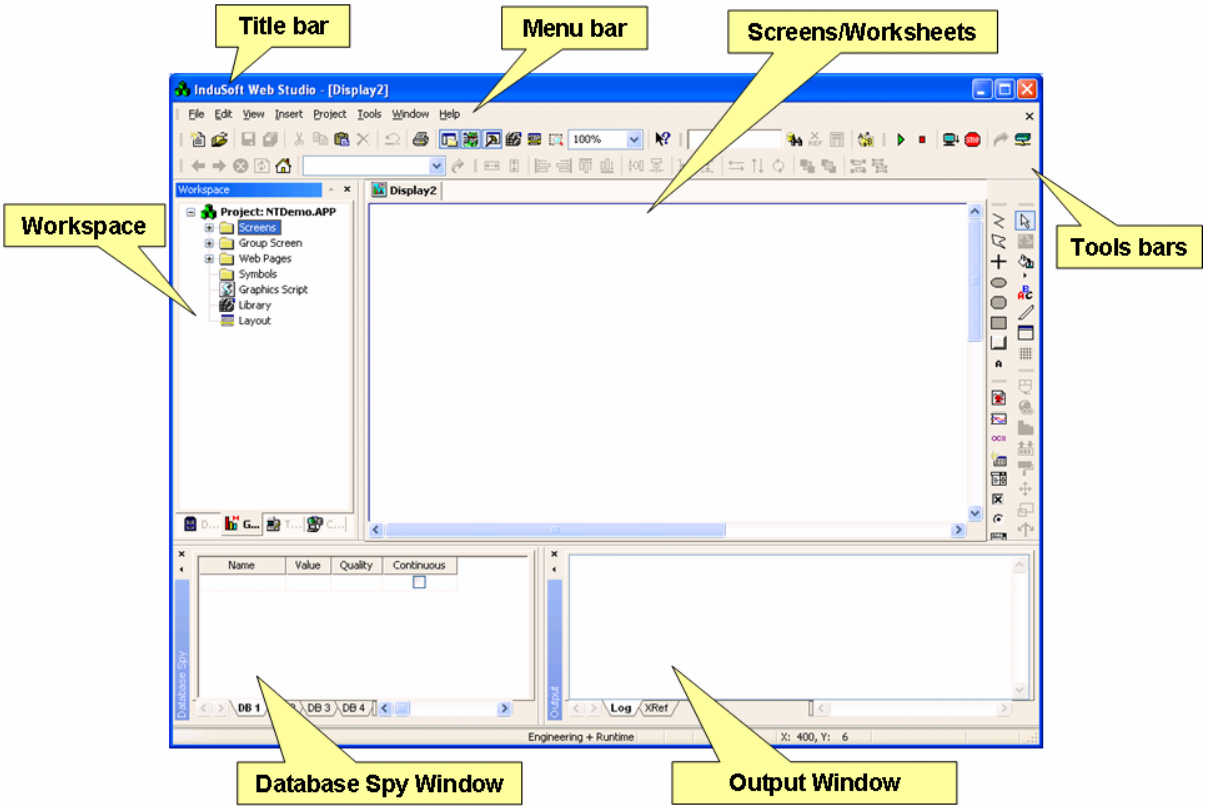
Chapter 5. InduSoft Web Studio Overview

This chapter provides a high-level overview of InduSoft Web Studio (IWS) product, the topics include:

- IWS Development Environment
- IWS Internal Structure
- Module Execution
- Microsoft Industry Standards Supported

IWS Development Environment

The InduSoft Web Studio uses standard, Windows-like tools and interfaces to provide an integrated development environment (IDE). This IDE is comprised of several sections:



IWS Development Environment

The development environment interface consists of the following features and functionality:

- Title Bar
- Menu Bar
- Toolbars
- Workspace
- Screen/Worksheet Editor
- Database Spy Window
- Output Window (*LogWin*)
- Symbol Library

Title Bar

The title bar (located along the top of the Studio window) displays the **InduSoft Web Studio** icon, the product name, and the name of the active, open display or worksheet (if any).

Example Title Bar

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

Minimize Click this button to minimize the window.

Resize/Maximize Click this button to toggle between the following two options:

- **Resize** tiles the window
- **Maximize** maximizes the window to fill your computer screen

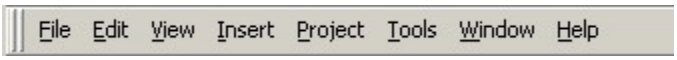
Exit (or **Close**) Click this button to automatically save the database, and then close the IWS development environment. If you modified any screens or worksheets, Studio prompts you to save your work. This button function is similar to selecting the **Exit** command from the **File** menu.

Note:

- Closing (exiting) the development system does not close down the local runtime system if it is running.

Main Menu Bar

The InduSoft Web Studio main menu bar contains the following menus:



IWS Menu Bar

File Contains options that enable you to manage application files.

Edit Contains options that enable you to manage displays and worksheets.

View Contains options that enable you to manage visible tools and provides shortcuts to the dialogs you open most frequently.

Insert Contains options that enable you to create/configure a variety of 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 tools.

Window Contains options that enable you to manage open displays and worksheets.

Help Contains options that provide links to information about the InduSoft Web Studio product and InduSoft.

Note:

- The Main Menu Bar is dockable.

The Status Bar

The status bar is located along the bottom of the Studio window and 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. Drag can be enabled or disabled by pressing **Crtl 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.

Toolbars

InduSoft Web Studio provides several different Toolbars that enable you to perform different actions 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 This Toolbar has functions such as creating/opening/saving a project, cut/copy/paste/delete, undo, print, zoom, toggle (Output Window, Workspace, Database Spy) and open the Symbol Library.

  Tag Properties This Toolbar has functions such as the Object Finder, Global Cross Reference, Tag Properties and Global Replace.

  Execution Control This Toolbar will start & stop the Test Display function, Stop & Run the application (on the local Target machine), send the application to a remote target platform, and open the Execution Environment dialog box.

  Web Allows you to access the Web from the development environment. A new screen is opened. The Web Toolbar provides the controls to navigate the Web.

  Align & **Distribute** Allows you to align and orient display the selected on the screen.

- The following toolbars contain screen editing tools, and are located along the right side of the interface window by default:

  Static Objects A set of Objects that can be placed on a Screen (e.g. rectangles, circles, buttons, lines, etc.). Can add Dynamic Properties to the Static Objects.

  Active Objects A set of sophisticated Objects that can be placed on a Screen. Many have a Task worksheet associated with it. (E.g. Alarm/Event Control, Trend Control, Grid Object, Smart Message, Combo Box, List Box, Radio Button, Check Box, ActiveX and .NET Controls). Can add select Dynamic Properties to the Active Objects.

  Dynamic Properties Adds properties to Static Objects and Dynamic Objects that can change based on a IWS tag value or an Object action. These properties include color, size, position, rotation, visibility, text I/O, command, hyperlink, and bar graph. Can have multiple Dynamic Properties on an Object.

  Mode A set of functions that include Object selection, Bitmap Editor, Fill Color, Background Color, Font selection, Line color and Grid (on or off).

  Bitmap This Toolbar allows editing of a bitmap down to the pixel level. The Toolbar is hidden by default, but can be manually enabled.

  Legacy This Toolbar contains legacy Active Objects (Alarm and Trend Display).

Notes:

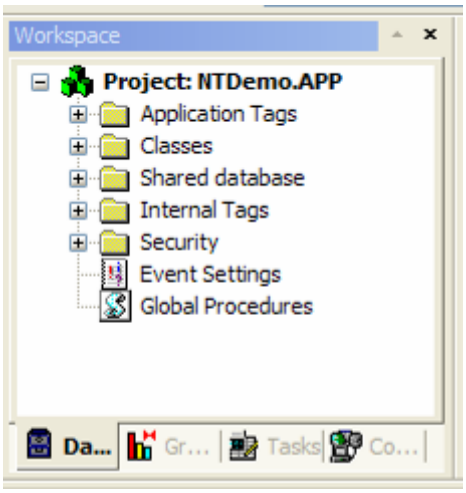
- All Toolbars are dockable Objects. You can move a Toolbar to a different screen location by clicking on its Title Bar and dragging it to a new location.
- The Status Bar (located at the bottom of the IWS interface can provide information on the selected screen Object.
- The Toolbar is customizable

Workspace

The Workspace is a user-friendly interface that enables you to quickly find any application component (e.g. tags, screens, worksheets, and communication functions). The application components are organized in a tree-view manner with each application component having its own icon and customized description. You can move, resize or hide the Workspace window.

The **Workspace** window is divided into four tabs:

- The **Database** tab enables you to access any available tags from the application, IWS Security System components, and VBScript Global Procedures. This tab includes the following folders:



Database Workspace

Application Tags Contains a spreadsheet-like list of the tags that are defined by the developer for the application.

Classes Contains Class definitions. These are not tags, but instead are definitions that are used when declaring Class Tags in the Application Tags folder.

Shared Database Declares (to IWS) tags that are generated by a PC-based control program.

Internal Tags Internal Tags are tags defined by IWS such as Time, Day, UserName, and other tags used by a typical application. These tags do not count against the IWS license tag limit.

Security This folder provides access to the IWS Security System. Security Groups and individual Users can be defined, along with Passwords, and Password control features (e.g. password aging, auto-logoff, lockup on multiple invalid attempts, etc.)

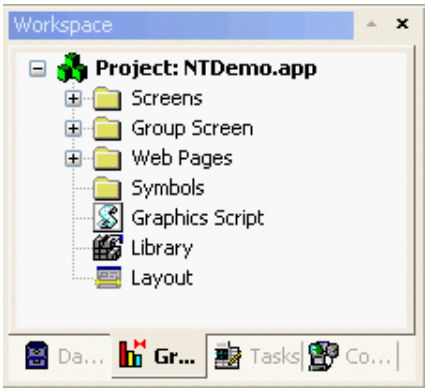
Event Settings Allows the developer to define the Events that can be logged to an Event Database. Events can include Security System events (e.g. Log On & Log Off), activating a Report or Recipe, opening or closing Screens, Tag Value changes (includes an optional deadband setting), System Warnings, and Custom Messages (from the Application)

Global Procedures Allows the developer to define VBScript procedures (Functions and Subroutines) that can be used by any VBScript code segment within the IWS application. Global Procedures can define variables to be used by a procedure, but these variables are not directly accessible outside of the Global Procedures section.

Notes:

- The Shared Database is only one way to access Tags from a PC-based Control program. The more common way is through OPC. The PC-based Control program vendor typically provides an OPC Server. IWS is the OPC Client that can browse and use these Tags.

- The **Graphics** tab enables you to access all screens and symbols in the application. This tab includes the following folders and icons:



Graphics Workspace

Screens Allows you to insert Screens and open existing Screens. Each Screen has definable display attributes, and script (IWS or VBScript) segments that can be run the Screen is opened, closed or while running.

Group Screen A Group Screen is a combination of one or more MDI Child Windows. The Group Screen has a unique name and can be referenced for opening or closing.

Web Pages These are HTML version of Screens and Group Screens. Used to support Web Thin Clients.

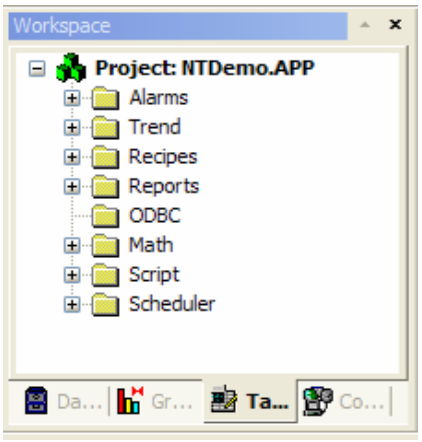
Symbols The Symbol folder contains user defined symbols that can be inserted onto a Screen.

Graphics Script This is a VBScript code segment that can be run whenever the first Screen is opened, when all Screens are closed, or when any screen is open.

Library This icon provides an alternative method of opening the IWS Symbol Library.

Layout The Layout icon will use the open Screen and display it as it will appear during runtime. Most of the Screen functions are not active in **Layout**. This function is used to give the developer an idea of what the runtime display for this Screen (or Group Screen) will look like.

- The **Tasks** tab enables you to access all task worksheets in the application. This tab includes the following folders:



Tasks Workspace

Alarms The Alarm Worksheet is used to define the IWS tags used for alarming purposes, and the alarm limits for the tags. The Alarm Worksheet is used in conjunction with the Alarm/Event Control Object (that displays Alarms & Events). Alarm information can be logged to a database and an email can be automatically generated.

Trend The Trend Worksheet is used to define the IWS tags to be used for data logging (historical trending). The Trend Worksheet works in conjunction with the Trend Control Object (displaying on-line and historical trend data). The historical trend database to be used can be specified in this worksheet.

Recipes The Recipe Worksheet is used to define the IWS tags to be used in conjunction with a Recipe file. A Recipe file defines a template for storing data that changes with a particular job or process. The data can be used to define IWS tags, or can be communicated to a PLC or other device. Multiple Recipe names can be created that use the same template. Recipe file access is controlled through recipe load and store commands. Recipes can be stored in ASCII or Unicode Text or XML format.

Reports The Report Worksheet is used to define a report in ASCII Text or RTF (Rich Text Format). A report can be printed or saved to a file.

ODBC The ODBC (Open Database Connectivity) Worksheet is used to define the database connection (DSN, Table, trigger, etc.) and the tag used for each Field in the ODBC database. Connections to ODBC-compliant databases can easily be configured; e.g. SQL Server, Oracle, Access, Excel.

Math Worksheets Math Worksheets are used in conjunction with the IWS Scripting Language to define programming instructions that run in the background. These Math Worksheets can be enabled or disabled by IWS tags or expressions. The Math Worksheet appears as a group of programming lines and execute sequentially, providing math & logical functions, as well as support for built-in IWS functions (e.g. File access, System functions, etc).

Script Script defines one or more VBScript code segments that run in a background mode (i.e. they are not associated with a Screen, a Screen Object (or command), or the Graphics Module. Scripts can be enabled or disabled by IWS tags or expressions. Scripts are similar to Math Worksheets but use VBScript instead of the IWS Scripting Language.

Scheduler Scheduler Worksheets are used to trigger activities in an IWS application. Triggers can be based on Time, a calendar setting, or by tag value change. Timing resolution to 100 milliseconds is provided.

Notes:

- If you are using ODBC, you must set up the ODBC worksheet(s). If you are using ADO.NET, it is only required to define the connection to the ADO.NET Provider. The ADO.NET Provider connection setup is done as part of the Project Settings configuration or by the Dynamic Object that uses the Database for storage or retrieval.

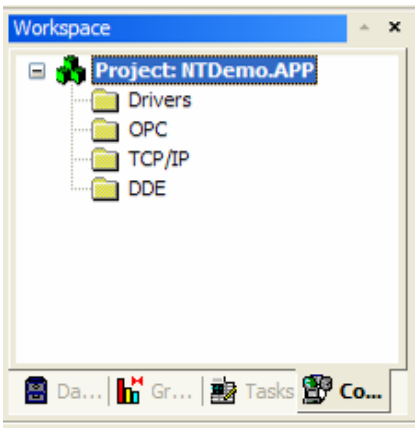
- The **Comm** tab enables you to access all worksheets used to configure communication with another device. This tab includes the following folders:

  Drivers The Driver Worksheet allows one or more IWS drivers to be configured to communicate to PLCs, RTU, and other devices. One Main Driver Sheet and multiple Standard Driver Sheets can be configured.

  OPC Used to configure IWS as an OPC Client. Remote OPC is supported.

  TCP/IP Used to configure communication between two or more PCs or devices using IWS (TCP/IP link required).

  DDE Used for DDE (Dynamic Data Exchange) communication



Communication Workspace

Notes:

- Your application can use multiple IWS drivers, up to the limit of your license.
- DDE is primarily used for support of legacy applications and is not recommended for new applications. Instead, it is recommended to use OPC.
- IWS can be both a OPC Client and an OPC Server. The OPC Server is activated in the **Project → Status → Execution Tasks** dialog box. No other setup for the OPC Server is required.

Screen/Worksheet Editor

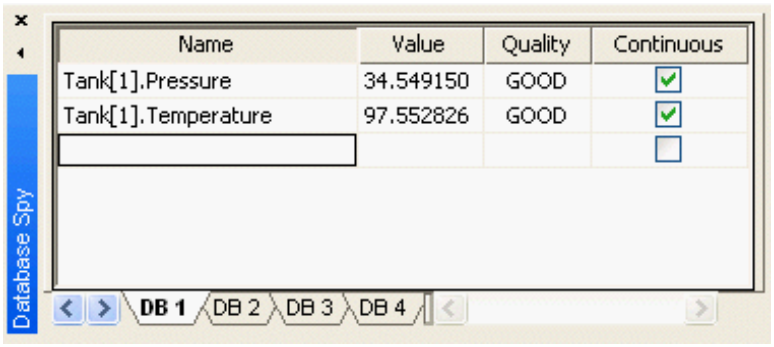
IWS comes with a powerful, object-oriented screen editor used to create and edit a variety of screens and worksheets for your applications. You can input information using your mouse and keyboard, output control data to your machine or process, and automatically update screens based on data input from your machine or process.

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

Database Spy Window

The **Database Spy** window is a key debugging tool. It lets you monitor and force Application tags and execute and test both functions and expressions. The Database Spy is available in both the development environment and the runtime environment. A separate Remote Database Spy tool is also available (select from the Main Menu Bar; i.e. **Tools → Remote Database Spy**).



Database Spy Window

Notes:

- If the Database Spy window is not shown in the development environment, from the Main Menu Bar select **View → Toolbars → Database Spy** (or press **Alt 2**).
- To access the Database Spy when the runtime is activated, either select from the Runtime Menu **Tools → Database Spy** (the Runtime Menu must first be activated in Project → Settings → Runtime Desktop), or toggle (**Alt Tab)** to the development environment. If the runtime platform is a Remote PC or device, you must use the Remote Database Spy tool.
- The Database Spy has the following fields:

  Name — Displays tag names, function names and expressions
  Value — Displays the tag values and expression results
  Quality — Displays a quality evaluation (**Good** or **Bad**) of the tag or function source. For example, **BAD** can mean an invalid read from a PLC register. When using with an OPC Server, **BAD** means that the OPC Server got a bad reading from the PLC; it is not an indication of the communications between the IWS OPC Client and the OPC Server.
  Continuous — Displays whether IWS is re-evaluating the tag, function or expression continuously

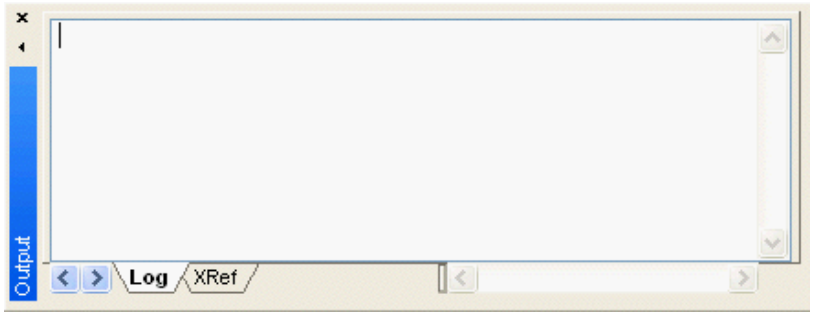
Output Window (LogWin)

Use the **Output Window** (also known as **LogWin**) is used to view a variety of items. The Output Window is used for two purposes, with a tab on the bottom of the window used to toggle between them. The first is the LogWin function and the other is for the XRef (Cross Reference Tool) output.

The LogWin messages include IWS Errors (displays compile and runtime errors) as well as a number of configurable outputs including:

- Serial and Ethernet communications
- Field Read & Writes (to a PLC or other device)
- OPC Transactions
- TCP/IP Transactions (to another IWS application or to a Web Thin Client)
- Screen (display) opening and closing
- Database Reads & Writes
- Log In & Log Out
- Recipes & Reports activated (including which Object activated the Recipe or Report)
- DDE transactions
- Trace Messages (from the IWS built-in function Trace)
- IWS Tag value changes

The LogWin Output Window can be configured by placing your mouse cursor in the Output Window and clicking the right mouse button. Select **Settings** and select the checkboxes for the items you want to monitor.



Output Window

Notes:

- The Output Window is a very powerful tool that lets you monitor activities in your IWS application.
- It is frequently effective with troubleshooting PLC communication issues. Check the Field Read & Field Write checkboxes along with the Serial (includes Ethernet) communication checkbox.
- The TCP/IP transaction checkbox monitors transactions between two or more PCs or devices using IWS (or CEView) as well as Web Thin Client communications. It is not monitor all Ethernet-based transactions.

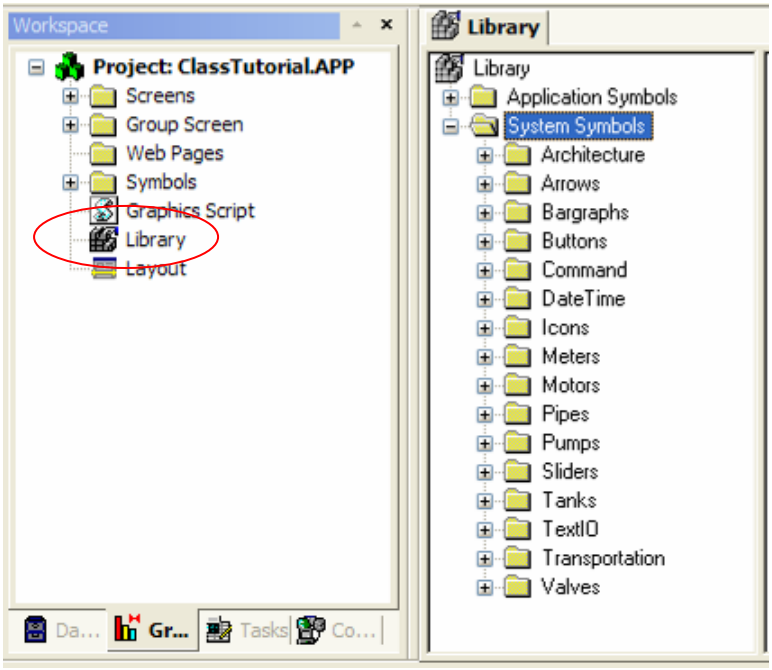
Symbol Library

Symbols are reusable objects (or groups of Objects) that you can store in a Library for reuse. IWS provides an extensive Symbol Library for your use. You can use the Symbol(s) in an application, edit the existing Symbols to create new Symbols, create you own Symbols, or add Symbols from a 3rd party library of Symbols.

There are several ways to access the IWS Symbol Library. The most common method is to click the **Open Library** icon in the Standard Toolbar (located just below the Main Menu Bar). The same icon is also located in the **Graphics Workspace**. Clicking (left mouse click) on either will open the Symbol Library.

By selecting **View → Library** in the Main Menu Bar (or pressing **Ctrl A**), you can also open the Symbol Library.

IWS Version 6.1 SP2 organizes the Symbol Library differently than previous versions. First, you have a folder of System Symbols (what comes with IWS 6.1 SP2) and a folder of Application Symbols for any Symbols you create for your application.



The System Symbols are organized by type, similar to previous version of IWS. However, many of these Symbols have a different look than Symbols in previous version of IWS. By clicking on any folder (sub-directory) of the System Symbols, you will be able to see all of the Symbols in the folder.

By clicking (selecting) one of the Symbols a folder, you can then open the application Screen you want to place the Symbol on. Position your mouse on the application Screen and click the left mouse button again.

Notice that even Symbols that look similar to previous versions of IWS may not be the same. That is because the many of the new Symbols are Linked Symbols. Support for Linked Symbols was added with IWS Version 6.1 SP1 but the Symbol Library was revised for SP2.

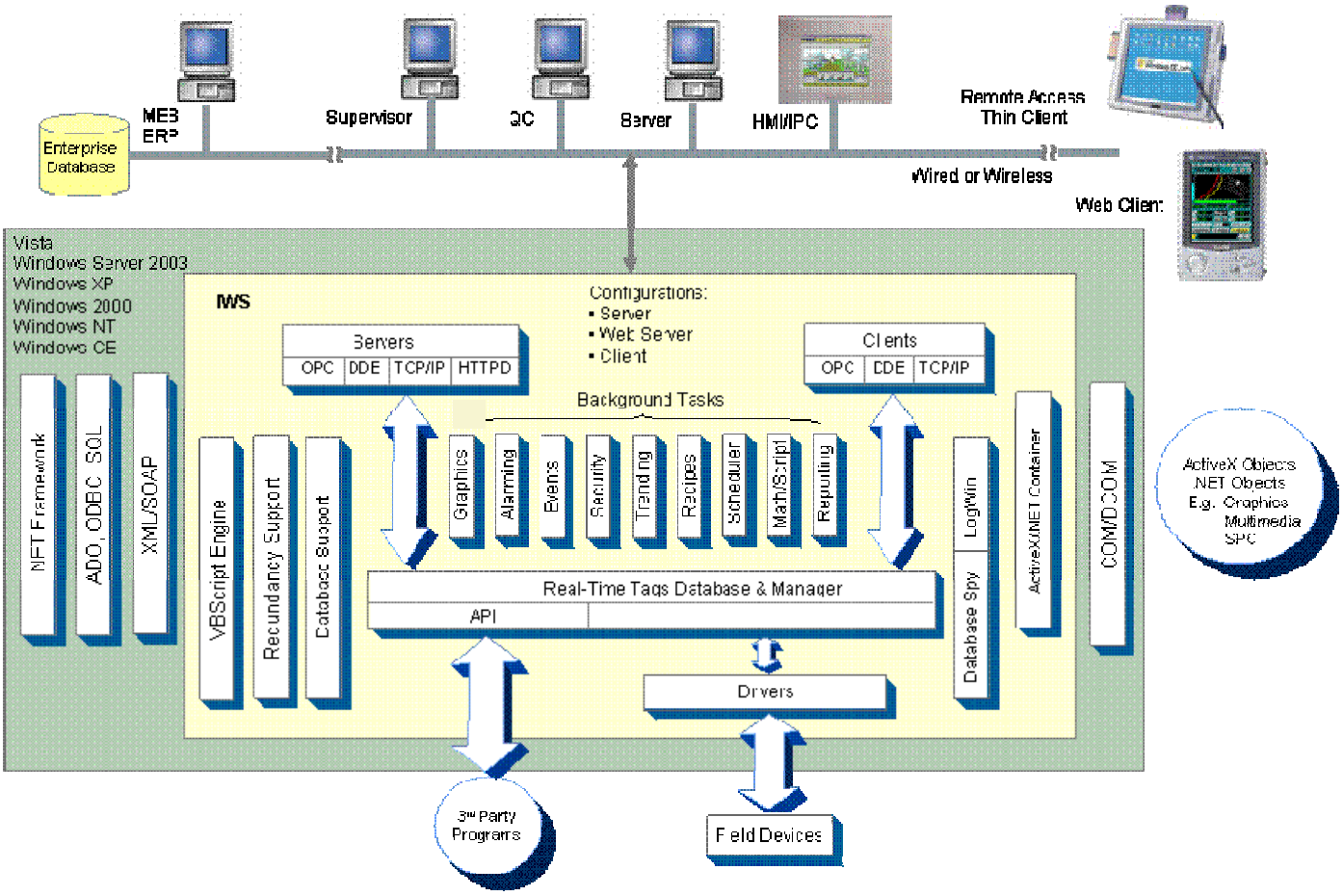
Linked Symbols are a significant upgrade to the previous IWS Symbol Library. The major benefit of a Linked Symbol is that if you change the Symbol in the Symbol Library, those changes are automatically replicated everywhere you use the Linked Symbol, in every application Screen. However, unlike the old Symbol Library, Linked Symbols cannot be rotated. To rotate a Linked Symbol, you need to unlink the Symbol, ungroup the Symbol (if necessary), then rotate the component(s). Then, you can regroup and Relink the Symbol.

Notes:

- The Symbol Library for IWS Version 6.1 SP2 is new and updated. Many Symbols now use a more modern color scheme and most Symbols are Linked Symbols.
- Changes to Linked Symbols are easily replicated throughout the IWS application.

IWS Internal Structure

The following diagram illustrates the basic architecture of IWS:



IWS Architecture

Real-Time Tags Database & Manager

The Real-Time Tags Database is the heart of InduSoft Web Studio product. In IWS, you use the same tag names in the worksheets and screens, and the IWS Tags Database Manager manages tag values among the modules. All modules share information through the Tags Database. The values of the application tags and InduSoft Web Studio internal tags are stored in this database during system execution. The application Tags Database is used by all modules to read or write values.

Configuring an IWS application consists of defining the tags used by each module. This means that application development follows the same logical sequence, regardless of the number of tags involved in a particular application.

In addition to Tag values, the Tags Database Manager keeps track of several tag properties (e.g. timestamp when a tag changed value, alarm limits for a tag, alarm condition for a tag, etc.).

Modules

IWS has various modules and tasks that can operate in a runtime environment to implement a user application. Many of these modules and tasks operate in a background mode.

Graphics

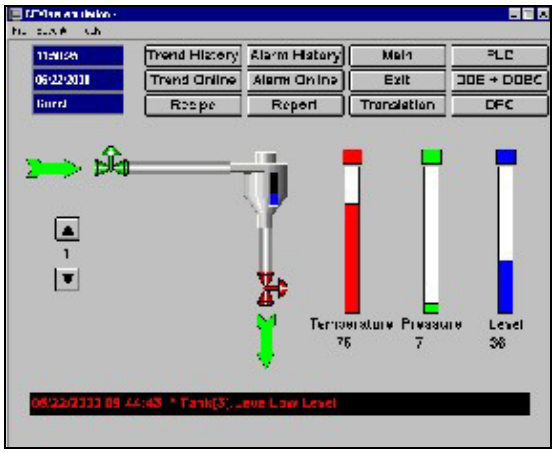
Graphical screen display is one of the most basic functions performed by InduSoft Web Studio, providing a window into the machine or process operational status. This capability is generally described as the Human-Machine Interface (**HMI**).

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 graphical **Screens**)
- The application run-time **Viewer** that displays the Screen(s)

Each Screen can use a number of Static and Dynamic Objects. These Objects can (optionally) have Properties associated with them that give they dynamic effects, such as changing color, size, rotation, etc., based on the value or a tag or expression.

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 runtimes require a Windows-compatible pointing device, such as a mouse or touchscreen. You can run an application in the Viewer without a pointing device if you configure a keypad or keyboard keys for all commands.

Background Tasks

IWS has a number of Background Tasks that execute on a Time-Slice basis. Their functions have previously been described. These Background Tasks include:

- Alarms Manager
- Event Manager
- Security System
- Trending (on-line and historical (data-logging to a database))
- Recipe Manager
- Reporting
- Scheduler
- Math Worksheets
- Script (VBScript)

Built-In Servers

IWS provides a number of built-in Servers to support communications with other software applications that may be executing on the same PC (or device) or running on a remote PC (or device). These Servers include:

- OPC Server (to communicate with OPC Clients, includes OPC HDA (historical data access) support)
- DDE Server (to communicate with DDE Clients, primarily for legacy support)
- TCP/IP Server (to communicate with other IWS applications, also used to communicate tag data to Web Thin Clients)
- HTTPD (Web page support for Web Thin Client applications)

Built-In Clients

IWS provides a several built-in Clients used to support communications with other software applications that may be executing on the same PC (or device) or running on a remote PC (or device). These Clients include:

- OPC Client (to communicate with 3rd party OPC Servers)
- DDE Client (to communicate with DDE Servers, primarily for legacy support)
- TCP/P Client (to communicate with other IWS applications

Database Support

IWS provides built-in support to connect to relational databases via ODBC or ADO.NET. ADO.NET database support is directly built into the Alarm Manager, Event Manager, Trend Manager and the Grid Object. This allows a direct database connection to be established without the need for programming. ODBC commands (including SQL queries) are provided through IWS built-in functions, and with VBScript support, access to relational databases can occur via ADO.NET or ODBC. **Note:** ODBC is not available for Windows CE applications.

Database Redundancy

IWS provides built-in support for redundant databases using ADO.NET Providers. The use of redundant databases is optional. If used, the Secondary database can be used in a full redundant mode (mirror) or a store-and-forward mode, whereby if the Primary database becomes unavailable, data is stored in the Secondary database. The Primary database is resynchronized when it comes available.

VBScript Support

Beginning with IWS Version 6.1, IWS supports VBScript. VBScript offers several advantages; it is programming language instead of a Scripting Language, it is supported on Windows CE runtime platforms, and it can run on a Web Thin Client. VBA does not offer the same advantages as VBScript. IWS supports VBScript version 5.6 as well as Microsoft Intellisense. VBScript can access IWS tags and IWS built-in functions.

Database Spy and LogWin

The Database Spy and LogWin (Output Window) are key development and runtime diagnostic tools. They can work on a local machine or with a remote PC (or device). The Database Spy tool can be used to monitor or force tag values, while the LogWin tool monitors module activity (e.g. driver communications, TCP/IP, OPC, and DDE transactions, Security System Log In/Log Out's, etc.

ActiveX and .NET Container

IWS is an ActiveX Container and a .NET Container. This means that ActiveX Controls and .NET Controls can execute in the IWS runtime environment, and IWS can Set and Get various Contol Properties (interacting with IWS tags) or initiate various Control Methods. These Controls are typically inserted on a specific IWS screen.

Drivers

InduSoft provides a number of drivers that can be used to directly communicate to PLCs, RTUs and other plant floor devices. These are high speed drivers and are included with the software. (Updates to the drivers are available for free, downloaded from InduSoft's website at *www.indusoft.com*). You can select any of the drivers to use for your application. The total number of concurrent drivers supported depends on your license.

Microsoft and Industry Standards

IWS conforms to a number of Microsoft standards as well as Industry Standards. In addition, InduSoft is a Microsoft Gold Certified Partner. Standards that IWS supports include:

- DDE (not available for Windows CE)
- OLE
- COM/DCOM
- .NET Framework
- Active X
- XML
- SOAP
- OPC
- TCP/IP
- MFC
- ODBC (not available for Windows CE)
- ADO.NET
- SQL
- VBScript

Notes

Chapter 6. Creating and Opening Projects

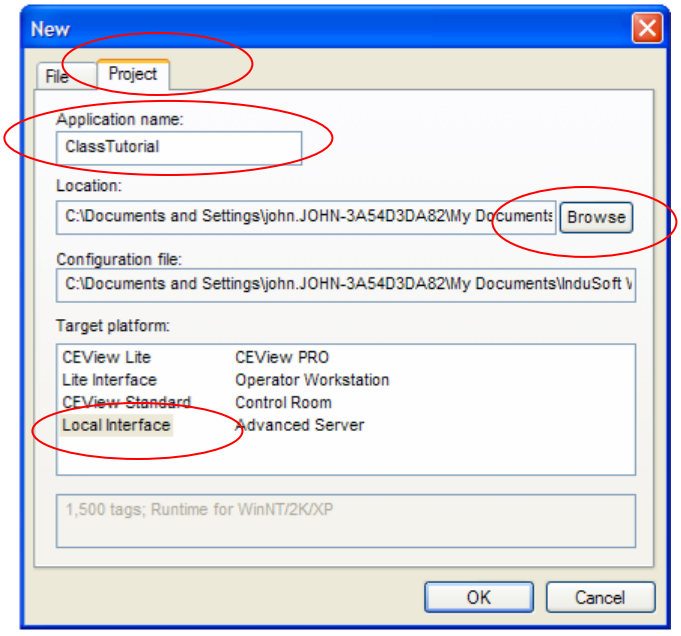
Creating a Project or opening a Project in IWS is a fairly straightforward task. There are only a couple steps to creating a new Project. Once a Project has been created, it can be saved and opened again at a later time, allowing for additional development or reconfiguration of the Project (e.g. redefining the screen size or the runtime environment).

Before creating a new Project, you obviously need to have InduSoft Web Studio (or CEView) development environment installed on your development PC. You should begin by starting the IWS development software. If IWS development has previously been run on your development PC, it will attempt to automatically open the last Project that was used. Otherwise, if this is the first time IWS has been run on your PC, it will not have any projects which to open. From there, you can create a new project or open an existing project.

Creating a new Project

Start the IWS development software and from the Main Menu Bar select **File → New.** The **New** dialog box will appear. Select the **Project** tab if it is not already selected. In the **Application Name** field, type the name of your new Project. The Folder (directory) to store your new application will be in the **Location** field, you can use this folder or specify another folder by selecting the **Browse** button. Your application file will be stored in the folder specified in **Location**, as a filename as specified in the **Application Name** field, with a .APP file extension. This is shown in the **Configuration File** field. Select your **Target Platform**, corresponding to a Windows XP/2000/NT/Server 2004/Vista platform or a Windows CE platform, along with the InduSoft license type to be used for this application.

Indusoft License	Max Tag Count	Windows CE	Windows XP/2000/NT Server 2003 Vista
CEView Lite	150	✓	
CEView Standard	1,500	✓	
CEView Pro	4,000	✓	
Lite Interface	150		✓
Local Interface	1,500		✓
Operator Workstation	4,000		✓
Control Room	64,000		✓
Advanced Server	512,000		✓



When you are done, select the **OK** button**.** You have now created a new Project.

At this point, don't worry whether you have enough tags to meet your application requirements. You can adjust the license configuration for your software by selecting from the Main Menu Bar the following:

Project → Settings → Options

Using the Target System combo box, select a new target platform configuration to meet your application requirements.

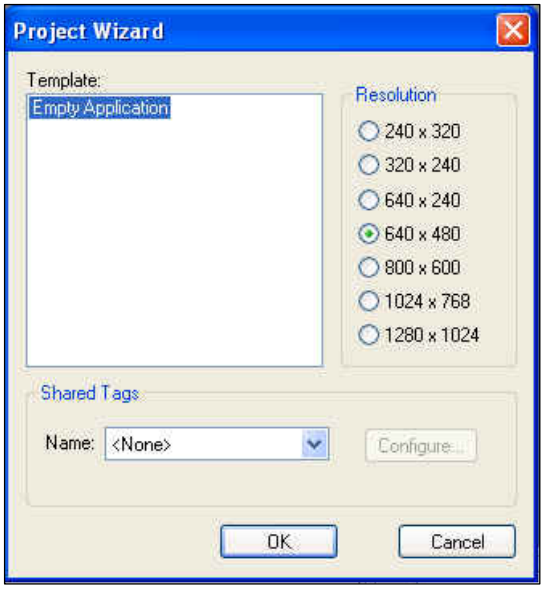
Notes:

- In the **New** dialog box, only the **Target Platforms** that your license authorizes you to use will be displayed (others will be grayed out).
- You cannot develop or open (in other than a Demo mode) a Project that uses a license configuration larger than what you are authorized to use.
- It is recommended to develop your application using the lowest license level possible. You can later change the license level for the target platform to a larger size if required.

Next, a **Project Wizard** dialog box will appear, as show to the right. Select the radio button corresponding to the screen resolution for your target platform and press the **OK** button.

The **Resolution** setting in the Project Wizard defines the default screen resolution for your application. As will be explained later, when you create a screen you have the option of defining the resolution of an individual Screen. If you are creating a Screen Group for an Multiple Document Interface (MDI) interface (i.e. a collection of smaller Screens that make up a Screen Group), it is quite normal that each screen is smaller than the default resolution. However, the collection of smaller (MDI) screens typically combines in a manner to fill the default screen size.

If you later decide to change the resolution of your target platform, this can be easily accommodated. In the Main Menu Bar under **Tools**, there is a **Convert Resolution** Tool that will allow you to change the screen resolution of your target platform.



Project Wizard Dialog Box

Note:

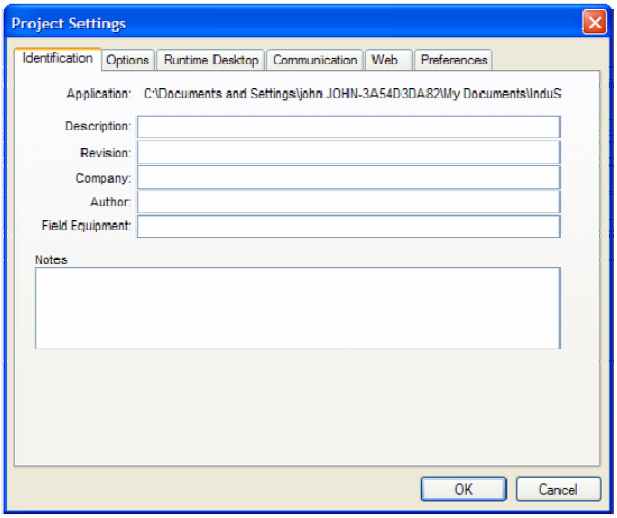
- The Project Wizard lets you define a PC-based Control program with which to share tags. However, be sure to verify the version level of the PC-based Control program supported by IWS. A more common method is to use OPC to communicate with the PC-based Control program.

You can now configure the Project Settings, used primarily to control the runtime attributes of the application. It is not necessary to configure these settings at this time, as this can be done later. However, it is a good idea to configure certain settings, such as database settings, at the beginning of your project.

Most of the configuration settings will be found by going to the Main Menu Bar and selecting **Project → Settings**. A new dialog box will appear, providing a number of configurations settings selectable through the tabs located on the top of the dialog box. These tabs include:

Identification

The **Identification** tab allows you to enter information to document your application. Fill in the appropriate fields and either select another tab on the top of the dialog box or press the **OK** button.

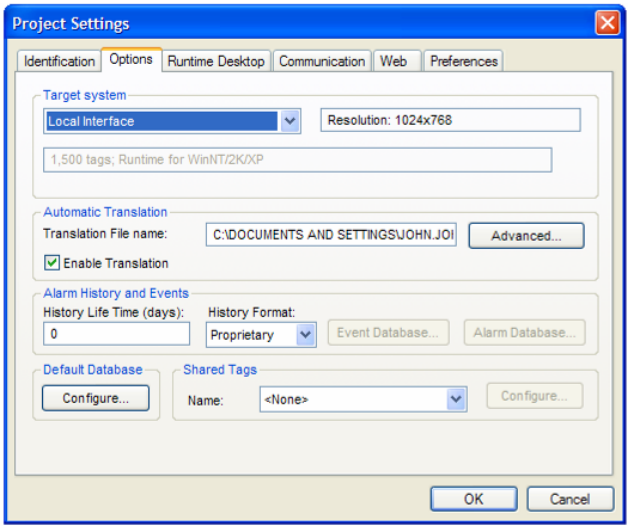


Options

The **Options** tab allows you to specify:

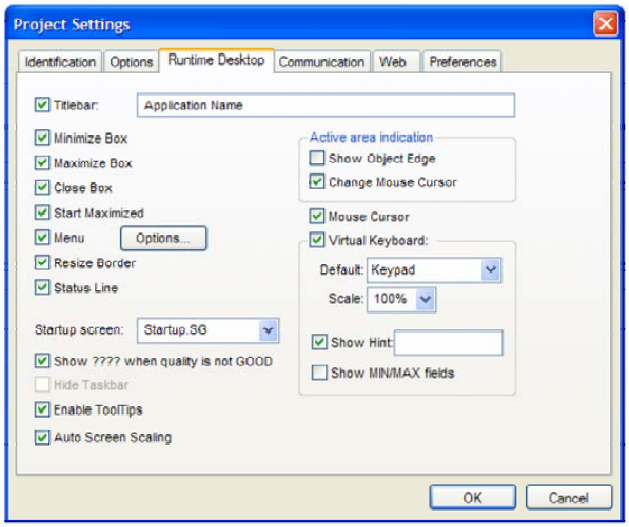
- The Target Platform
- Enabling of run-time translations and the Language Translation file to be used.
- The database settings for Alarm and Event historical data, and the length of time (days) to keep this data. The data can be stored in an IWS proprietary format or on a relational database.
- Default database configuration
- Shared tags (PC-based Control)

When this information is entered, either select another tab on the top of the dialog box or press the **OK** button.

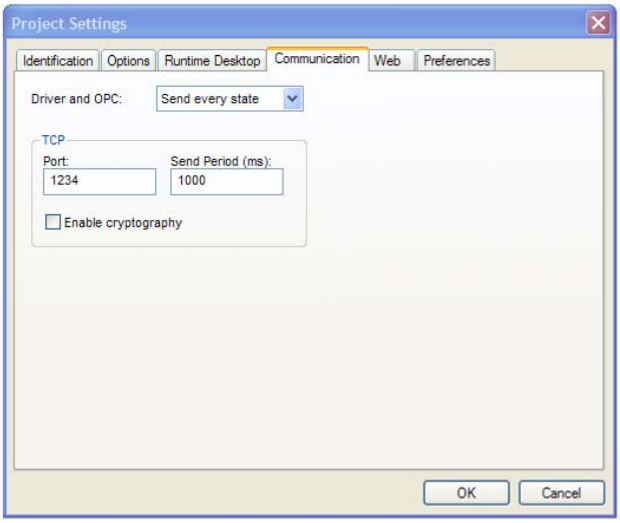


Runtime Desktop

The **Runtime Desktop** allows you to configure the look and functionality available on the runtime display (or Viewer), including the functions on the Menu Bar. The startup screen (or Screen Group) can be specified, along with the default Virtual Keyboard (popup keyboard) type. New with IWS 6.1 SP2 is the ability to specify a default **Hint** and the option to show the **Min/Max** numerical values when a numeric-style Virtual Keyboard is used to enter a numeric value for a tag.



Communication

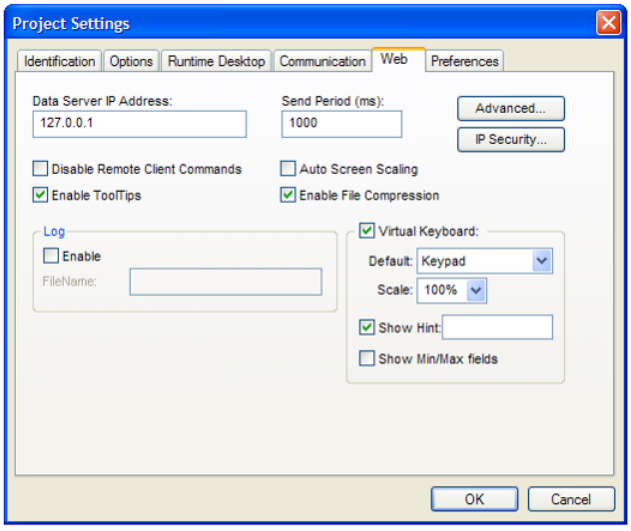


Communication settings can be set here. The **Driver and OPC** communication combo box defines the method to communicate to Drivers and OPC Servers. The options are to buffer all tag state changes, or to send only the last tag state to a driver or an OPC Server.

The **TCP Port** field defines the TCP port used by the TCP/IP Server and Client modules used in the local PC or device. This field must be identically set in other PC's or devices that communicate vial the TCP/IP Client or Server modules. Default is Port 1234.

The **Send Period** defines the period between consecutive messages between TCP/IP Client and Server modules. The lower the value, the lower the delay but more network traffic may result. **Cryptology** between TCP/IP Server and Client modules can be enabled, increasing communications security.

Web



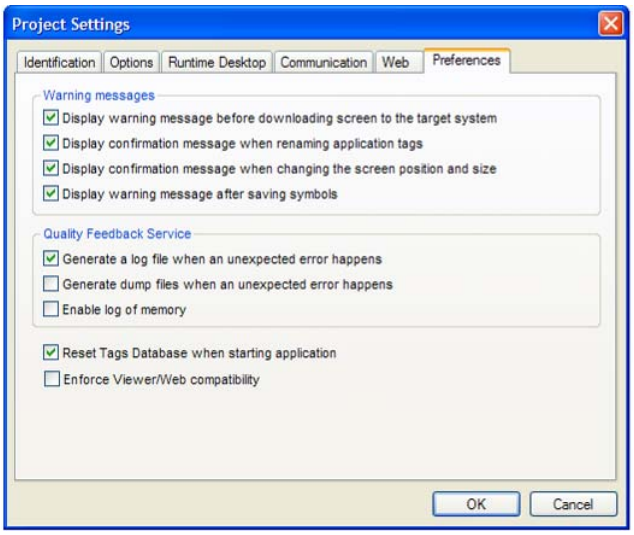
The **Web** settings define most of the configurations settings for Web Thin Client access. The **Data Server IP Address** field defines the IP address where the Web Thin Client looks for the Data Server. This is usually set to the IP address of the runtime PC (or device). You can select the **Auto Screen Scaling** check box if your Web Thin Clients use different display resolutions. The **Advanced** button allows you to define a secondary Data Server, backup URL and the URL to download ISSymbol (the ActiveX control used to enable communication with the Data Server and IWS Web Server , if it not already installed in the Web Thin Client). You can also enable Web Tunneling for working with a corporate firewall. **IP Security** lets you block IP addresses (or range of Addresses) to enhance security. The **Enable File Compression** checkbox tells the IWS runtime to compress the HTML files (located in the \Web folder in the Application directory), speeding downloads.

Note:

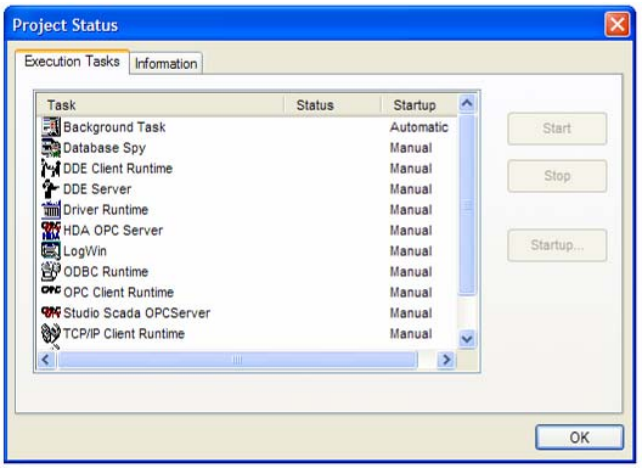
- In addition to the Data Server IP address, the Web Server will need to point to the IP address (or network location) of the HTML files for the IWS application.
- InduSoft provides light-weight Web Servers (NTWebServer for Windows XP/2000/NT/Server 2003/Vista and CEWebServer for Windows CE), but InduSoft recommends using a higher transactional duty Web Server for applications that support multiple Web Thin Clients (e.g. IIS or Apache for Windows).

Preferences

The **Preferences** tab lets you specify the warning messages that will display at development, let you specify how to display tag quality, and control some operational items (i.e. resetting tag values every time the runtime is started, and enforcing compatibility of functions used on a Server screen with a Web Thin Client (e.g. some functions only work in the Server or the Web Client)).



Another configuration area that likely need to be set up is the **Execution Tasks** tab of **Project Status** dialog box. This dialog box lets you define which IWS Tasks are to startup automatically (or manually) when the runtime is started on the target system. To get to this dialog box, from the Main Menu Bar select **Project → Status** and then select the **Execution Tasks** tab.



At a minimum, the **Background Task** and the **Viewer** tasks should be set to Automatic. Depending on your application, other tasks should be set to Automatic, so that they automatically start when the runtime is started. To change to automatic, select the Task, and then click on the **Startup** button. A pop-up display will let you change the startup status between **Manual** and **Automatic**.

Notes:

- If you are using an IWS Driver to communicate to a PLC, RTU or other plant floor device, be sure the **Driver Task** is set to **Automatic**.
- Normally, when a Task is started, its status will be displayed as **Started**. However, the Driver Runtime will not be displayed as **Started**. Instead, individual icons for each driver running will be displayed as an individual icon in the Windows Taskbar. You can right mouse click on the individual driver icon in the Windows Taskbar and select **Exit** to stop the driver.
- Do not start any unnecessary (i.e. unused) tasks as this may impact your runtime performance.

These Execution Tasks and their functions are as follows:

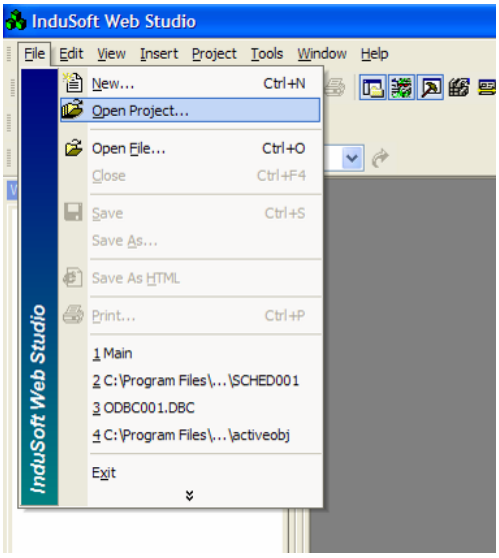
Task	Default Status	Available for Windows CE	Notes
Background Task	Automatic	Yes	The background supervisory task that runs all other tasks. Set to automatically start
Database Spy	Manual	No	Allows the Database Spy to work with the runtime module(s)
DDE Client Runtime	Manual	No	Used to communicate with a DDE Server (local or remote).
DDE Server	Manual	No	Used to communicate with one or more DDE Clients (local or remote)
Driver Runtime	Manual	Yes	Manages read & write commands in the Driver worksheets. Needs to be enabled when using one or more drivers.
HDA OPC Server	Manual	No	Enables IWS to be an Historical Data Access OPC Server
LogWin	Manual	No	Enables LogWin function (supports trace messages from modules/tasks).
ODBC Runtime	Manual	No	Enables IWS to run ODBC
OPC Client Runtime	Manual	Yes	Enables IWS runtime to be an OPC Client with either a local or remote OPC Server.
Studio Scada OPC Server	Manual	Yes	Enables IWS runtime to be an OPC Server to a local or remote OPC Client.
TCP/IP Client Runtime	Manual	Yes	Enables tag data exchange with an remote IWS Server. Used by an IWS application.
TCP/IP Server	Automatic	Yes	Used to serve tag data to remote IWS runtimes and to Web Thin Clients.
Viewer	Automatic	Yes	Used for Screen display and Screen Scripts

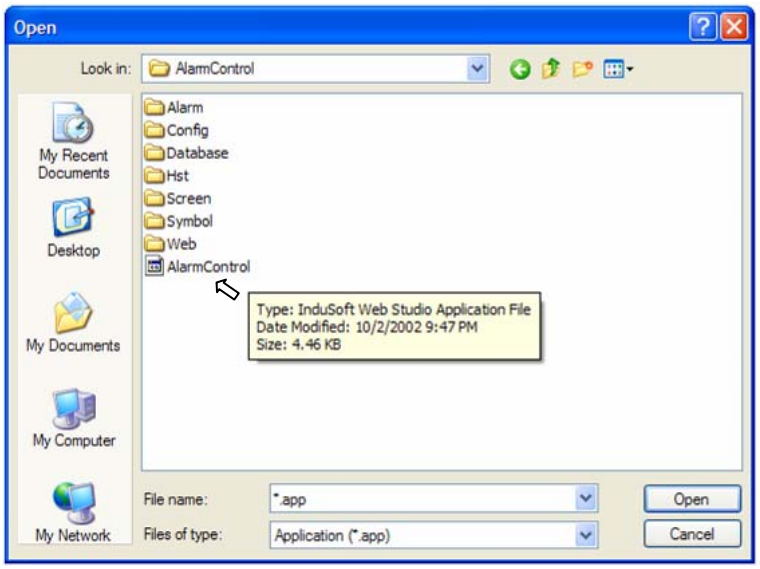
Saving your new Project

Saving your new Project is easy. You should save changes to and Screen or Worksheet. But if you attempt to Exit (i.e. exit by selecting from the Main Menu Bar **File → Exit**) or by Closing the IWS development environment (i.e. select the Close button in the top right corner of the IWS development environment window), you will be offered the option of saving any changes before the IWS development environment closes.

Opening an existing Project

Opening an existing Project is easy. From the Main Menu Bar, select **File → Open Project.** A standard Windows Open dialog box will appear, allowing you to navigate throughout the file system that is available to you. You need to select an IWS application file (the file extension is .app). Next, select **Open**. The IWS application will then open and you can begin any development work.

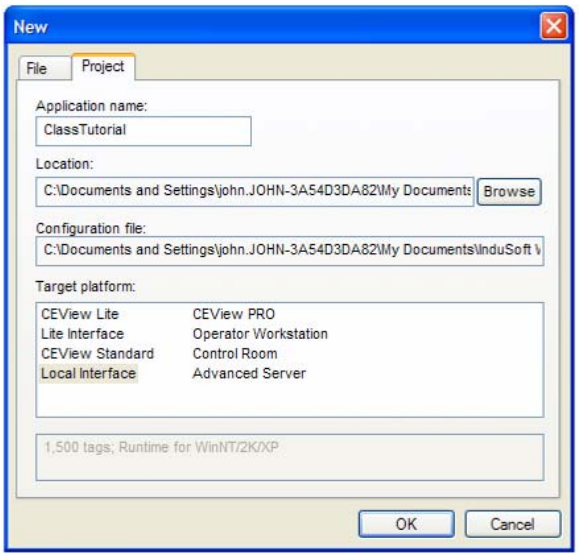


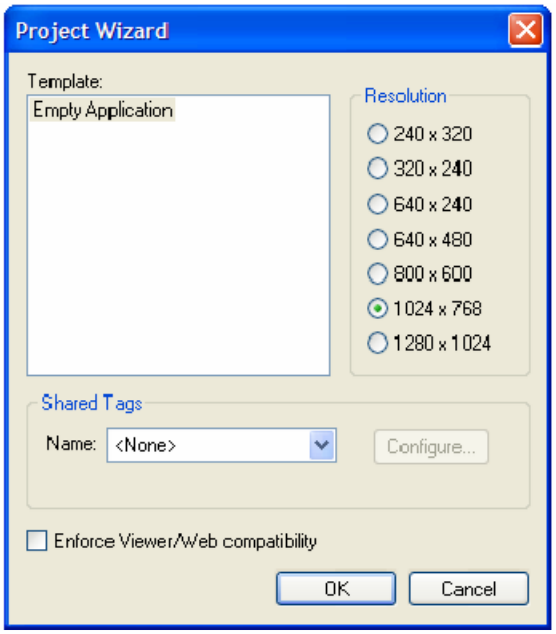


Exercise: Creating a new Project

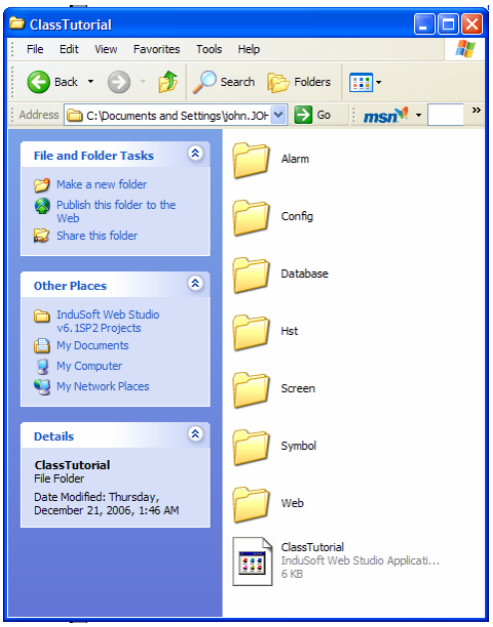
In this example, we will create a new Project that we will use during this class. Implement the following steps:

- ☐ Be sure IWS Version 6.1 SP2 is installed on your PC.
- ☐ Start the IWS development software.
- ☐ From the Main Menu Bar, select **File → New**.
- ☐ When the **New** dialog box appears, click on the **Project** tab.
- ☐ Type **ClassTutorial** in the **Application name** field. This will be your Project name. A folder named ClassTutorial will be saved in the directory path specified in the **Location** field, and in the folder **ClassTutorial** will be the application file ClassTutorial.app and several other subfolders that will store application specific files..
- ☐ Verify that **Local Interface** is selected in the **Target Platform** pane. This will provide the application with an initial tag limit of 1,500 tags.
- ☐ Click on the **OK** button.
- ☐ When the **Project Wizard** dialog box appears, select the 1024 x 768 resolution radio button and select **OK**.



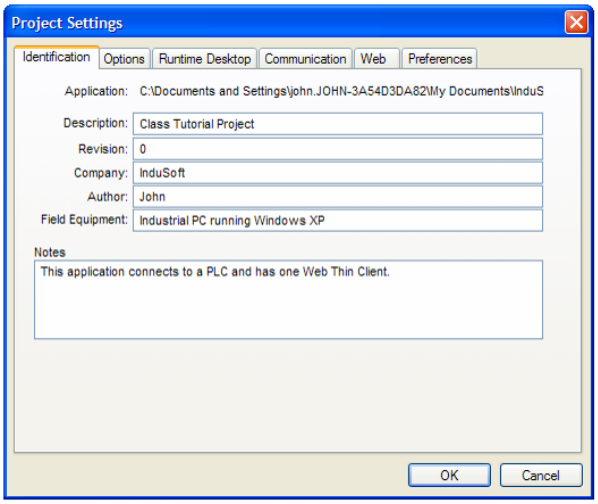


- Note that the folder created for our application **ClassTutorial** contains several subfolders (**\Alarms, \Config, \Database, \Hst, \Screen, \Symbol** and **\Web**). These subfolders contain various files used with the application.



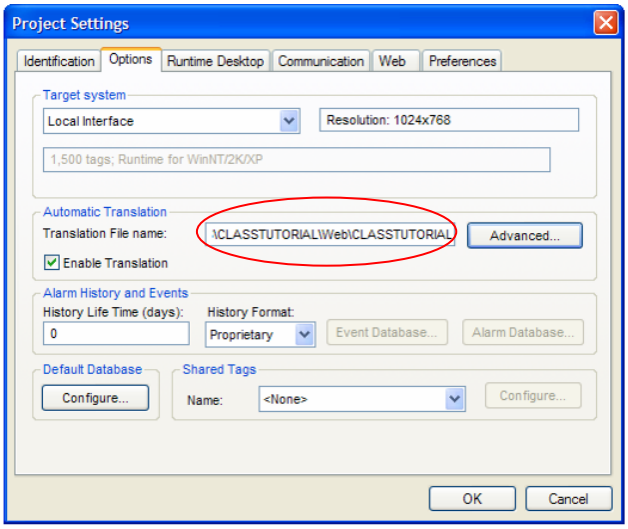
Now that we have created our first Project, we will configure some of the Project settings. We will not configure all of the runtime settings at this time, but as our application is developed, we will come back to the Project Settings and specify additional settings to be used.

- From the Main Menu Bar, select **Project → Settings.** This will open the **Project Settings** dialog box.
- Select the **Identification** tab
- Fill in the various fields. You can use your own information and name. This information is for documentation purposes only.

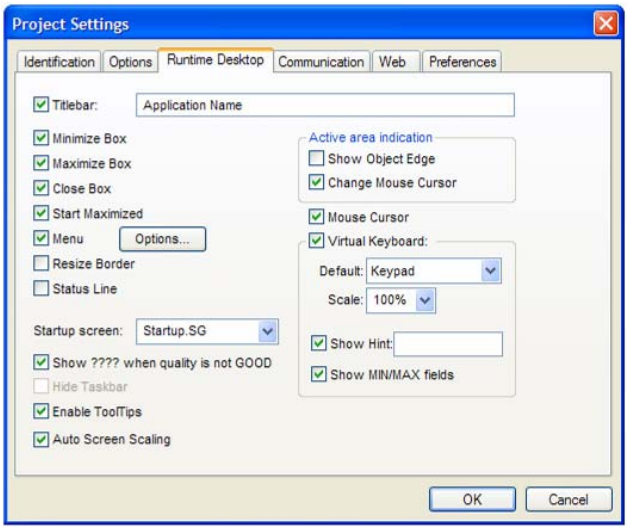


- ☐ Select the **Options** tab.
- ☐ In the **Translation File Name** field, note that IWS has automatically defined a path for our runtime translation file. The default name for this file is **ClassTutorial.csv**, and is located in the \ClassTutorial folder. We want to change this to the **\Web** subfolder located in the \ClassTutorial folder. This can be easily done by inserting **\Web** before **\ClassTutorial.csv** at the end of the **Translation File Name.**

  Note: As will be discussed in more detail later, the reason for doing this is so runtime translation can be used by Web Thin Clients as well as the Server.
- ☐ Click on **OK.**

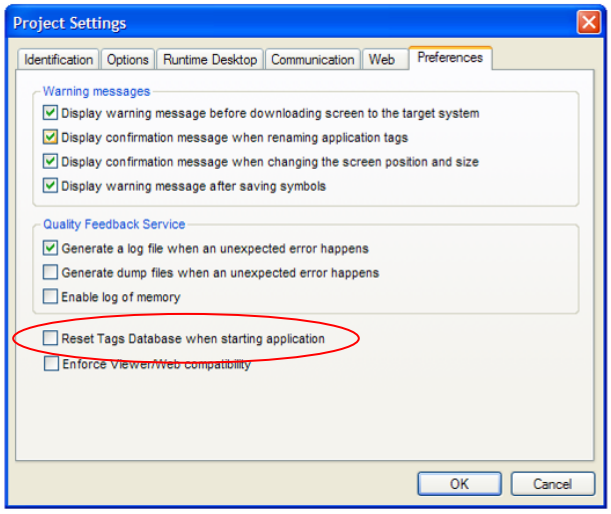


- ☐ Select the **Runtime Desktop** tab.
- ☐ Select the following checkboxes:
  - Titlebar
  - Minimize box
  - Maximize box
  - Close box
  - Start Maximized
  - Menu
  - Show ???? when quality is not GOOD
  - Enable ToolTips
  - Auto Screen Scaling
  - Change Mouse Cursor
  - Mouse Cursor
  - Virtual Keyboard
  - Show Hint
  - Show Min/Max fields



- ☐ When we create application Screens, we will create a startup Screen Group called **Startup.SG**. This can be specified now in the **Startup Screen** field, or later when the Screen Group is created.
- ☐ Click on **OK**.

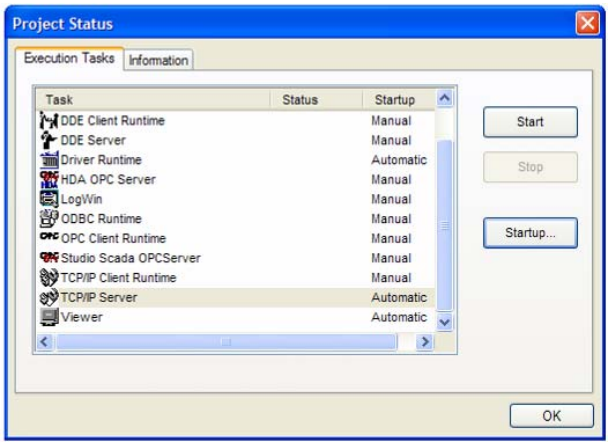
- ☐ We will leave the **Communication** and **Web** settings unchanged for now
- ☐ Select the **Preferences** tab.
- ☐ The checkboxes shown should be checked by default.
- ☐ Note than the Reset Tags Database when starting application checkbox is unchecked. As we will later see, we can specify a startup value for a tag. We may or may not want to reset the database every time we start the application, especially during development.
- ☐ Click on **OK**.

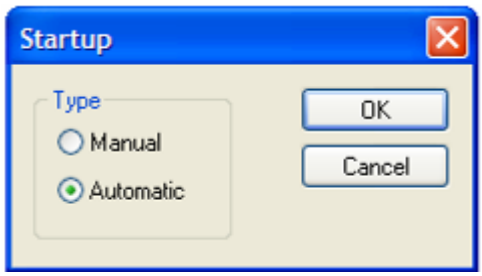


At this point, we are almost done with the initial application settings. The last step is to define the runtime Execution Tasks that need to start automatically. Since we defined our application as connecting to a PLC and having one Web Thin Client, we need to automatically start the **Driver Runtime** and the **TCP/IP Server.**

- ☐ From the Main Menu Bar, select **Project → Status** to open the **Project Status** dialog box.
- ☐ Select the **Execution Tasks** tab.
- ☐ Make sure the **Background Task** is set to Automatic
- ☐ If the **Driver Runtime** task is not set to Automatic, select the **Driver Runtime** task and click the **Startup** button. When the **Startup** dialog box appears, select the **Automatic** radio button and press **OK.**
- ☐ If the **TCP/IP Server** task is not set to Automatic, select the **Driver Runtime** task and click the **Startup** button. When the **Startup** dialog box appears, select the **Automatic** radio button and press **OK.**
- ☐ Click **OK** to close the **Project Status** dialog box.

We have just created our first IWS Project !!





Notes:

- The **Execution Tasks** tab is only available when the Target Station option from the Execution Environment dialog box (**Project → Execution Environment)** is set to **Local**. With CEView applications, you cannot modify the Startup setting for each task (Automatic or Manual)
- The **Status** field in the **Execution Tasks** dialog box monitors the current status of each runtime task.
- The **Stop** and **Start** button can be used to control the status of the runtime task.

Notes

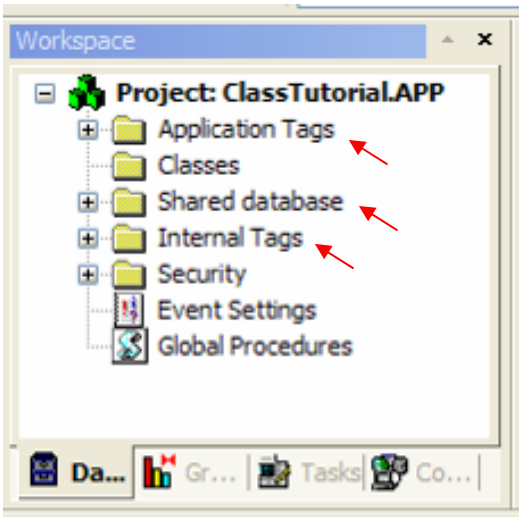
Chapter 7. Tags

Overview

Tags are a core component of any InduSoft Web Studio (IWS) application. From a simple viewpoint, tags are variables used by IWS for a variety of purposes, such as to store the current data points obtained from communication with plant floor devices (e.g. PLCs) and storing the results of calculations. Tags are used to display information on screens (and web pages), to manipulate IWS Objects, and with worksheets (Math, Scheduler, Alarm, Trending, Event Management, Communications, Recipes, and Reports).

But tags are more than simple variables. IWS includes a real-time database manager for the tags that provides a number of sophisticated functions such as time stamping of any value change, checking tag values against runtime minimum and maximum values, comparing tag values to alarming limits, etc. An IWS tag has both a value and various properties that can be accessed, some at development and others only at runtime. .

Tag Categories



InduSoft Web Studio defines three (3) different tag categories within an application. These are:

- **Application tags** are tags you can create during application development. Example where application tags are used include:
  - Displays
  - Tags that read from/write to field equipment
  - Control tags
  - Auxiliary tags used to perform mathematical calculations
- **Internal tags** are predefined tags with predetermined functions that are used within InduSoft Web Studio's supervisory tasks. Most internal tags are *read-only*. Example internal tags include
  - Date tags, which holds the current date in string format
  - Time tags, which holds the current time in string format.

  Note: Internal tags do not count against the IWS application tags license limit
- **Shared tags** are created in a PC-based control software program and imported into InduSoft Web Studio. You cannot modify these tags within IWS, but you can modify these tags in the PC-based control software, then you can update and use them in the IWS *Application* database.

  Note: Shared tags do not count against the IWS application tags license limit

All tag definitions are found in folders located in the Database Tab of the IWS Workspace. Application tags are declared in the **Application Tags** database folder.

Tag Nomenclature

When defining application tags, you must adhere to the following syntax rules:

- You must begin the tag with a letter. After that, you can use any combination of letters, numbers, and the underscore character (_).
- You can use a maximum of 32 characters (for a tag name) or 16 characters (for a class member name).
- Prohibited symbols in the tag name are ` ~ ! @ # $ % ^ & * ( ) - = \ + \ [ ] { } < > ?
- Application tag names must be unique from all other tag names and IWS built-in function names

- You can use upper- or lowercase letters in a tag name. Tag names are not case sensitive, and you can use both upper and lowercase characters to make tag names more readable (for example: TankLevel instead of tanklevel).

IWS Tag Types

IWS allows you to define the following standard tag value types:

Boolean: (One Bit). A digital value of 0 or 1 representing **False** or **True.**

Integer (Four bytes): Signed integer number (positive, negative, or zero). Equivalent to the C-type signed long integer. (with a range of –2147483647 to 2147483647).

Real (Floating point, eight bytes): Real number internally stored as a double word. Equivalent to the C-type double precision floating point.

String (alphanumeric data, up to 256 chars): Character string up to 255 characters (from 0 to 254) that holds letters, numbers, or special characters. IWS supports both ASCII and UNICODE characters.

Array: An array tag is a set of tags of the same data type, with the same name, that use an index to uniquely identify an item. IWS supports 1-dimension arrays only.

Class: A class tag is a compound tag type that has been defined by a class template. A class template can consist of one or more data types. Class tags are helpful when developing applications that have items (e.g. tanks) that have multiple properties being monitored or controlled (e.g. fill level, temperature, pressure).

Pointer: A pointer tag lets you indirectly refer to another tag in the IWS database.

Notes:

- Regardless of the number of bytes required to store a tag value, a Boolean, Integer, Real, String or Pointer tag counts as only one tag.
- VBScript Strings and IWS Strings are not equivalent. IWS tags are a maximum of 255 characters in length, while there is no limit to the length of a VBScript string.

Tag Scope

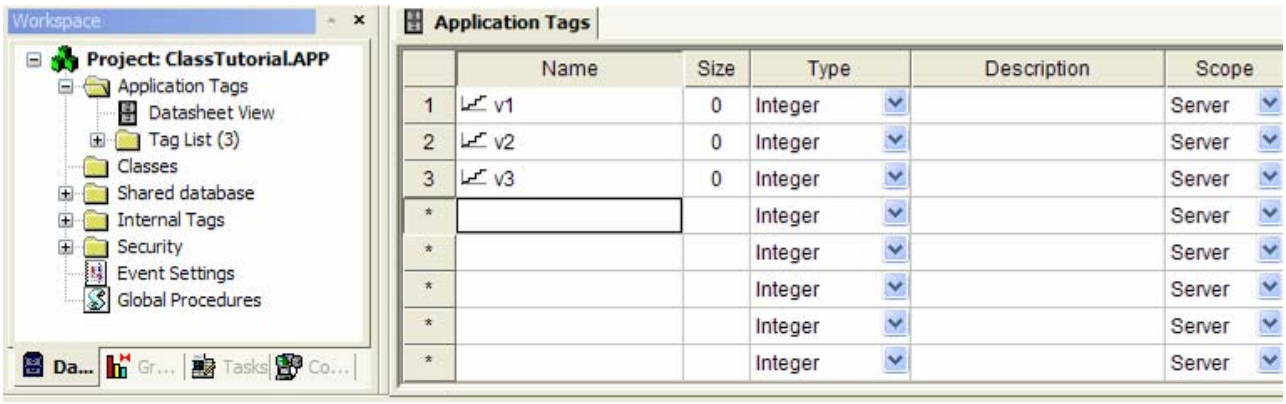
All tags have Scope, the Scope types being either **Server** or **Local**. If a tag is defined as having **Server** Scope, this means that the tag is accessible both to the local PC (that is executing the IWS application runtime) and to any Web Thin Clients connected to the local PC (called the Server). If a tag is defined as having **Local** Scope, this means that the tag is only accessible (usable or viewable) from the local PC (the Server).

Notes:

- IWS Application Tags can be defined as having **Local** or **Server** scope. This determines the usability or ability to view on a PC other than the Server.
- If you want to remotely acknowledge alarms (via a Web Thin Client), make sure that any tags associated with the alarm have their **Scope** set to **Server**.
- IWS Internal Tags have their scope predefined and cannot be altered.

Declaring Tags

To declare a tag, you open the Application Tags folder in the Database Tab of the Workspace. This is where all Application tags can be declared. In the **Name** field, you enter a unique tag name. In the **Type** combo box, click on the down button and select the tag type (Boolean, Integer, Real or String). An optional **Description** field is provided to allow for a short description of the tag to be entered. The **Scope** field allows for setting of the tag scope (Server or Local). The **Size** field defaults to 0, meaning the tag is a single (non-array) tag. If the value in the Size field is greater than 0, this defines a tag array (discussed further in the section below).



Declaring Tags

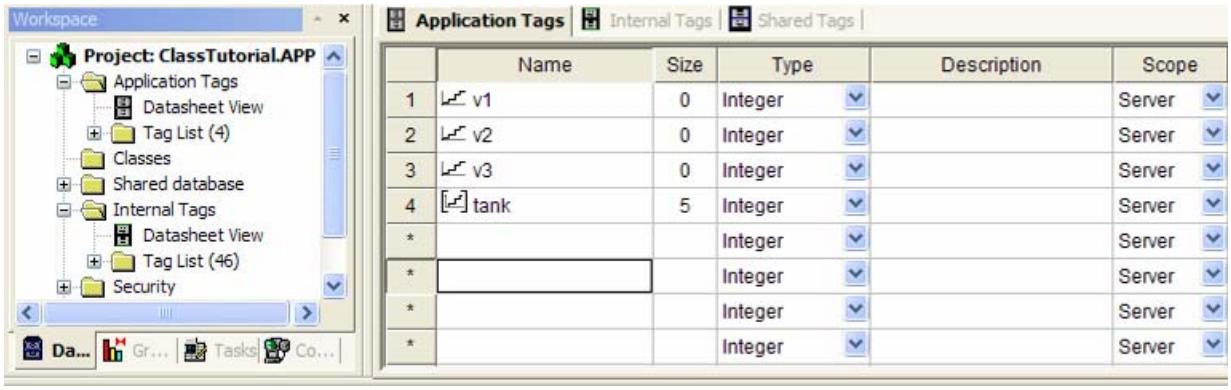
Notes:

- When declaring Application tags, it is not necessary to use the combo box for defining the **Type** or **Scope** fields. After typing in the tag name, simply press the tab key twice to tab over to the **Type** field and press B for Boolean, I for Integer, R for Real or S for String. Press the Tab key twice again to tab to the **Scope** field and type S for Server, or L for Local.
- If you place your mouse cursor on the blank square in the upper left corner (left of the Name field) and click the right mouse button, you can sort the tags by name in ascending or descending order.
- If you want to delete or insert tags, first place your mouse cursor on the blank square in the upper left corner (left of the Name field) and click the right mouse button. Select the **Disable Sort** (if it is not grayed out). Next, position your mouse cursor over the line you want to delete, or the line to insert a new line above, and click the right mouse button. Select **Delete Line** or **Insert Line**.
- **Note:** You will not be able to Insert or Delete a line if the IWS application is running. Click on the **STOP** icon in the main Toolbar to stop the IWS application.
- You can select which fields to view in the Application Tags database Datasheet View. Simply put your mouse cursor on one of the lines, and click the right mouse button. The Name, Size, Type, Description and Scope fields are the default fields to be displayed, but by selecting **More Columns** you can view other tag (Property) fields that can be specified during development. Tag Properties will be discussed in a later section.
- Declaring tags in the Application Tags database is not the only method of declaring IWS tags. As we will see later, with most Objects you can simply type in the name of a tag and if the tag is not already defined, IWS will ask you if you want to declare the tag. This allows you to create new tags while developing your application without first declaring them in the Application Tags database. This method is suitable for declaring tags of data type Boolean, Integer, Real, String and Array.
- In the Datasheet View, each tag will have an icon proceeding the tag name, signifying the data type of the tag (e.g. Boolean, Integer .. etc.).

IWS Arrays

An **Array** tag consists of a set of tags that are all of the same data type, have the same name, and use an index to uniquely identify a tag that is an element (member) of the array. The use of array tags can simplify the method used to access multiple data items that are of the same type (e.g. recipe names, report names, communication points, etc.). Array tags can easily be declared, saving development time.

To define an array tag, you open the Application Tags folder and select Datasheet View. Type the name of the array tag in the **Name** field, select the array tag's data type (all elements in the array must be the same data type) in the **Type** field, and specify the size of the array (maximum number of elements) in the **Size** field. You may also want to enter a tag description and set the array tag scope. Note that all IWS arrays are 1-dimensional only, and a zero-based arrays (meaning the first element in the array is index 0).



Declaring Array Tags

The above graphic shows how an Integer Array tag called **tank** is declared. The size value is set to 5, and since all Arrays are zero-based, there are really 6 elements in the array [0, 1, 2, 3, 4, 5]. To reference a specific element in the tag, you would use (in an Object or one of its Dynamic Properties) the following syntax: **Array[Index]**, e.g. **tank[0], tank[1], .. tank[5].** Any index values greater than 5 would be invalid for the tank array since it was declared as Size 5.

Notes:

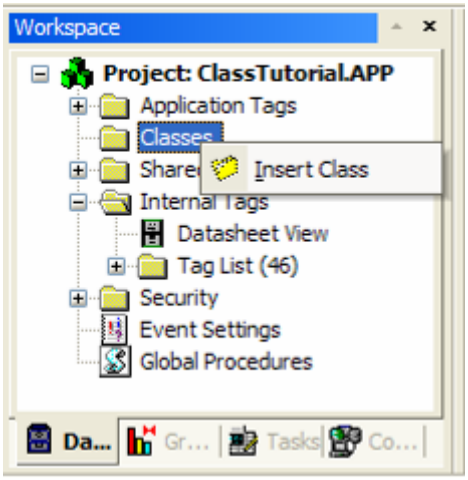
- The Array name must meet the same naming syntax requirements as all other IWS tags
- All Arrays are zero-based (i.e. first index is 0). Many applications do not use the 0 index, to simplify the application. This is up to the developer.
- No negative index values for the Array are allowed.
- The array index can be a simple numeric value, a tag or an expression. The tag or expression value cannot exceed the maximum array size, otherwise an error will occur.
- IWS uses square braces **[ ]** to surround the Array index of an IWS Array tag. This is different than VBScript, which uses parentheses **( ).**

IWS Class Tags

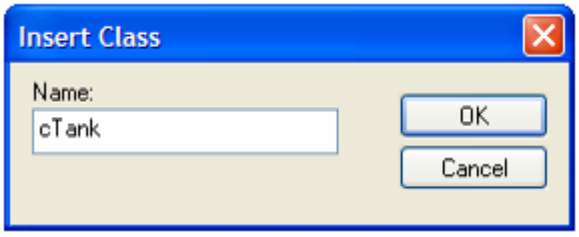
Class tags can be a confusing at first, but are really a simple, yet powerful concept to describe logical groupings of data. Before we cover the mechanics of creating and using class tags, let's touch on where they might be used. For example, let's assume we have an application that has a storage tank. The tank may have certain parameters associated with it:

- Name
- Physical location
- Capacity
- Level
- Temperature
- Pressure

In this example, Name and Physical Location would be stored as String tags, while Capacity, Fill Level, Temperature and Pressure would be stored as Real tags.

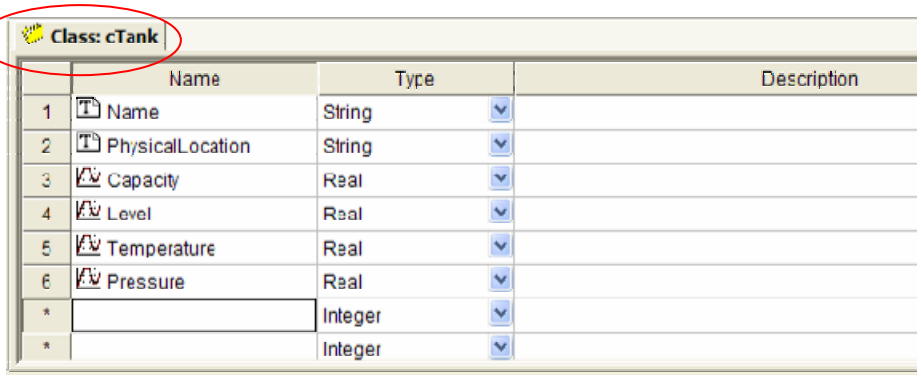


To create a Class tag for the storage tank, we will first need to define a **Class,** which is merely a template or data structure. In the Database Workspace, position the mouse cursor over the **Classes** folder and click the right mouse button. Click the left mouse button on **Insert Class**. A dialog box will appear asking you to define the Class. In this example, we will use cTank (small c to designate Class).



After clicking on **OK**, IWS will open a new worksheet that lets you define the elements in the Class **cTank**. These elements are collectively called **Members,** with each element being called a **Member**. The Members are not tags, they are simply the definitions for the elements that define a Class. Note that in this example, we have defined the **Names** and **Type** of the Members as defined above. An optional **Description** field is available.

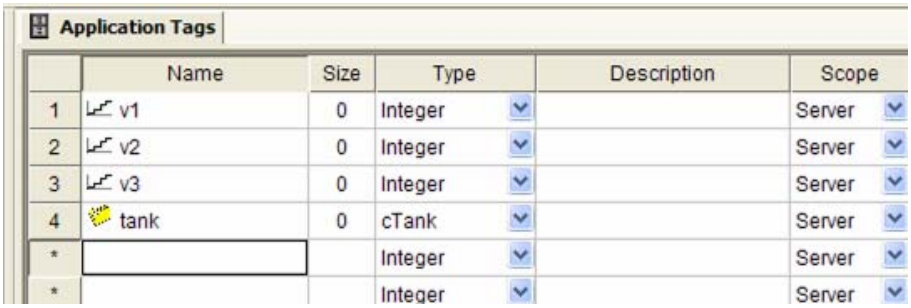
When finished, place mouse cursor over this icon, left click and select **Close**.



Once you have defined the Class Members for **cTank**, close the Class definition Worksheet by selecting (from the Main Menu Bar) **File → Close**, or position your mouse cursor over the **Class:** icon in the upper left corner of the Class definition Worksheet, right click and select Close.

We have just completed the first part of defining a Class tag, i.e. defining the Class. The next step is to declare the Class tag in the Application Tags database. Open the Application Tags folder and select Datasheet View.

As shown in the graphic below, we have now defined a Class tag called **tank** and its data type is **cTank**, the newly defined Class definition.



Declaring a Class Tag

Now that we have defined the Class **cTank** and declared Class Tag **tank**, let's look at the syntax for using a Class Tag. The syntax is: **ClassTag.Member** (e.g. tank.Name, tank.PhysicalLocation, tank.Level, etc.).

Suppose we had five (5) tanks instead of a single Tank. We could declare five Class tags of type cTank (e.g. Tank1, Tank2, ..Tank5). But a better way might be to declare an Array of Class tags. Using the above example, we simply add a size of 5 to the **Size** field as show below.

Declaring an Array of Class Tags

Note that since Size is set to 5, there are actually six (6) elements to Array of Class tags. To access a specific tank in the Array of Class tags, we would use the following syntax: **ClassTag[index].Member** (e.g. tank[1].Name, tank[2].PhysicalLocation, etc.)

Notes:

- The steps to creating Class Tags are:
  - Add a new Class definition. (You may want to use a "c" in the first part of the Class name)
  - Define the Class Members (name & data type). Close the Class definition worksheet.
  - Open the Application Tags database and add a Class tag, using the Class definition as the data type.
  - If you want to make the Class Tag an Array of Class Tags, set the Size field to a number > 0
- Class Members (defined in the Class definition worksheet) are simple tag types (Boolean, Integer, Real or String). Arrays and Classes are not allowable as part of the Class Member definition.

IWS Pointer Tags

Pointer tags allow you to indirectly reference tags in the IWS tag database. In reality, Pointer tags are not widely used, but with certain Objects such as the Alarm/Event Control Object, Pointer tags are very useful. [e.g. The Alarm Control Object allows you to acknowledge a selected tag. The selected Alarm/Event returns the name of the tag that triggered the Alarm/Event, and the returned name (in a string) can be used as an indirect reference to the actual tag].

Pointer tags can be used to point to any other type of tag (e.g. Boolean, Integer, Real, String, Array, or Class tag). The benefit of Pointer Tags is that they let you point to any other type of tag, keeping you from having to duplicate tags. There are two (2) ways to declare Pointers. Which method you use is up to you, although Pointer tags based on the String data type are arguably more flexible.

Method 1: Using a Pointer that is a String data type

With this type of pointer, we do the following:

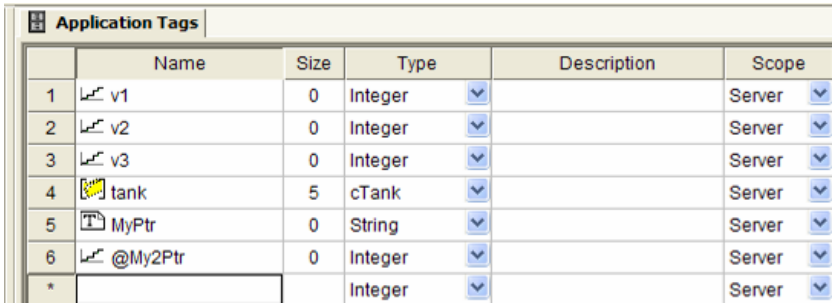
- In the Application Tags database, define a string tag. E.g. **MyPtr**
  Note: It may be helpful to use **Ptr** as a prefix or suffix to signify the tag is a Pointer tag.
- In your application, a Text/IO Object, IWS math worksheet or VBScript code segment would set the Pointer (String data type) to a string value corresponding to the name of a tag you want to reference.
  E.g. **MyPtr = "v1"**
  Note: A Pointer based on the String data type must be used in the Alarm/Event Control Object in the **Selected Tag** field (Advanced Settings).
- When referencing a tag with the Pointer tag, use a **@** in front of the Pointer tag.
  E.g. **v2 = @MyPtr** (this takes the v1 tag value and stores it in tag v2)

Method2: Using a Pointer that is of the same data type as the tag it is indirectly referencing

With this type of pointer, we do the following:

- In the Application Tags database, define a tag that is of the same type as the tag you want to reference. Put a **@** character in front of the pointer tag name. E.g. @My2Ptr
- **Note:** It may be helpful to use **Ptr** as a prefix or suffix to signify the tag is a Pointer
- In your application, a Text/IO Object, IWS math worksheet or VBScript code segment would set the Pointer (String data type) to a string value corresponding to a tag name. The tag name is for the tag you want to reference.
  E.g. **My2Ptr = "v3"**
- When referencing a tag with the Pointer tag, use a **@** in front of the Pointer tag.
  E.g. **v2 = @My2Ptr** (this takes the v3 tag value and stores it in tag v2)
  Note: With this type of Pointer, the tag being indirectly referenced and the Pointer tag data type must be the same.

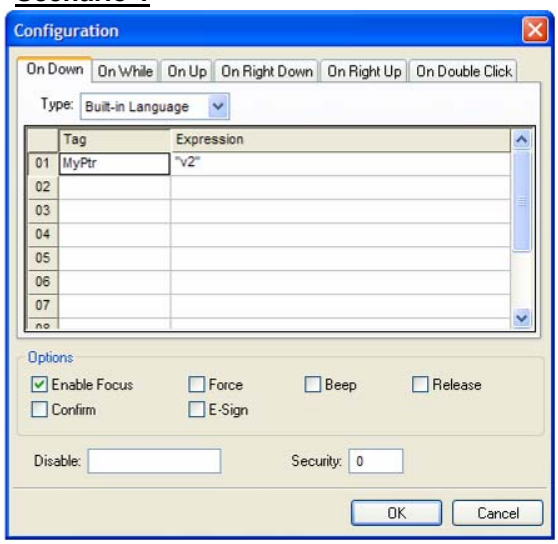
The following is an example of a Text Static Object with a Text I/O Dynamic Property used to display a tag value pointed to by a Pointer tag, and a Command Property (e.g. used with a Button Static Object) to specify which tag the Pointer tag is to indirectly point to.



Declaring a Pointer

As mentioned, Pointer tags can reference Class tags or an Array of Class tags. The Pointer tag type must be either a String tag or a Class tag of the same Class definition as the tag it is indirectly going to point to. Suppose we use our **tank** Class tag and the **MyPtr** Pointer tag (string type). To reference the Class tag Member **tank[2].Temperature**, we would use:

Scenario 1



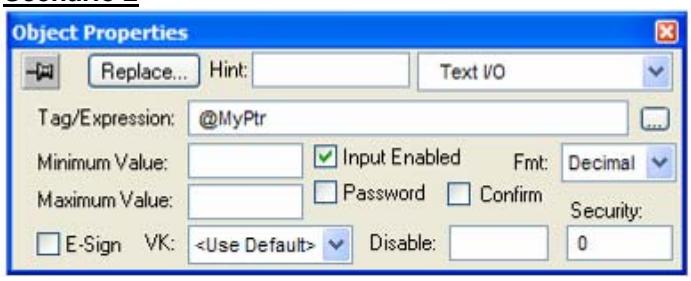
Using the IWS built-in language worksheet in a Command Property to specify which tag to indirectly point to.

If **MyPtr** is set to a value of “tank”

then to reference **tank[2].Temperature**, we use:

@MyPtr[2].Temperature

Scenario 2



Using a Pointer tag in a Text/IO Property to display the value of the tag indirectly pointed to

If **MyPtr** is set to a value of “tank[2]”

@MyPtr.Temperature

Notes:

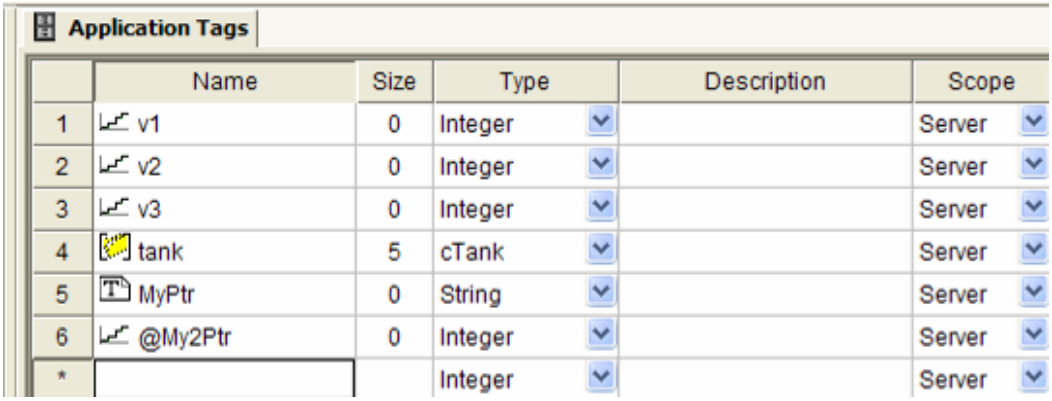
- Use Pointer tags that are of a String data type unless required to otherwise do so. They are much more flexible.
- To use Pointers with VBScript, be sure to enclose the tag name string in double quotes **“ “**
- When an application initially is started, until the Pointer tag is set with a value to indirectly point to a valid tag, any Object that uses the Pointer tag will reference an invalid value. If the Pointer tag is used in a display type Object will display **??** . This situation can be avoided by specifying an initial value to the Pointer tag when the application runtime starts (e.g. in a Math Worksheet, Background Script, Graphics Script or Screen Script).

Determining Tag Usage

Since IWS (and CEView) are licensed based on tag count, it is important to understand how tags are counted and how many tags are required (or used) for your application. The various tag limited licenses were covered in Chapter 2 of this Training Manual. All Application Tags, whether active or not, count against the license limitation.

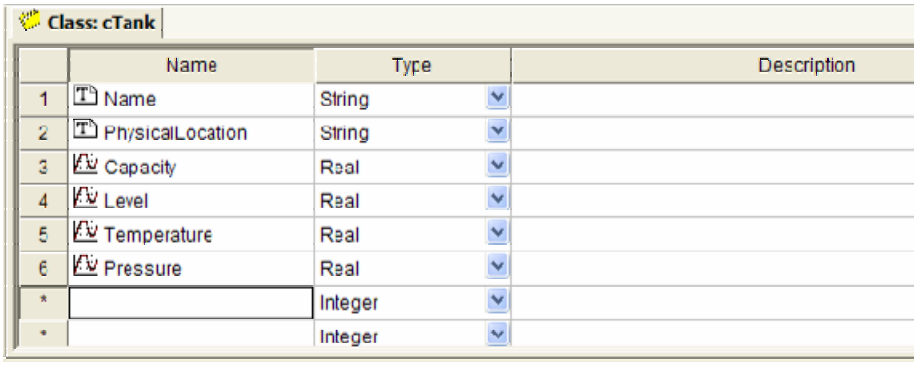
Tags have two primary purposes; to store values corresponding to real-world data (e.g. PLC register values, RTU values, and other device input/outputs) and to store intermediate results used for processing and display purposes (e.g storing temporary data). As a rule of thumb, you should use a runtime license that supports a tag count equal to 1.5 - 2 times the number of I/O points required. Of course, this can vary widely based on your application.

From the examples in the previous section, we had defined some tags in the Applications Tags database. Let's look at the tags required for this application.



	Name	Size	Type	Description	Scope
1	v1	0	Integer		Server
2	v2	0	Integer		Server
3	v3	0	Integer		Server
4	tank	5	cTank		Server
5	MyPtr	0	String		Server
6	@My2Ptr	0	Integer		Server
*			Integer		Server

Application Tags Database



	Name	Type	Description
1	Name	String	
2	PhysicalLocation	String	
3	Capacity	Real	
4	Level	Real	
5	Temperature	Real	
6	Pressure	Real	
*		Integer	
*		Integer	

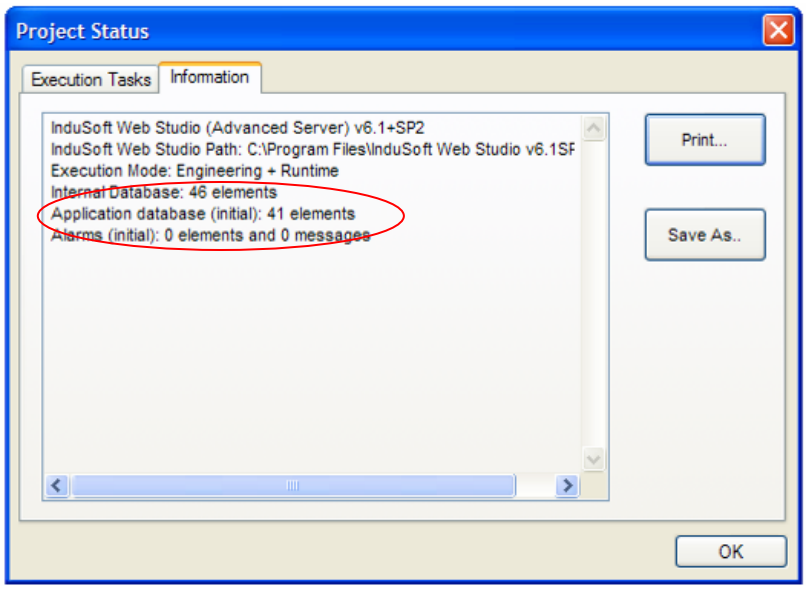
cTank Class definition

The Class definition for Class cTank defines a Class that will use 2 String Tags and 4 Real Tags. Since the Class definition does not declare the Tag, no tags are actually used. But for each Class tag declared, 6 tags will be required.

In the Applications Tags database, we have defined the following tags:

Tag	Type	Size	Tags/Item	Tags Used
v1	Integer	0	1	1
v2	Integer	0	1	1
v3	Integer	0	1	1
tank	cTank (Class)	5	6	36
MyPtr	String	1	1	1
@My2Ptr	Integer	1	1	1

Thus, we have used a total of 41 tag.. This calculation can be verified by clicking (in the Main Toolbar) on **Project → Status→ Information** and viewing the **Application database (initial)** value.



Notes:

- Class definitions do not use any tags. Tags are used only when declared in the Application Tags database.
- A Pointer tag only counts a one tag, regardless of whether it is used to point to a simple tag data type (Boolean, Integer, Real or String) or if it is used to point to an Array tag or Class tag.
- Internal Tags, Shared Database Tags and VBScript variables are not IWS tags, and do not count against the license tag limit.
- All Arrays are zero-based. The number of tags used for an Array tag of Boolean, Integer, Real, or String tags is (**Size + 1**). For example, if Size = 5, there are really 6 elements (tags) used.
- To calculate the number of tags for an Array of Class tags, use the following formula:
  - (# tags defined in a Class) * (Size +1)

Defining Tag Parameters

As previously discussed, IWS tags are not simply variables. Every IWS tag has a number of Properties associated with it. Certain tag Properties are only accessible in the development environment. Some tag Properties are available in the development environment and during runtime, while others are only accessible during runtime. Tags that are accessible during runtime are called Tag Fields. Some runtime Tag Fields are Read-Only while others are Read-Write.

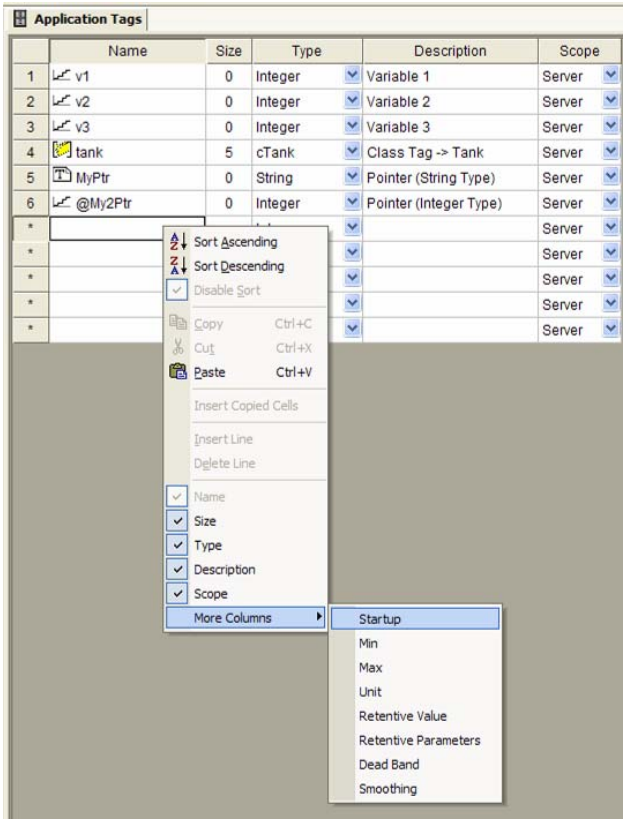
Some of the Tag Parameters are very useful, depending on your application. Some Tag Properties (Fields) worth noting are:

Max & Min values	These are minimum and maximum values for a tag. If exceeded during runtime, an warning occurs. These values can be changed during runtime
Quality	Indicates whether an improper evaluation occurred (e.g. divide by 0), or if there was a bad read from a communication driver (e.g. read or write error to a PLC, read or write error from the OPC Server to a PLC or other device)
TimeStamp	Tells you when a tag last changed value
Alarm Status & Alarm Limits	Allows you to set Alarm Limits at runtime, check to see if an alarm is active (caused by the tag), and to clear any alarm caused by the tag.

Accessing tag Parameters during development

Actually, we have already seen some of the tag Properties in the Application Tags Database (e.g. Size and Description). IWS provides two (2) methods of accessing Tag Properties in the development environment. The first is in the Application Tags Database, and the second is by selecting an Application tag in the Application Tags database and selecting the Tag Properties icon in the Toolbar.

Open the Application Tags folder in the Database Tab of the Workspace. Select Datasheet View. Note that the default tag Properties that are displayed are **Name, Size, Type, Description, and Scope.** However, if you position your mouse cursor in any row of the Datasheet View and click the right mouse button, you will get a list of options. At the bottom on this list of options is the option to view additional (Property) columns.



In the Datasheet view, these additional columns are:

Startup	An initial startup value for the tag if the Retentive Value option is unchecked.
Min	A minimum value that is allowed for the Tag. If the tag evaluates to a value below the minimum, a warning to the LogWin Window will occur. Values below the Minimum are not allowed to be entered by any Object (e.g. Text I/O).
Max	A minimum value that is allowed for the Tag. If the tag evaluates to a value below the minimum, a warning to the LogWin Window will occur. Values below the Minimum are not allowed to be entered by any Object (e.g. Text I/O).

Unit A text field used to describe the engineering units for the tag.

Retentative Value A checkbox that, if checked, tells the IWS database manager to save all changes to a Tag's value during runtime operation.

Note: checking this option can cause frequent access to your hard drive or flash device, based on your application

Retentative Parameters A checkbox that, if checked, tells the IWS database manager to save all changes to the Tag's Field values during runtime.

Note: checking this option can cause frequent access to your hard drive or flash device, based on your application

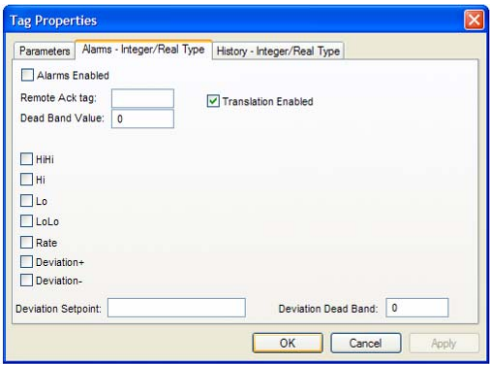
Dead Band The Dead Band field is used to filter alarms. It is a value of variance around an Alarm value that the tag can vary before triggering an alarm. For example, if an Alarm Hi Limit is set to 90 and the Dead Band value set to 5, then an Alarm will be generated when the tag value is ≥ 95, and the Alarm will return to a Normal state (normalized) when the tag value is ≤ 90.

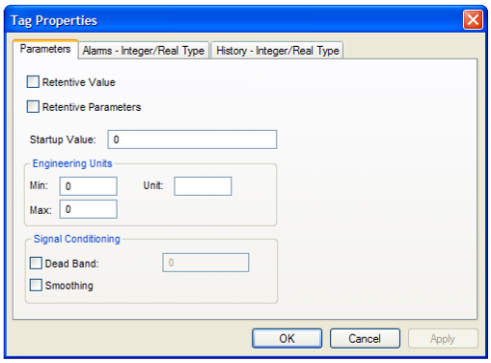
Smoothing A checkbox that, when checked, enables the IWS database manager to reduce the rate of change for a tag by averaging the current value with the previous value. The averaged value is then stored in the tag value.

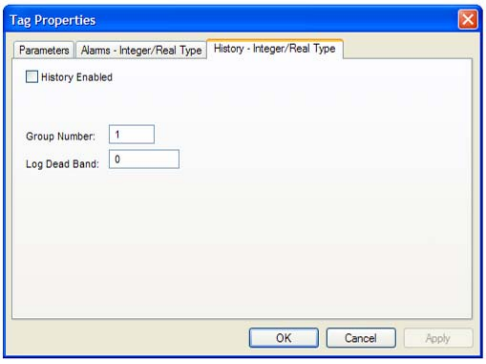
Along with the tag Properties that are accessible in the Datasheet View, additional tag Properties can be accessed by the Tag Properties Tool. This tool can be used by opening the Datasheet View, selecting (left mouse click) the tag of interest and then activating the Tags Property tool (left mouse click on the icon) in the Toolbar located below the Main Menu Bar.

Tag Property Icon

The Tag Property Tool has three (3) dialog boxes, selectable by tabs on the top of the dialog box. The first tab allows you to specify the same Tag Properties that are accessible from the Datasheet View (with all columns enabled). The second tab allows you to set tag parameters that are also set by an Alarm Worksheet (Task Tab of the IWS workspace), although an Alarm Worksheet provides additional configuration options. The last tab allows setting of some of the parameters that are specified in a Trend Worksheet (Tasks Tab in the IWS workspace). It should be noted that if any of these Properties are changed in the Alarm or Trend Worksheet, those values will be written into the fields shown here.







Accessing Tag Fields during runtime

Accessing Tag Fields (Parameters) during runtime is straightforward, and is done through the following syntax:

Tag->FieldName

where **Tag** is the name of the tag (for all tag types)
-> is the minus sign followed by a right arrow
FieldName is the name of the Tag Field (see table below).

Here are a couple examples:

v1->Max will return the maximum value allowed for the tag. (normally max & min are set to 0 to allow all value ranges)

v2->Quality returns the tag quality value (192 = Good, 0 = Bad)

If the tag is an array, depending on the Field used, you will want to either use just the array name or use the array name with the index. For example, if we have an array called Switches with a Size of 5:

Switches->Size will return a value of 5

Switches[Count]->Index will return the value of the tag Count used as the index to the array Switches

If the tag is an Array of Class tags, you can refer to the Array name, Array name with Index, or the Class Members. Using the previously defined Array of Class tags (tanks), for example:

tank->Size	returns a value of 5
tank[1]->Index	returns a value of 1
tank[2].Level->MemberName	returns the string “Level”
tank[3].Level->TimeStamp	returns a time & date when the level last changed

Notes:

- Tag Properties provide access to critical tag information beyond the tag’s value
- Depending on the Tag Property, it can be accessed:
  - in the development environment only
  - in both the development environment and the runtime environment
  - in the runtime environment only

IWS Tag Properties (Fields) Accessible During Runtime

Field Name	Description of Value Associated w/ Each Field	Tag Type Associated with Field				R=Read Only RW=Read+Write
		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 used must be: **<Class>.<Member>->MemberName** Example: tank[1].Level->MemberName returns a string “Level”	✓	✓	✓	✓	R
Index	The index number of an element in an Array tag. An Array is any tag with a Size greater than 0. **Note:** the syntax used must be: **<Tag>.[<Index>]->Index** Example: tank[1]-> Index returns a “1”	✓	✓	✓	✓	R
Description	Description of tag as configured in the Tags database.	✓	✓	✓	✓	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
Quality	Tag quality (192=GOOD; 0=BAD). This Field updates every time the tag receives the result of an expression or when tag receives a value from a communication task (Driver or OPC). Invalid expressions (such as: division by zero) or reading communication errors associated with tag will set quality to BAD.	✓	✓	✓	✓	R
Size	Array Size. If the tag is not an array tag, it returns the value 0	✓	✓	✓	✓	R
TimeStamp	Time & date when tag changes value	✓	✓	✓	✓	R
Unit	Brief description (up to 9 characters) of the engineering unit for the tag value (for example: "Kg")	✓	✓	✓	✓	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).	×	✓	×	×	RW

Field Name	Description of Value Associated w/ Each Field	Tag Type Associated with Field				R=Read Only RW=Read+Write
		Boolean	Integer	Real	String	
AlrStatus	Integer value with the status of all currently active alarms associated with tag. Each bit of this integer value indicates a specific status: Bit 0 (LSB): HiHi Alarm active Bit 1: Hi Alarm active Bit 2: Lo Alarm active Bit 3: LoLo Alarm active Bit 4: Rate Alarm active Bit 5: Deviation+ Alarm active Bit 6: Deviation- Alarm active For example: If **Tag->AlrStatus** returns value 2, "Hi" alarm is active. If **Tag->AlrStatus** returns value 3, "HiHi" and "Hi" alarms are active simultaneously. If **Tag->AlrStatus** returns value 0, there are no active alarms associated with this tag. For Boolean tags, only 1 (bit 1), 4 (bit 2) or 16 (bit 4) values are returned.	✓	✓	✓	✗	R
Ack	This field can have two values: **0**: No alarms associated with this tag, requiring acknowledgment **1**: At least one alarm associated with this tag, requiring acknowledgment **Note:** This works as a global acknowledge for the tag and returns to 0 only when the all alarms for the tag have been acknowledged	✓	✓	✓	✗	RW
AlrDisable	This field can have two values: **0**: Enables alarm associated with tag, meaning when an alarm condition occurs, the alarm becomes active. **1**: Disables alarm associated with tag, meaning that even if alarm condition occurs, alarm will not become active.	✓	✓	✓	✗	RW
HiHi	If **0**, HiHi alarm is inactive. If **1**, HiHi alarm is active.	✗	✓	✓	✗	R
Hi	If **0**, Hi alarm is inactive. If **1**, Hi alarm is active.	✓	✓	✓	✗	R
Lo	If **0**, Lo alarm is inactive. If **1**, the Lo alarm is active.	✓	✓	✓	✗	R
LoLo	If **0**, LoLo alarm is inactive. If **1**, the LoLo alarm is active.	✗	✓	✓	✗	R

Field Name	Description of Value Associated w/ Each Field	Tag Type Associated with Field				R=Read Only RW=Read+Write
		Boolean	Integer	Real	String	
Rate	If **0**, Rate alarm is inactive. If **1**, the Rate alarm is active.	✓	✓	✓	×	R
DevP	If **0**, Dev+ alarm is inactive. If **1**, the Dev+ alarm is active.	×	✓	✓	×	R
DevM	If **0**, Dev- alarm is inactive. If **1**, Dev- 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
LoLoLimit	Limit value for LoLo alarm.	×	✓	✓	×	RW
RateLimit	Limit value for Rate alarm.	×	✓	✓	×	RW
DevSetpoint	Set point value for Deviation alarms.	×	✓	✓	×	RW
DevPLimit	Limit value for Deviation+ alarm.	×	✓	✓	×	RW
DevMLimit	Limit value for Deviation- alarm.	×	✓	✓	×	RW
Blocked	Tag is blocked from use	✓	✓	✓	✓	RW

Notes:

- You cannot use tag fields (such as Bit fields) to configure Alarm or Trend worksheets.
- If the application tries writing a value outside the range specified in the **Min** and **Max** fields, the Tags database will not accept the new value and a warning message is sent to the LogWin window. If you configure both **Min** and **Max** properties with the value 0 (zero), any value applied to the tag type can be written to the tag.

Working with Tags

IWS provides a number of Tools to work with tags. These Tools are:

Object Finder The Object Finder Tool is primarily used to specify a tag to be used with an Object. It is a quick, easy alternative to opening the Datasheet View of the Application Tags database. Using the Object Finder eliminates spelling errors when specifying a tag.

Object Finder Icon

Tag Properties The Tag Properties Tool provides access to a Tag's Properties that are available in the runtime environment. You must first select the tag (e.g. from the Datasheet View in the Application Tags database and then select the Tags Property Tool.

Tag Properties Icon

Cross Reference The Cross Reference Tool will search all the application's screens and Worksheets (e.g. Alarm, Trend, Math, Script, Communications, OPC, etc.) and output to the LogWin Output Window wherever the Tag is used in the application. To use this tool, select the tag of interest (e.g. from the Datasheet View of the Application Tags database) and then select the Cross Reference icon.

Cross Reference Icon

Global Replace Tags The Global Replace Tags Tool will replace an existing tag name with a new tag name, everywhere in the application that the tag is used. The new tag name must be a unique tag name. To use this tag, first click on an existing tag (e.g. a tag in the Datasheet View of the Application Tags database) and then click on the Global Replace Tags icon. Enter the current tag name in the **From** field and enter the new tag name in the **To** field.

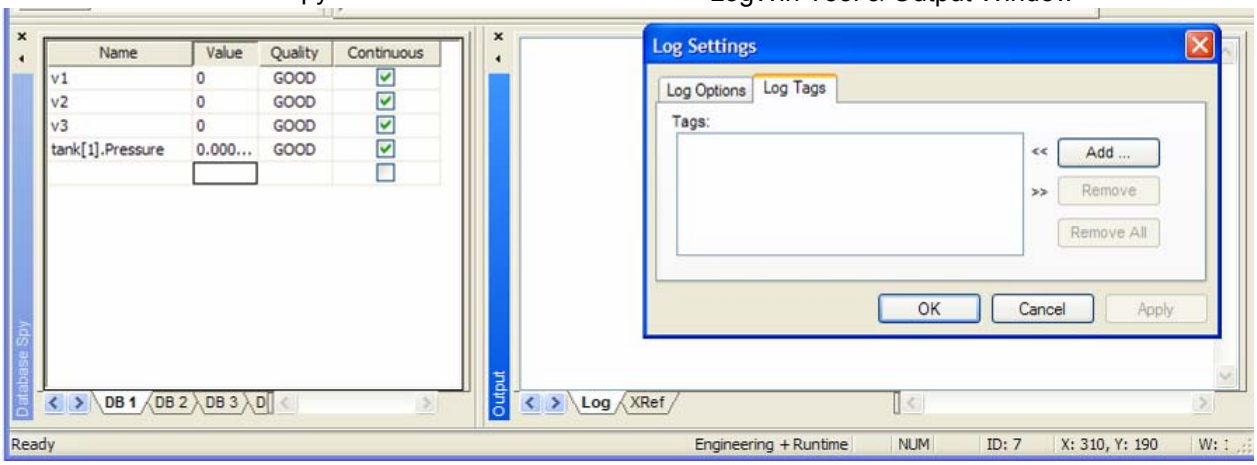
Global Replace Tags Icon

LogWin LogWin is used for a number of purposes. One of these is to log (to the Log Win Output Window) any changes to tag values that occur. To use this tool, place your mouse cursor in the LogWin Output Window, click the right mouse button and select **Settings.** Select the **Log Tags** tab, and click on **Add** to define the tag and add it to LogWin.

Note: Only tag values (i.e. not the Property values) are logged by this Tool.

Database Spy The DataBase Spy monitors tag values and can force tag values. Position the mouse cursor in an empty **Name** field and either click the left mouse button and type the tag name, or double click the left mouse to use the Object finder.

Database Spy LogWin Tool & Output Window



Adding and Editing Tags

IWS tags can be added in the following ways:

- In the Application Tags Datasheet View, position the mouse cursor on an empty line (with a * in the Index Field), click the left mouse button and type a new tag name
- the Application Tags Datasheet View, position the cursor on any line and click the right mouse button and select Insert Line.
  Note: To insert a new line in the Application Tags database, the runtime application must be stopped and select the **Disable Sort** option if it is enabled.
- Use a tag in an IWS Object or in a VBScript code segment. If the tag does not exist (and is a unique tag name), it will be created
- You can copy & paste between the Datasheet View and other Worksheets (e.g. Communications). But since each tag is unique, any inserted tag(s) in the Application Tags DataSheet View will automatically get a unique suffix. E.g. If you copy tags v1 & v2 and then paste in the Datasheet View, the pasted tags will be v_2 & v_3. If you paste again, the new tags will be v_4 and v_5. You can rename these tags later with the Global Replace Tool.
  Note: If the original tag(s) do not have a numeric suffix, the inserted suffix may be different after the paste function is executed.
- To delete tags, open the Tag Data Sheet view. Highlight the tag or tags you want to delete and right click. Select Delete Line.
  Note: To delete an existing line in the Application Tags database, the runtime application must be stopped and select the **Disable Sort** option if it is enabled.

Notes:

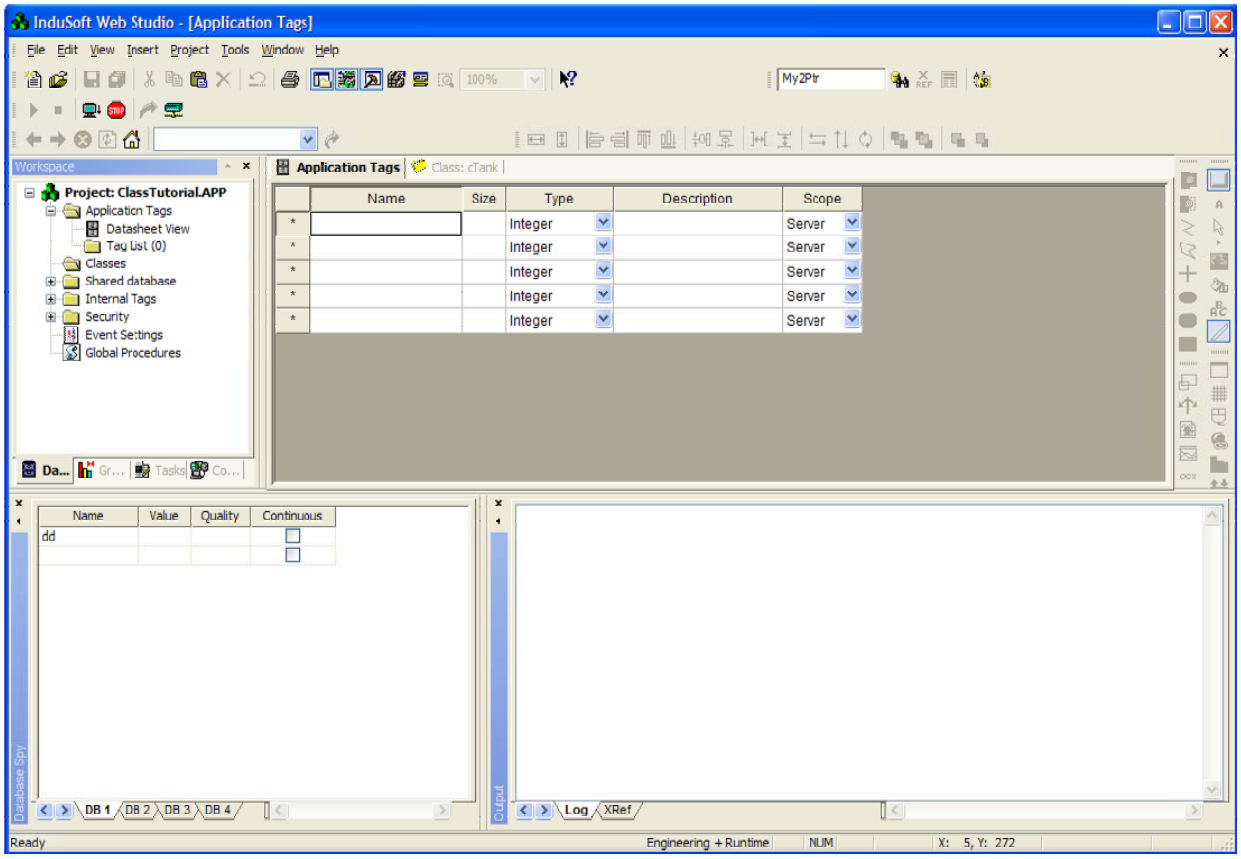
- You can Copy (Ctrl C) and Paste (Ctrl V) between the Application Tags database and other Worksheets (and even Microsoft Excel). This is useful when configuring Worksheets.
- After you click on a tag in the Datasheet View, hold the left mouse button to highlight multiple tags (lines). Using this method, you can select and delete multiple lines at the same time.
- If you cannot insert or delete tags from the Datasheet View of the Applications Tags database, be sure:
  1) the IWS (or CEView) runtime is stopped
  2) the **Disable Sort** option is grayed out (not enabled)

IWS Tags Database Exercise

For this exercise, we will begin to define some tags for the application we will be building in subsequent chapters. We will add more tags later, as required, for our application.

Let's begin by doing the following:

- ☐ Open your Project (ClassTutorial.app) if it not already opened
- ☐ Open the Datasheet View in the Application Tags database folder (select the Database Tab in the Workspace Window)



Datasheet View in Application Tags

- ☐ Position your mouse cursor in the **Name** field of the first line and click the left mouse button. Tab through each of the columns and enter the following information for this tag:

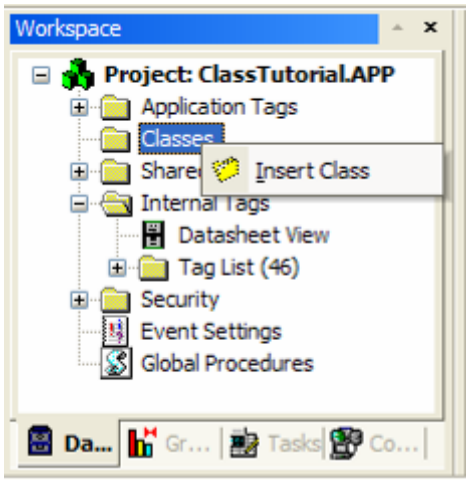
  Name: v1
  Size: 0
  Type: Integer
  Description: Variable 1
  Scope: Server

☐ Tab down to line 2 and enter the following information:

Name: v2
Size: 0
Type: Integer
Description: Variable 2
Scope: Server

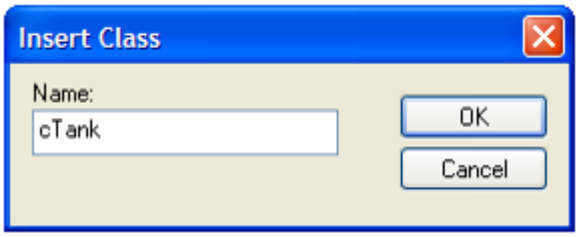
☐ Tab down to line 3 and enter the following information:

Name: v3
Size: 0
Type: Integer
Description: Variable 3
Scope: Server

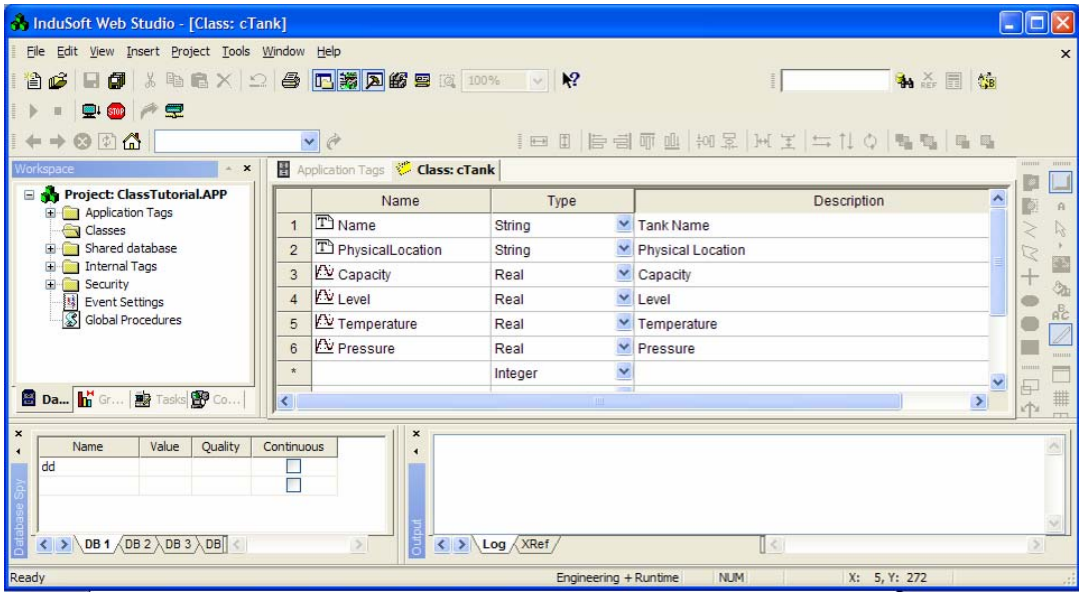


☐ We will now define a Class named cTank. To do this, position you mouse cursor over the **Classes** folder in the Database Tab of the Workspace. Click the right mouse button. Now, position your cursor over the **Insert Class** option and click your left mouse button.

☐ Enter cTank into the **Name** field of the Insert Class popup window. Click on **OK**.



☐ Enter Class Member information as shown below:



☐ Position you mouse cursor on the icon **Class:cTank** above the worksheet, click the right mouse button and select **Close.**

☐ Open the Application Tags database Datasheet View, and enter the following information of line 4

Name: tank
Size: 5
Type: cTank
Description: Tank Array
Scope: Server

☐ Tab down to the next line and enter the following information:

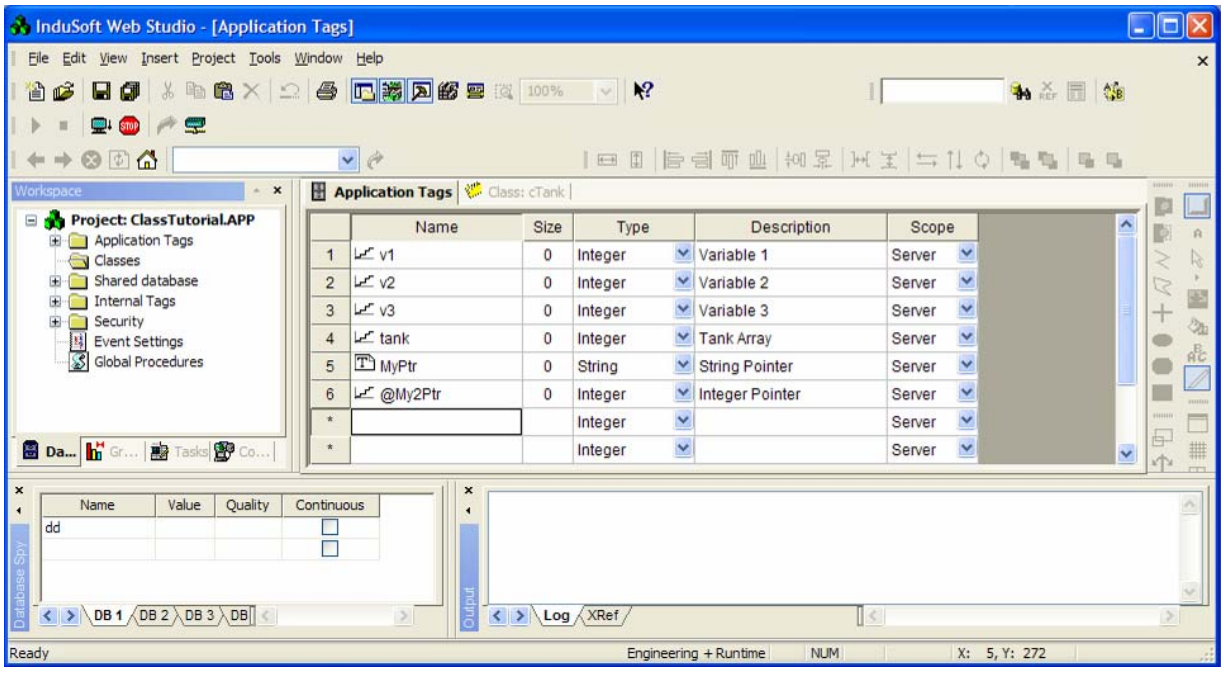
Name: MyPtr
Size: 0
Type: String
Description: String Pointer
Scope: Server

☐ Tab down to the next line and enter the following information:

Name: @My2Ptr
Size: 0
Type: Integer
Description: Integer Pointer
Scope: Server

☐ Tab down to the next line

☐ When you are finished, the Application Tags database should look as follows:



Notes

Chapter 8. Application Screens and Static Objects

Displaying machine or process data and accepting Operator input is a basic requirement of any HMI/SCADA software package. IWS provides a rich set of native graphic Objects, support for ActiveX & .NET Controls, as well as a MDI (Multiple Document Interface) graphical screen environment.

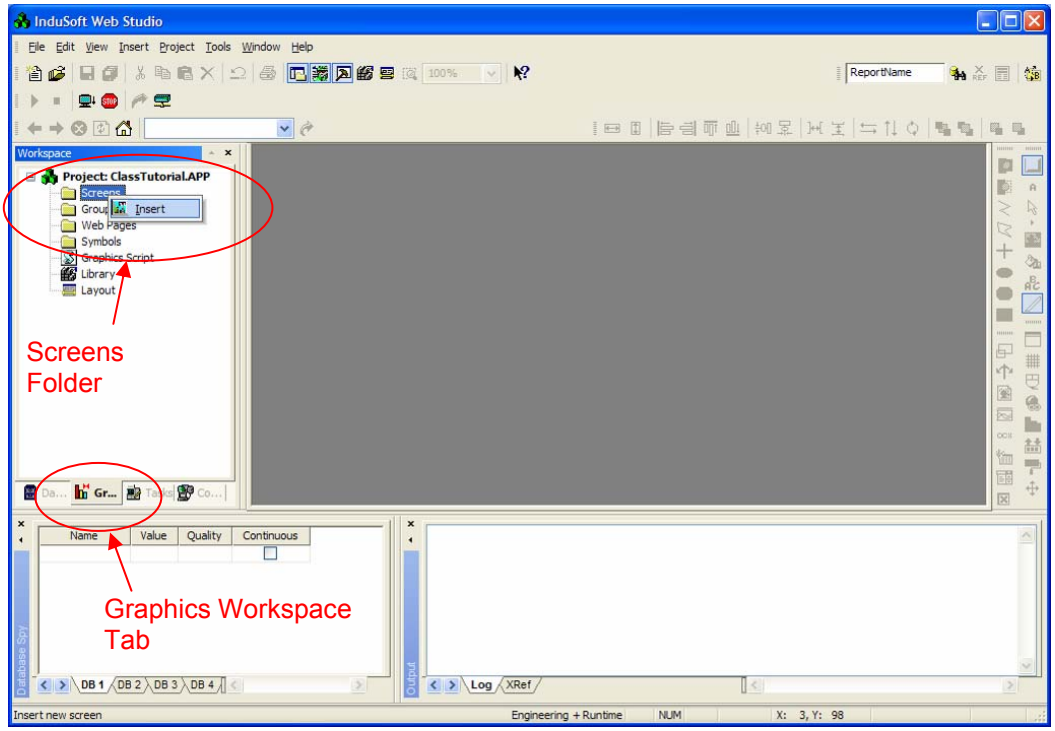
Standard Screens and MDI Screen Groups

A Standard screen is a screen which generally has the same resolution as your LCD screen and is used without other Screens in the same active display. For example, many 15" PCs and Monitors have a resolution of 1024 x 768 pixels. This is then the same resolution that we would set for our Standard (application) Screen.

A MDI Screen is a Screen that is used with one or more other Screens to form an active display. Collectively, these are called a Screen Group. Creating an MDI Screen is really no different than creating a Standard Screen, except it an MDI Screen usually has a Screen resolution that is less than the LCD screen resolution.

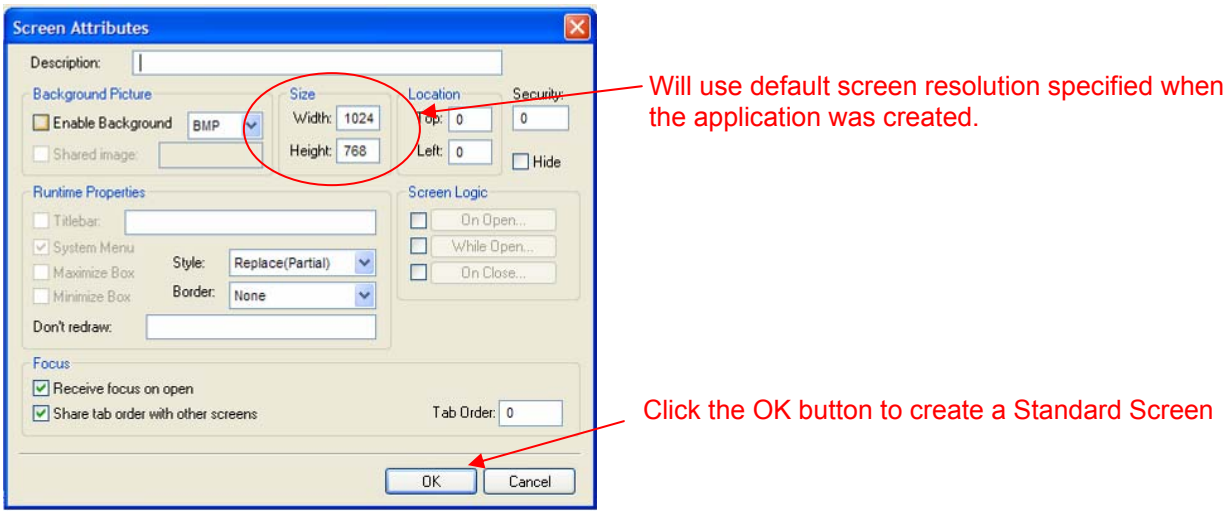
Creating a Standard Screen

A Standard Screen is the simplest and most basic screen type. To create an Standard Application screen, we need to first open an existing Project or create a new Project. Select the **Graphics** Workspace Tab. (If the **Workspace** view is not displayed, select **View** from the Menu Bar, then select **Toolbars**, then select **Workspace**. Alternatively, you can press **Alt + 0**). Next, right click on the **Screens** folder to select the option to insert a new Screen. Left click on the **Insert** popup to insert the new Screen.



Inserting a new Screen

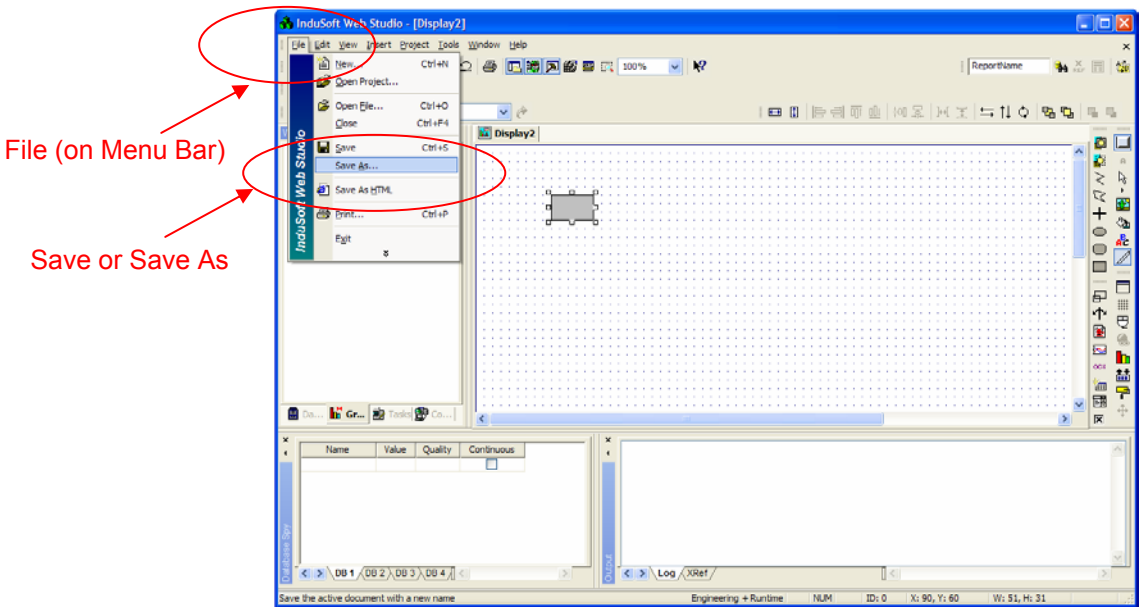
After inserting a new screen, a **Screen Attributes** dialog box will appear. Simply click the **OK** button in the lower right corner, and you will be creating a standard screen. Normally, you will place various IWS Objects and/or ActiveX & .NET Controls on the Screen. Later, when you go to save the screen, IWS will prompt you for the Screen Name and will save it with a .scr file extension, signifying it as a Screen file. Note that screen default resolution will be the standard application default resolution as specified when the application was created.



Screen Attributes Dialog Box

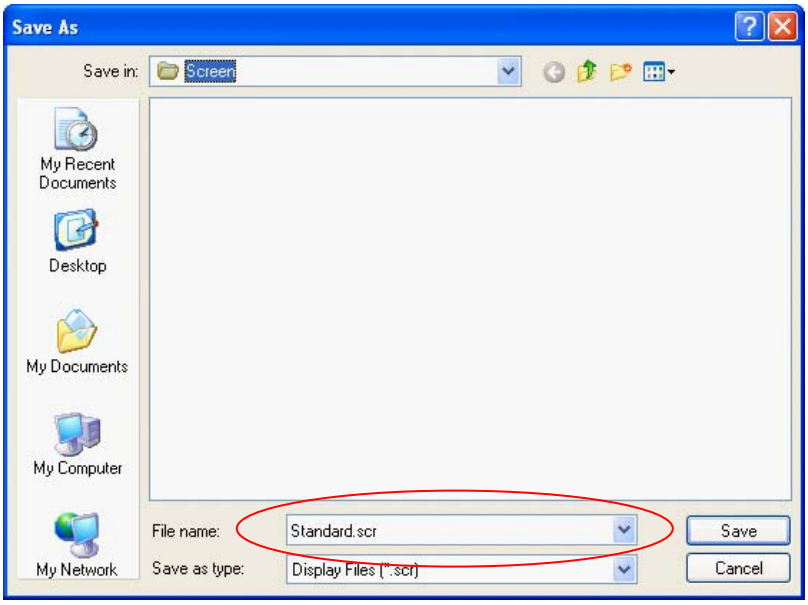
Once you have placed some IWS Objects (or ActiveX/.NET Controls) on the Screen, you need to save the Screen. This can be done by using one of three methods:

- In the Menu Bar, click on **File**, then select **Save As** if this is a new Screen. Otherwise select **Save** if the Screen already exists and you are just saving the latest change (or you can click on the **Save** icon or right click on the Screen name (above the Screen) and select **Save**) . If the Screen in a new Screen, you will be prompted to enter a file name for the Screen. The Screen will be saved as a Display File with a .scr extension. All screen files are automatically saved in the application folder in the \Screen



Saving the Screen

Below is an example of the dialog box to define the file name for the Screen. This dialog box will appear when you select **Save** or **Save As** when the Screen has not been previously been saved, or if you select **Save As** with an existing Screen.



Saving a new Screen or renaming an existing Screen

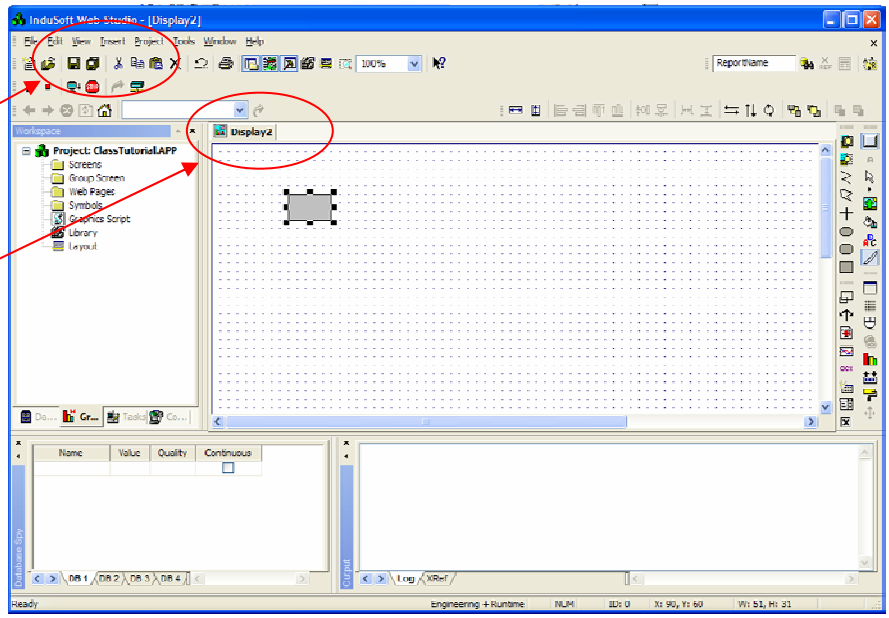
Note:

- All screen files are automatically saved in the \Screen subfolder in the application folder. Screen files need to be stored in this subfolder. Do not locate them into another subfolder.

- Click on Save or Save All buttons in the top Toolbars section. Note that these buttons have the characteristics as selecting **File** then **Save** or **Save As** from the Menu Bar.
- Right click on the display name that is just below the top toolbar. This will give you the option to **Save** or **Close** the Screen. If the Screen is new, you will be prompted for a filename.

[File] Save and Save All buttons located in the top Toolbar

Screen Name located just below top Toobar



Tips & Tricks for using Standard Screens:

1. To save a blank Screen, you must use **SAVE AS in the IWS Main Tool Bar (e.g. FILE → SAVE AS).** You cannot right click on the Screen name tab (above the Screen), use the **SAVE** command in the IWS Main Tool Bar (e.g. **FILE → SAVE),** or use the Save icon unless you first placed one or more Objects (IWS Objects, ActiveX or .NET Controls) on the Screen.
2. To rename a Screen, first close your IWS application and the development environment. Open the \Screen subfolder in your application directory. Each Screen will have three (3) files associated with it. Rename all three of these Screen files to the same new name. The Screen name must be unique from the other Screen name(s) and must follow standard Windows file naming conventions.
3. There are two methods to copy a Screen into your current application.
   a. In the Menu Bar, Select **File**, then **Open File**. Be sure the File Type is specified as *.scr. A list of screens in your current \Screen subfolder will be displayed. The drop-down box and the **Up a Level** folder button can be used to navigate to other directories that contain Screen files your are interested in copying. Select the Screen file you are interested in importing into you application and press the **Open** button in the lower right corner of the dialog box.

      NOTE: A Screen imported by this method will not be saved unless a change is made to the imported screen. If you do not want to make any changes to the Screen, then simply move an Object on the Screen and then return it to its original position.

      NOTE: The Open File dialog box can be used to import several different types of IWS files. Right now, we are only concerned with Screen files, but several other type of IWS files can be imported into the current application using the Open File command.

   b. First close your IWS application and development environment. Next, open the \Screen subfolder in your application directory (or other \Screen subfolder for the application screen you wish to copy). Each Screen will have three (3) files associated with it. Copy the three (3) files and paste them into your \Screen subfolder of your current directory (you will already be in your application's \Screen subfolder unless you have browsed to another application's \Screen subdirectory). If the name of the Screen file(s) is not unique from other Screen file names in your current \Screen subdirectory, you must rename these three (3) Screens to a new name (must be the same name for all three files). The new Screen name must follow standard Windows file naming conventions. When you restart the IWS development environment, the new Screen(s) will appear.
4. If you need to change the resolution for all your Screens (e.g. you are moving your application to a new hardware platform with a different display resolution), this is supported by IWS. You can resize your Screens at any time. Simply close all your Screens, then select **Tools** from the Tool Bar. Then select **Convert Resolution**, and in the dialog box that pops up, specify the resolution you want to convert to and press **Convert.**

   NOTE: There are practical limits when converting from one Screen resolution to another. Generally, moving to a large screen resolution is not a problem although you may want to increase the size of your Objects to take advantage of the increased display resolution. But when decreasing display (screen) resolution, an IWS Object may be placed on top of another Object. Before changing resolution, it is a good practice to make a backup copy of your application in case the original resolution needs to be restored.
5. To rename a Screen, first close your IWS application and development environment, then open the \Screen subfolder in your application directory. Each Screen will have three (3) files associated with it. Rename all three Screen files to the same new name. The Screen name must be unique from the other Screen name(s) and must follow standard Windows file naming conventions.

Tips & Tricks for using Standard Screens, continued:

7. After you save your first Screen, the **Screens** folder may appear that nothing is in the **Screens** folder. Just click on the **Screens** folder and your newly saved Screen will be shown.
8. You will want to specify the Screen that will be opened automatically when your application runtime is started. This Screen can be specified in one of two ways:
   a. Open the **Screens** folder. Place your mouse on the specific Screen you want to use as your startup Screen and right click. You will see an option appear that allows you to **Set as Startup**. Choose this option.
   b. In the Menu Bar, select **Project**, then **Settings**, then **Runtime Desktop.** There is a dialog box with a pull down list of all your Screens. Select the Screen you want as your startup Screen.

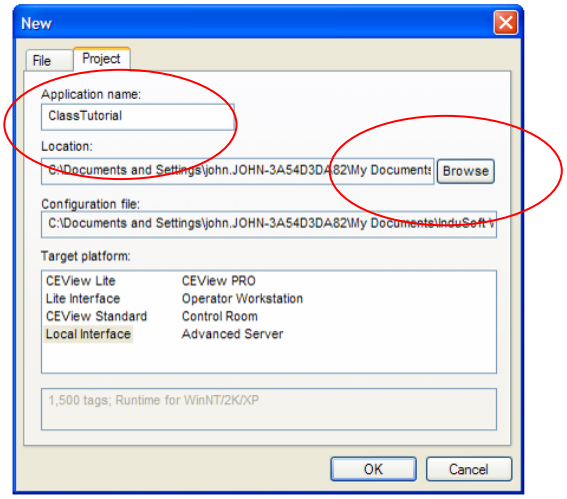
   NOTE: In order to have your startup Screen be the Screen that starts whenever your application runtime is started, all other Screens in the development environment must be closed.
9. IWS Objects with a Command Property must be used to navigate to other Screens

Exercise: Create a Standard Screen

In this exercise, we will create a Standard screen. This Screen can be used with an application or as a Screen template. Once a Screen template is created, it can be repeatedly saved as a new screen with a unique name. This allows you to create application Screens that all have the same attributes and basic objects.

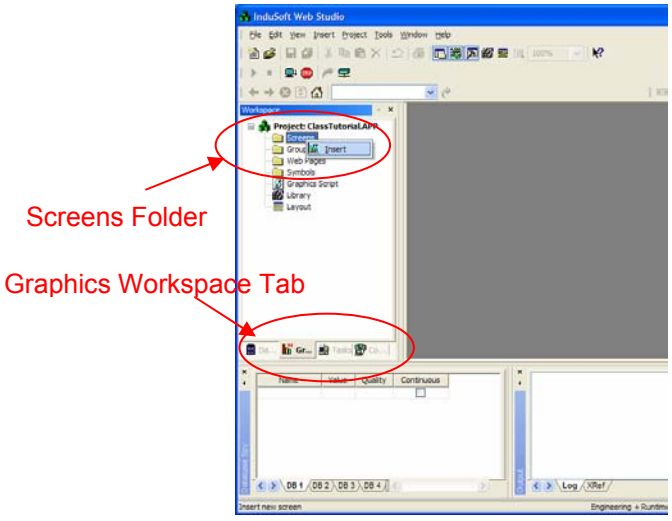
Use the following procedures to create your first project screen:

- ☐ Open the IWS development environment (if it is not already opened)
- ☐ Create a new Project Folder by the following steps:
  - In the IWS Main Menu Bar, click on **File** then **New.**
  - Click on the **Project** tab
  - If you need to specify a directory to locate the Application Folders, click on **Browse**, then navigate to the proper directory
  - Specify an **Application Name**. Descriptive Application Names are best. In this example, the Application Name will be **ClassTutorial.**
  - Depending on your license, you can specify one or more **Target Platforms.** For this example, specify **Local Interface** (this will let us use up to 1,500 IWS Application Tags).
  - Click **Ok**

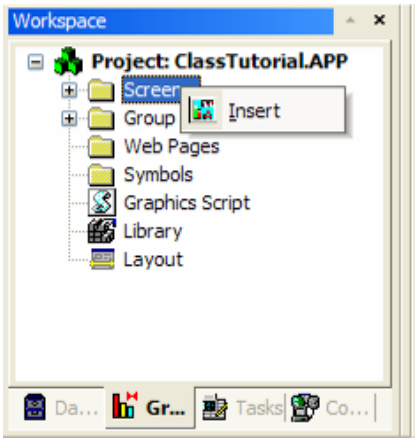


Open a new Project

- ☐ In the **Workspace**, select the **Graphics** tab.
- ☐ Right-click the **Screens** folder and select **Insert** or select **Insert →Screen** from the IWS Main Menu Bar.

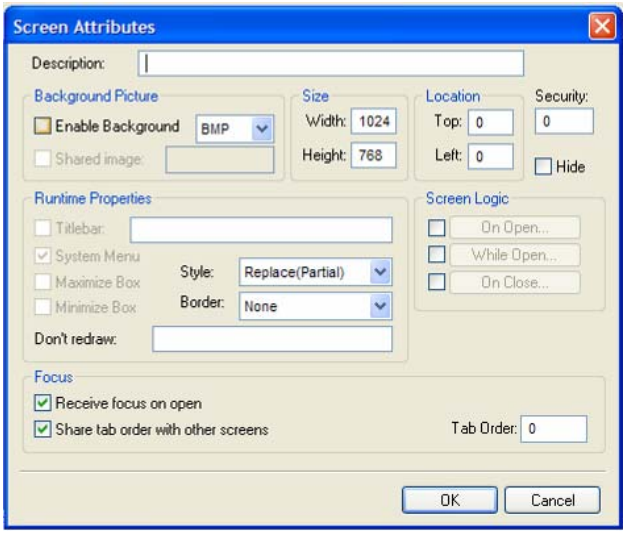


Select Graphics Workspace Tab



Right click to insert a new Screen

- When the *Screen Attributes* dialog displays, type **Standard Screen** into the **Description** field and then click **OK**. An empty Screen will display.

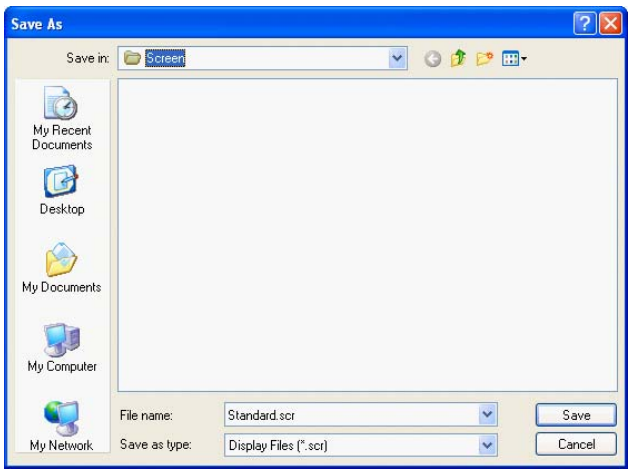


Screen Attributes Dialog

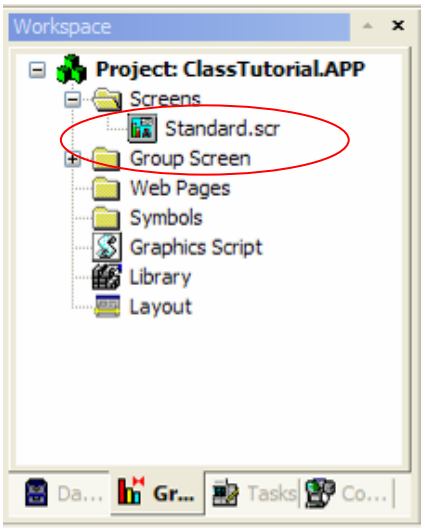
- Select **File →Save As** on the **Main Menu Bar** to save the new screen. When the **Save As** dialog displays, type **Standard.scr** (or **Standard**) into the **File name** field.

  Note: The .scr file extension is automatically used.

  Expand the **Screens** folder in the **Graphics** Workspace tab to see the saved screen.



Specify the File Name and Save

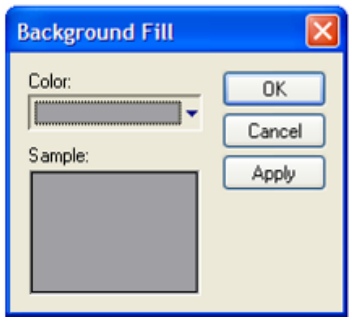


Checking for the New Screen

Change the background color of the Screen to gray. This is done by clicking the **Background Color** icon located on the **Toolbar** or right-click on the blank screen and choose **Background Color** from the pop-up menu.

Background Color icon

- When the **Background Fill** dialog box appears, click on the Color combo-box and select the light gray color. Click **Apply** to see the color applied to the background. Click **Ok** to close the **Background Fill** dialog box**.**

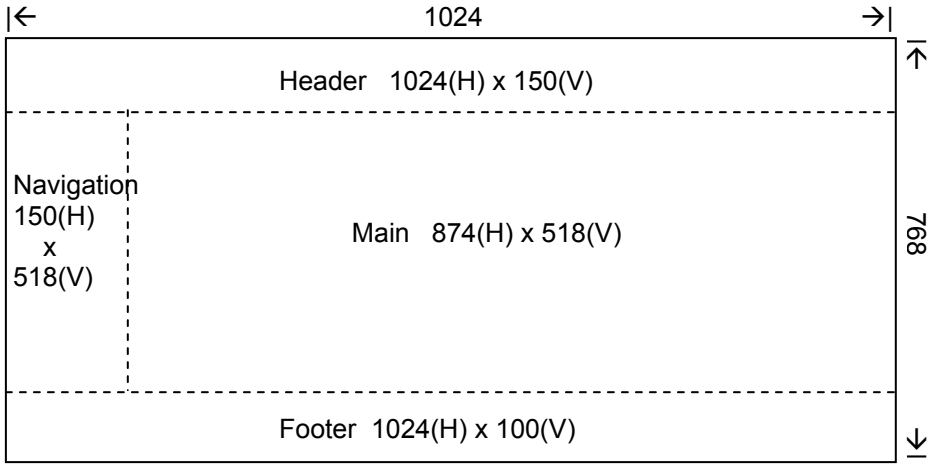


Background Fill Dialog Box

Creating MDI Screens

Before we cover the native graphic Objects and ActiveX/.NET Controls, we will first cover how to set up IWS to support an MDI environment. While using an MDI environment is not a necessity, it will make your application screens more user-friendly to navigate and easier to maintain since a common screen can be used throughout you application.

Let's start by considering the structure of a typical application screen. For example, we could create a 1024 x 768 screen that is divided into four components, called Child Windows:



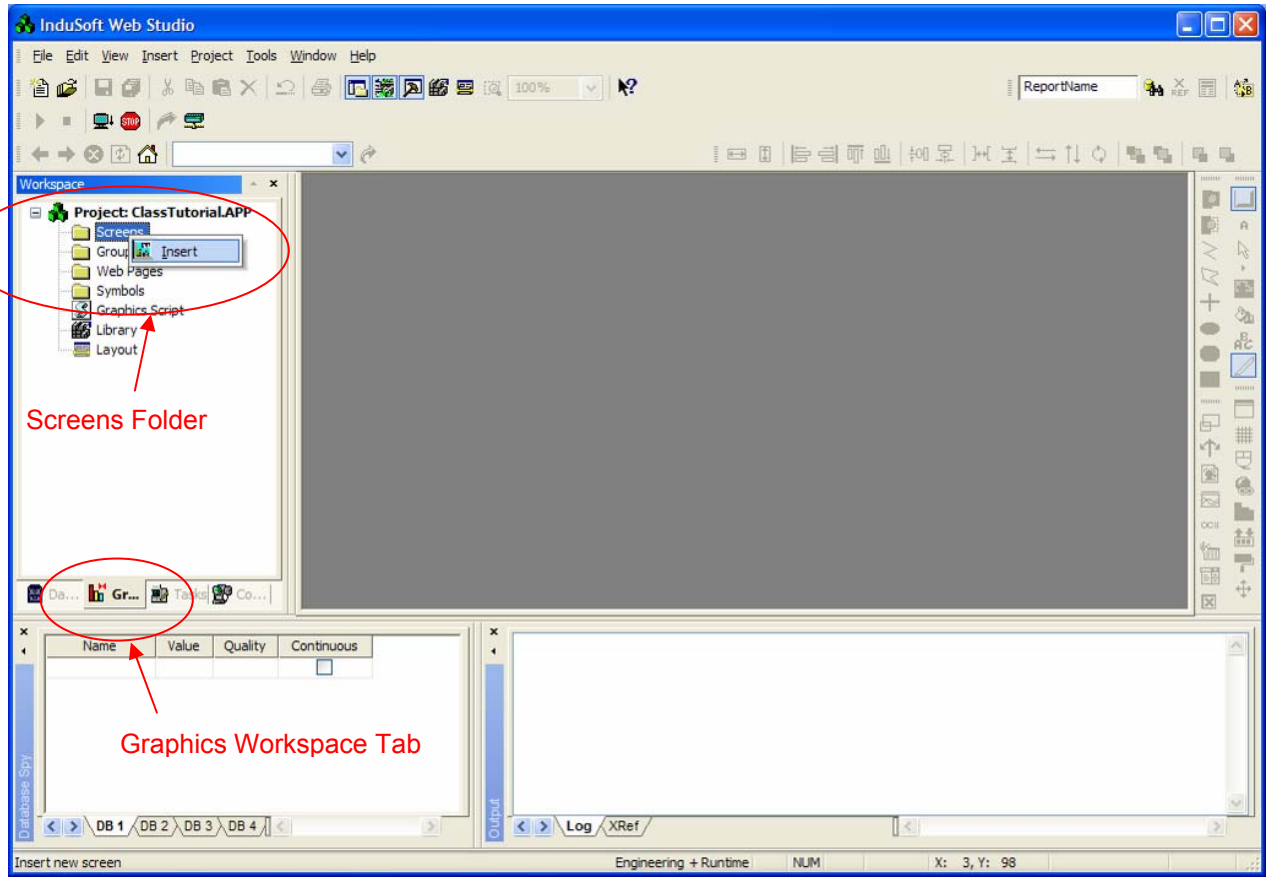
Each Child Window of this application screen has its own Horizontal (H) width and Vertical (V) height. In this example, the Child Windows consist of a:

- **Header**:
  A narrow region located at the top of the screen that can be used to provide standard information (Date, Time, Username, etc.)

- **Footer**:
  A narrow region located at the bottom of the screen that can be used to provide important application information (Alarm data)
- **Navigation**
  A region located on one side of the screen that can contain navigation buttons used to open other screens that replace the body section of this screen.
- **Main**
  The remaining area of a screen that can be used to provide specific information about the machine or process, detailed alarm screens, trends, etc.

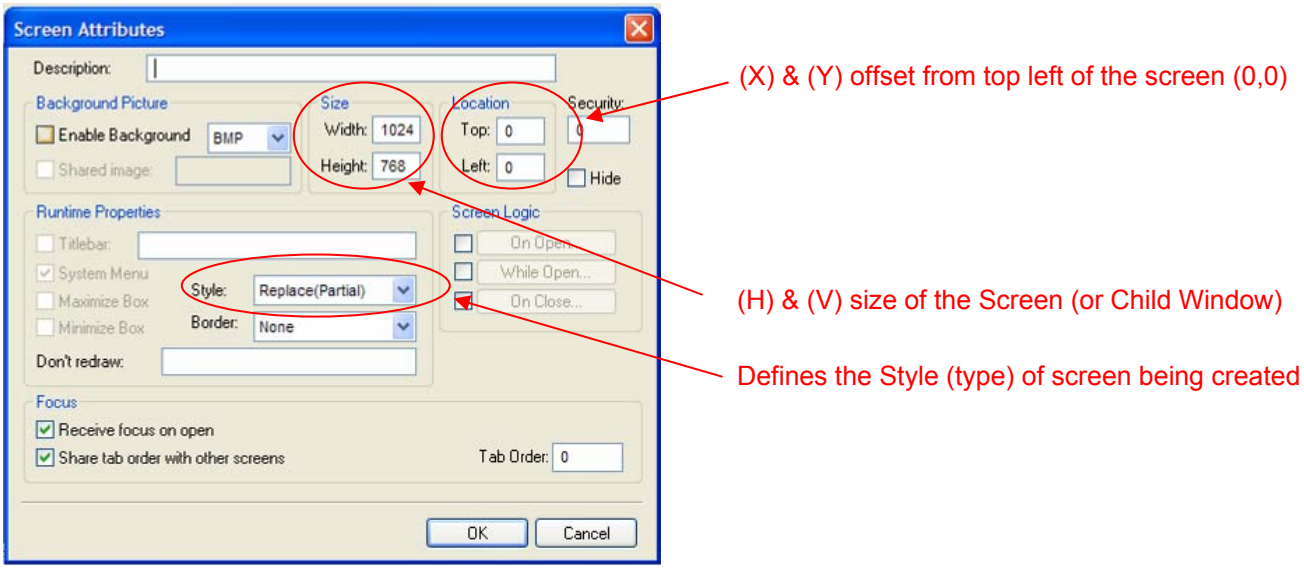
Your MDI Child Windows could use a similar arrangement or a completely different arrangement, it is up to you to determine the screen configuration that meets your application needs, providing a balance between sufficient screen area to display the information required and user-friendly operation.

To create an application screen that can be used as a stand alone screen or an MDI Child Window, we need to insert a new screen. After opening an existing Project or creating a new Project, select the **Graphics** Workspace Tab, and right click on the **Screens** folder to insert a new Screen. Left click on **Insert** to insert the new screen.



Inserting a new Screen

After inserting a new screen, a **Screen Attributes** dialog box will appear. To create an MDI Child Window, we will need to change some of the settings in the **Screen Attributes** dialog box. Next, just as with the Standard Screens, you place various IWS Objects and/or ActiveX & .NET Controls on each Child Window. Later, when you go to save the Child Window, IWS will prompt you for the Screen Name and will save it with a .scr file extension, signifying it as a Screen file. This is similar to Standard Screens.



Setting Screen Attributes in the Dialog Box

To create a MDI Screen as shown in the above example, you would create four (4) separate screens called MDI Child Windows.

Child Window Screen Name	Width	Height	Top	Left	Style
Header	1024	150	0	0	Replace (Partial)
Footer	1024	100	668	0	Replace (Partial)
Navigation	150	518	150	0	Replace (Partial)
Main	874	518	150	150	Replace (Partial)

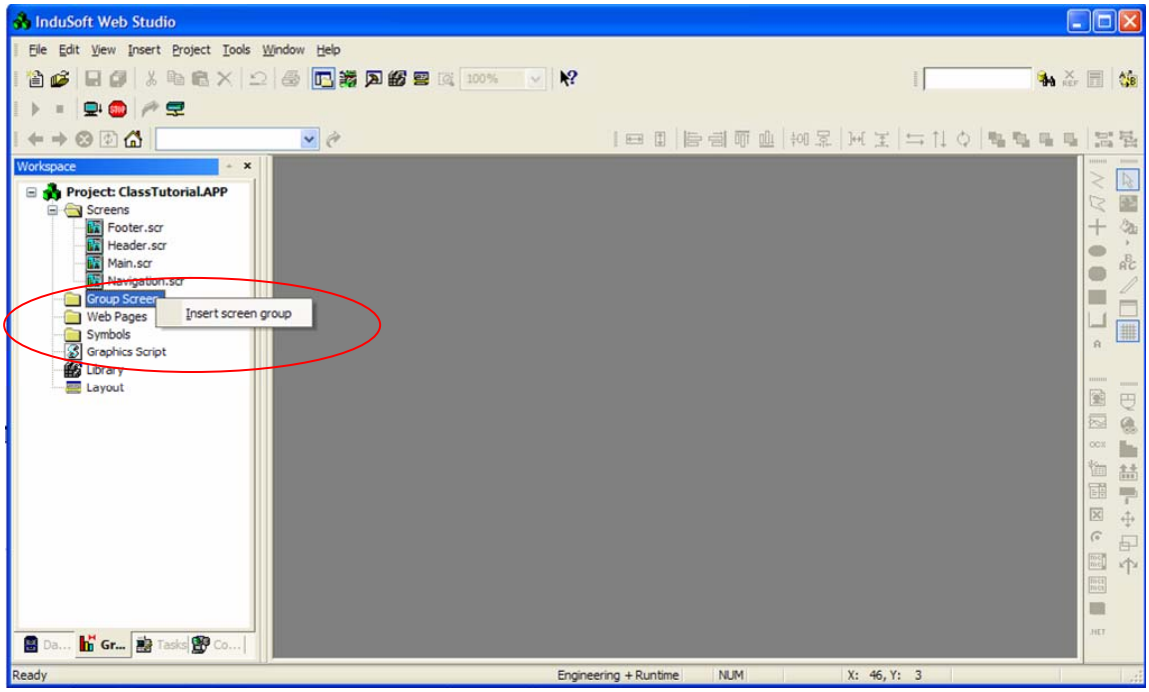
Important Note:

The numeric values for Width, Height, Top and Left are pixel value offsets relative to the upper left corner of the Screen, which is 0,0. The maximum values for a Child Window (Width & Height) plus and offset must not exceed the maximum size for the display as specified in your project settings.

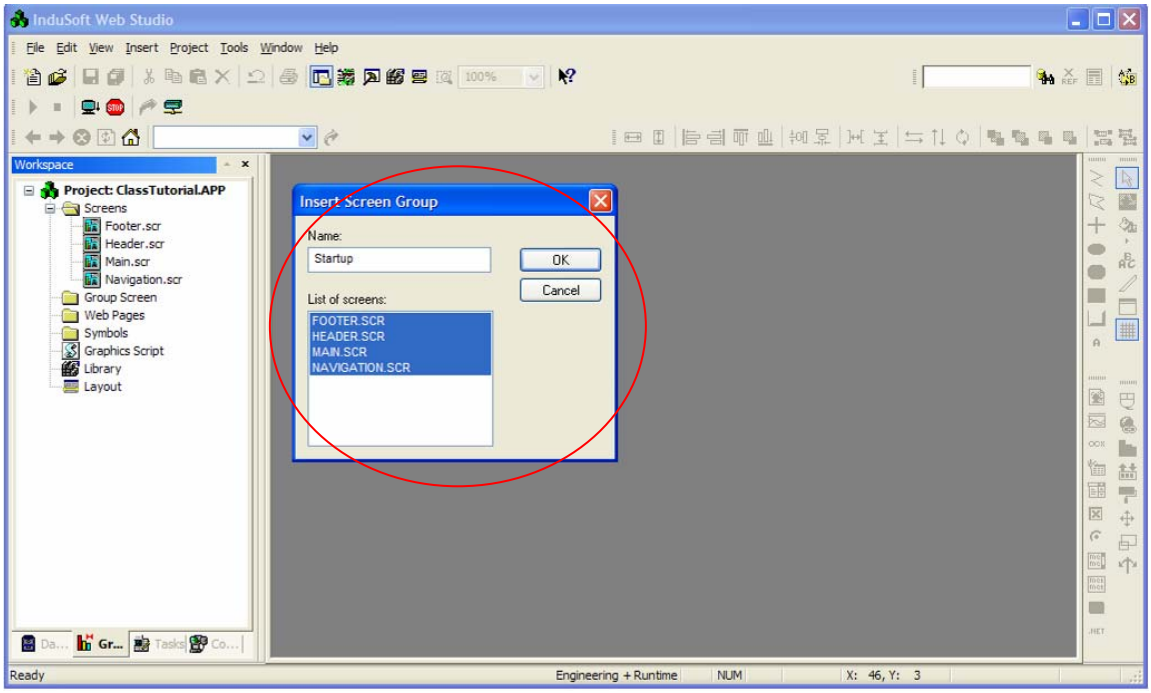
In this example, the Child Window “Main” will be replaced by other Child Windows of the same size as Main. Buttons (or other Objects) with a Command Property in the Navigation Child Window can be used to open new Child Windows, replacing Main (or another Child Window opened in its place). You can create other Child Windows the same size and style as Main (e.g. Main1, Main2, etc.).

Once the initial group of Child Windows (Header, Footer, Navigation and Main) have been created and saved as Screens, we will group them together. This grouping is called a Group Screen and is how the Child Windows come together to make what looks like a Standard Screen. However, any of the Child Windows can be replaced with a new Child Window.

To create a Group Screen once the Child Windows (Screens) are created, position your mouse over the Group Screen folder and right click on it.



When the **Insert Screen Group** Dialog Box opens, enter a name for the Screen Group (e.g. “Startup”) and click on each of the Child Windows in the List of Screens box that you want to be part the specifed Screen Group. Click on **Ok** to save the Screen Group.



Selecting Child Windows (Screens) to be part of a Screen Group

If your application uses a Screen Group, it is usually opened when the runtime is started. Whereas a Standard Screen defines a Screen (with a .scr file extension) to be the Startup Screen, you simply define a Screen Group (with a .sg file extension) to be the Startup Screen. Then, during operation, you have IWS objects that provide navigation among the various Child Windows, opening the individual Child Windows (Screens with a .scr file extension) via Command Properties. The new Child Window that is “opened” replaces and existing Child Window being displayed

Tips & Tricks for using MDI Child Windows (Screens) and Screen Groups:

1. A Screen Group is just a collection of individual Screens (Child Windows) that are combines in an arrangement that fits within the resolution of the PC’s display.
2. Screen Groups are saved with a .sg file extension (e.g. Startup.sg), whereas Screens are saved with a .scr file extension. Screen Group files are saved in the \Screen subfolder of your application directory.
3. To save an blank Child Windows (Screen), you must use **SAVE AS in the IWS Main Tool Bar (e.g. FILE → SAVE AS).** You cannot right click on the Screen name tab (above the Screen), use the **SAVE** command in the IWS Main Tool Bar (e.g. **FILE → SAVE),** or use the Save icon unless you first placed one or more Objects (IWS Objects, ActiveX or .NET Controls) on the Screen.
4. IWS does not let you save a blank Child Window (Screen). You must first place one or more Objects (IWS Objects, ActiveX or .NET Controls) on the Child Window before saving.

Tips & Tricks for using MDI Child Windows (Screens) and Screen Groups, continued:

6. To rename a Screen Group, first close your IWS application and the development environment. Open the \Screen subfolder in your application directory. A Screen Group will have only one (1) file associated with it, and uses a .sg file extension. Rename this file to a new name. The Screen Group name must be unique from the other Screen Group name(s) and must follow standard Windows file naming conventions.
7. If you want to specify a Screen Group to be opened automatically when your application runtime is started, this can be done in one of two ways:
   a. Open the **Group Screen** folder. Place your mouse on the specific Screen Group you want to use as your startup Screen Group and right click. You will see an option appear that allows you to **Set as Startup**. Choose this option.

      NOTE: When right clicking on a specific Screen Group, you will see another option called "Set Open Order". This lets you define the order by which Child Windows open when the Screen Group is loaded. This option is specified in case a Child Window has a Script (IWS Worksheet or VBScript Screen Script) associated with it. This subject will be covered later in more detail.

   b. In the Menu Bar, select **Project**, then **Settings**, then **Runtime Desktop.** There is a dialog box with a pull down list of all your Screens and Screen Groups. Select the Screen Group you want as your startup Screen Group.

      NOTE: In order to have your startup Screen Group be the Screen Group that starts whenever your application runtime is started, all other Screens and Screen Groups in the development environment must be closed.
8. IWS Objects with a Command Property must be used to navigate to other Child Window (Screens) or Screen Groups.

Exercise: Creating a MDI Screen Group

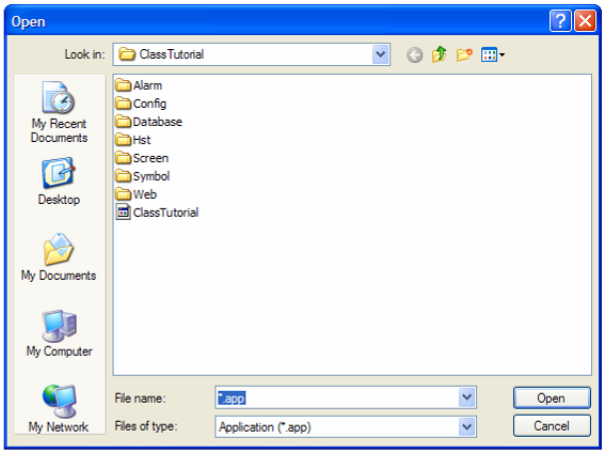
In this exercise, we will add an MDI Screen Group using the four (4) Child Window Screens. This exercise builds on the previous exercise. Right now, we are merely building Screens that will be used by future Exercises to build an application.

Use the following procedures to create an MDI Group Screen:

- ☐ Open the IWS development environment (if it not already opened).
- ☐ Open the ClassTutorial project. If an application other than ClassTutorial.app is open, first save this application. This can be done by several methods:
  - If your application Screen(s) have not previously been saved or if you want to rename a Screen: Open the application Screen then click on **File** then **Save As** in the IWS Main Menu Bar. Repeat for each of the application Screen(s). if they have not been previously saved
  - If you have already named all your screens and just want to close any open Screens and/or save changes to the Screens: In the IWS Main Menu Bar, click on **File → Close All.** You will be prompted to save any unsaved changes to application Screens.

    Note: After you click on **File,** you may need to select the down arrow at the bottom of the drop-down box to view the **Close All** option.

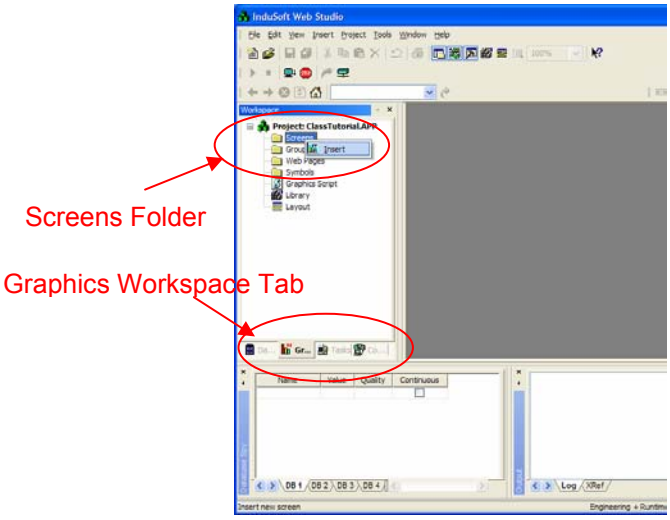
Open Project Icon



Open a new Project

  - If you have already named all your screens and just want to open the ClassTutorial.all application, first saving any changes to the Screens in your existing application: In the IWS Main Menu Bar, click on **File → Open Project** or click the Open Project icon. Open the ClassTutorial Application folder and click on the ClassTutorial.app file. Click on the **Open** button in the Open Project dialog box.

- ☐ In the **Workspace**, select the **Graphics** tab.
- ☐ Right-click the **Screens** folder and select **Insert** or select **Insert →Screen** from the IWS Main Menu Bar.



Select Graphics Workspace Tab

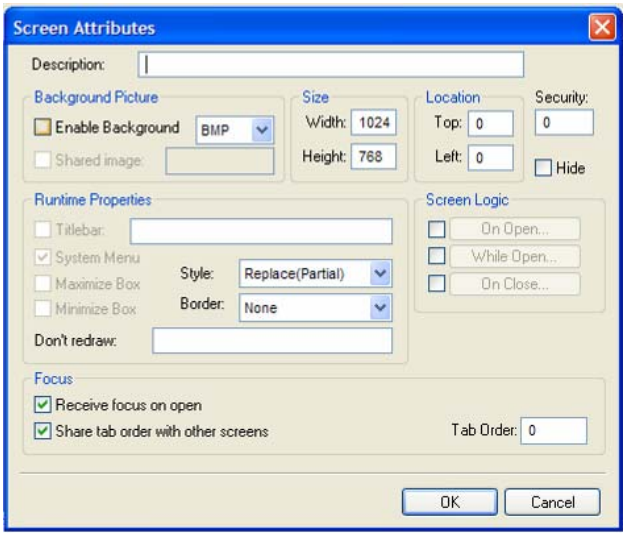
- ☐ When the Screen Attributes dialog displays, type **Header** into the **Description** field.
- ☐ Set the following Size Properties:

  Width: 1024

  Height: 150
- ☐ Set the following Location Properties:

  Top: 0

  Left: 0
- ☐ Click the **Ok** button to close the Screen Attributes dialog box.
- ☐ Set the background color to light gray (see the Create Standard Screen Exercise)
- ☐ Select **File →Save As** on the **Main Menu Bar** to save the new screen. When the **Save As** dialog displays, type **Header.scr** (or **Header**) into the **File name** field.



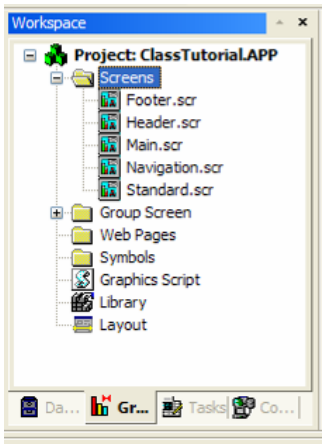
Screen Attributes Dialog

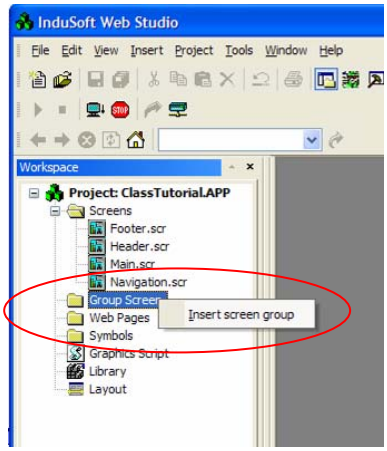
Note: The .scr file extension is automatically used.

Repeat the above procedures, creating the Footer, Navigation and Main Screens using the Screen Attributes as specified. Type the Child Window Screen name into the **Decription** field.

Child Window Screen Name	Width	Height	Top	Left	Style
Header	1024	150	0	0	Replace (Partial)
Footer	1024	100	668	0	Replace (Partial)
Navigation	150	518	150	0	Replace (Partial)
Main	874	518	150	150	Replace (Partial)

- ☐ Open the **Screens** folder to verify the Screens are saved.
- ☐ Close the **Screens** folder.
- ☐ Right click on the **Group Screen** folder (located below the Screens folder.
- ☐ Click on **Insert Screen Group**



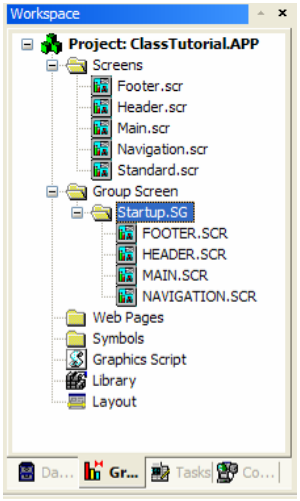


- An **Insert Screen Group** dialog box will appear. Enter **Startup** in the **Name** field.

  Note: In this example, the Screen Group will be saved as Startup.sg. IWS automatically inserts the .sg file extension. This file is stored in the \Screen subfolder in the Application directory.

- Click on the individual Screens you want to include in the Screen Group. In this example, we want to include the Child Window (Screens) Footer, Header, Main and Navigation.
- Click **Ok.**
- Open the Group Screen folder to verify the Startup.sg is located in the Group Screen folder.
- Open the Startup.sg folder (if it is not already opened). Verify that all the Child Window (Screens) are part of the Screen Group Startup.sg.
- Right click on the Startup.sg folder.
- Click on the **Set as Startup** option.

  Note: This option will



IWS Toolbars for Screen Objects

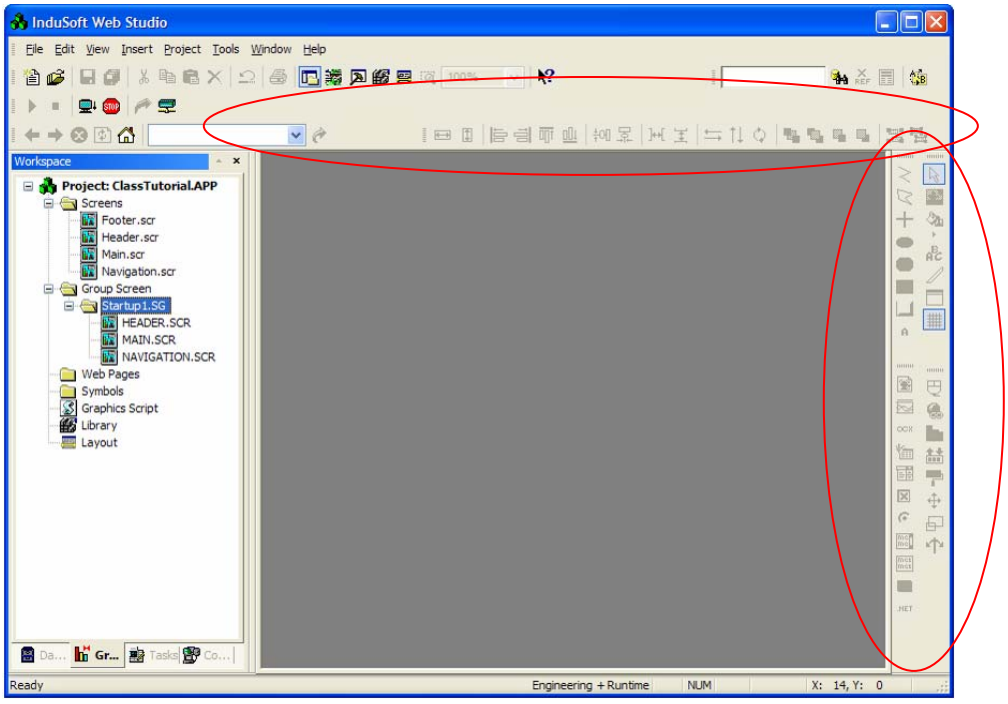
So far, we have created Screens and Screen Groups. But by themselves, Screens don't do much. IWS provides a number of Objects accessible from the Toolbar that can be used with Screens to display and set tag values, provide Dynamic Properties to the Objects, and other functions. This section of the Training Manual reviews some of these Objects and Properties. Specifically, in this section we will review the following Objects and Properties:

- **Static Objects**
  Line, Ellipse (includes the Ellipse, Chord, Arc, and Ring), Rounded Rectangle, Filled Rectangle, Button, Text, Open Polygon, and Closed Polygon.
- **Align & Distribute Toolbar**
- **Dynamic Properties**
  Command, Bar Graph, Text I/O, Position (and Visibility), Size, Rotation, Dynamic Color Control, and HyperLink
- **Mode Toolbar**
  Selection (of an Object on the Screen), Fill Color, Fonts, Line Color, BitMap Editor, Change Color not covered by Bitmaps, Toggle Grid

Other Toolbars items (e.g. Active Objects such as Alarms, Trends, etc.) will be covered later in this manual.

Notes:

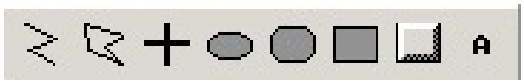
- The Toolbars for Static Objects, Dynamic Properties and Mode appear on the right side of the IWS development environment by default. However, they can be relocated (via drag & drop) to the Tool Bar area below the Main Menu Bar. If any Toolbars are not displayed when the IWS development environment is opened, you can display them by clicking on **View** in the Main Menu Toolbar, then select **Toolbars** and select the **Toolbar** of interest that is not shown. If the Toolbar has a lit checkbox beside it, the Toolbar will be displayed in the IWS development environment.
- It a Tool is grayed out, it is not available in the current mode or with the current Object



Toolbar Locations

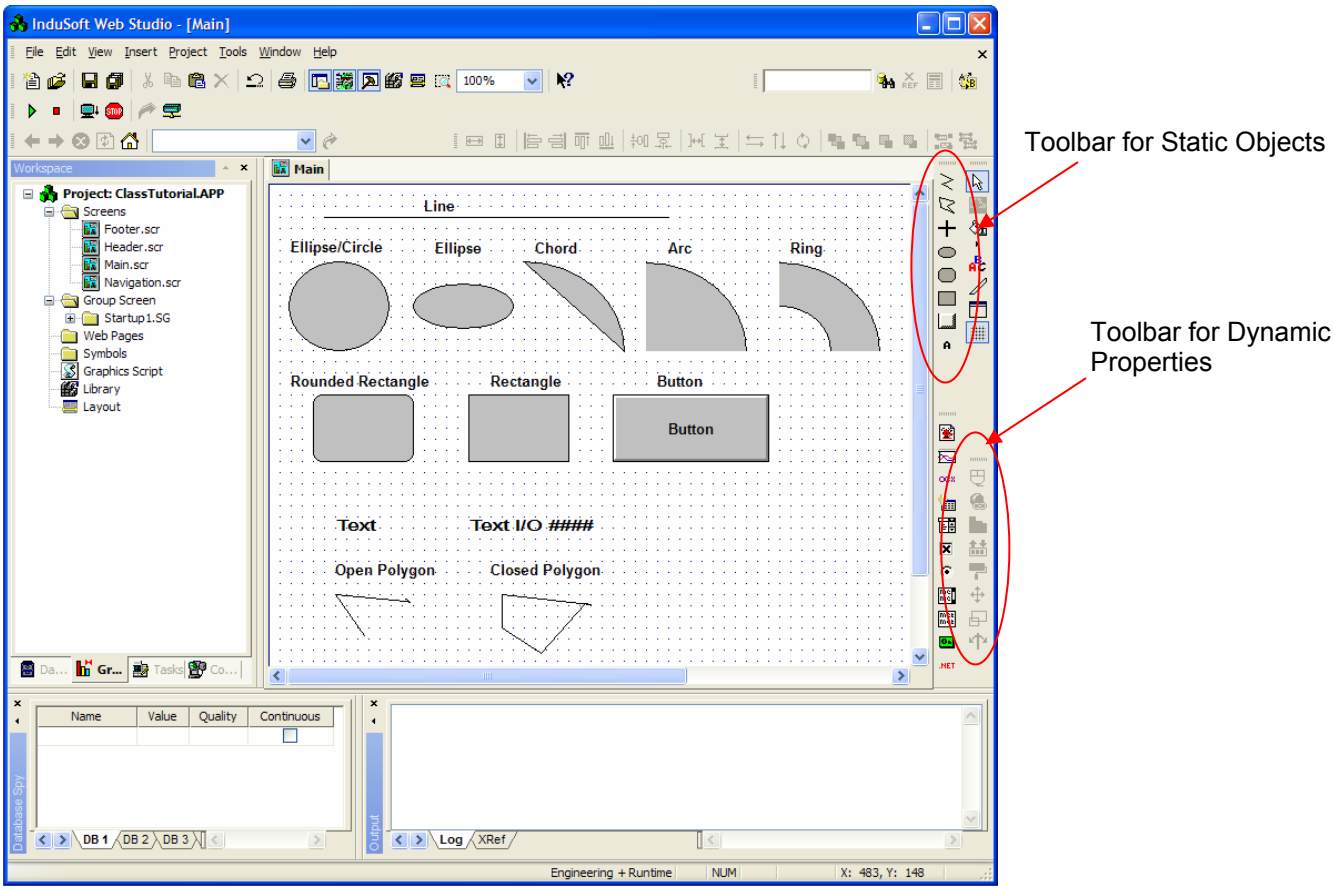
IWS Static Objects and Properties

The **Static Objects** toolbar provides buttons that you can use to insert Static Objects on a Screen. You must first have a Screen open, and then you can select from the Static Object tools in the Toolbar.



Static Objects Toolbar

The following picture shows the Static Objects that can be created in the IWS development environment.



IWS Static Objects

Notes:

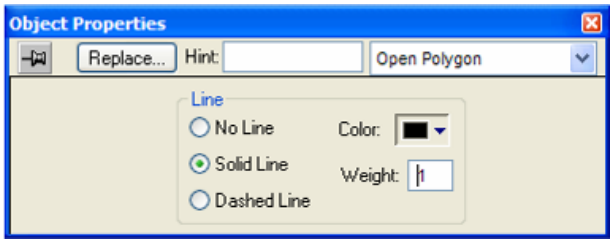
- To use the Static Objects toolbar to insert an Object on a Screen (or Child Window), you must first open the **Screens** folder in the **Graphics** workspace and select the screen you want to use or insert a new **Screen.**
- When you select a tool, it will become “lit”. As long as it is lit, the tool will remain selected.
- Tools that are “grayed out” are not available in the present mode
- After you select a tool, you move your mouse cursor to the area of the screen where you want to place the Object. A small cross will appear, signifying where the Object will be placed.

Open Polygon

The **Open Polygon** tool will draw a multi-sided polygon object that is not necessarily closed (i.e. the beginning and ending vertices are not connected unless you connect them). To insert an Open Polygon:

- select the Open Polygon tool icon (as shown)
- place your mouse cursor on the screen at the starting point and click the left mouse button
- move the mouse cursor to the next location and click again to form the next vertices
- repeat this process until you have created the desired polygon shape
- double click to stop drawing the Open Polygon

Open Polygon parameters



Replace: Not applicable without one or more Dynamic Properties.

Hint: If the Hint box contains any text, the Hint will be displayed at runtime when the mouse cursor is over the Open Polygon Object.

Line: Specifies the outside line style, color & weight (thickness)

Color: Specifies the color of the border line.

Weight: Specifies the thickness (in pixels) of the border.

Note:

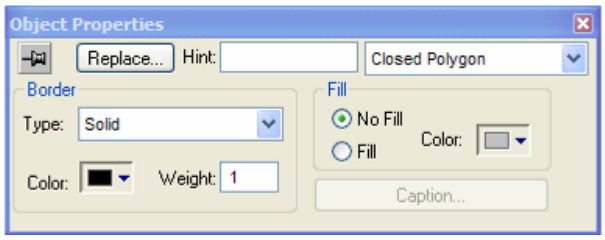
- When you double click on the last vertices of the Open Polygon, you must double click rapidly, otherwise another vertices will be drawn. Another alternative is to click the right mouse button (which opens a options menu) and then move the mouse cursor to another area of the screen and click the left mouse button.

Closed Polygon

The **Closed Polygon** tool will draw a multi-sided polygon object that is closed (i.e. the beginning and ending vertices are automatically connected). To insert a Closed Polygon:

- select the Closed Polygon tool icon (as shown)
- place your mouse cursor on the screen at the starting point and click the left mouse button
- move the mouse cursor to the next location and click again to form the next vertices.
- repeat this process until you have created the desired polygon shape
- double click to stop drawing the Closed Polygon

Closed Polygon parameters



Replace: Not applicable without one or more Dynamic Properties.

Hint: If the Hint box contains any text, the Hint will be displayed at runtime when the mouse cursor is over the Closed Polygon Object.

Type: Combo box used to specify the border style (No Line, Solid Line or Dashed Line)

Color: Specifies the color of the border line.

Weight: Specifies the thickness (in pixels) of the border.

Fill: Indicates if the Object is to be filled or not

Color: Specifies the color of the border line

Caption: The caption is not functional with this Object, despite being grayed out.

Notes:

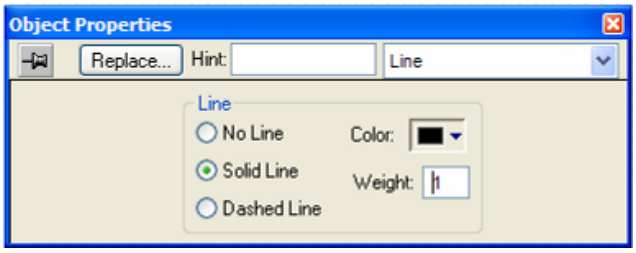
- When you double click on the last vertices of the Closed Polygon, you must double click rapidly, otherwise another vertices will be drawn. Another alternative is to click the right mouse button (which opens a options menu) and then move the mouse cursor to another area of the screen and click the left mouse button.
- Use the **Selection** Tool to select the Closed Polygon object. The Closed Polygon object will have Handles at the vertices of the Closed Polygon. You can grab and move these Handles to reshape the Closed Polygon.

Line

The **Line** tool will draw an orthogonal line object in the drawing area. To insert a Line:

- select the Line tool icon (as shown)
- place your mouse cursor on the screen at the starting point and click the left mouse button
- with the left mouse button held down, drag the mouse cursor to the ending point of the line and release the left mouse button

Line parameters



Replace: Not applicable without one or more Dynamic Properties.

Hint: If the Hint box contains any text, the Hint will be displayed at runtime when the mouse cursor is over the Line Object.

Line: Specifies the line style (No Line, Solid Line, Dashed Line)

Color: Specifies the color of the line.

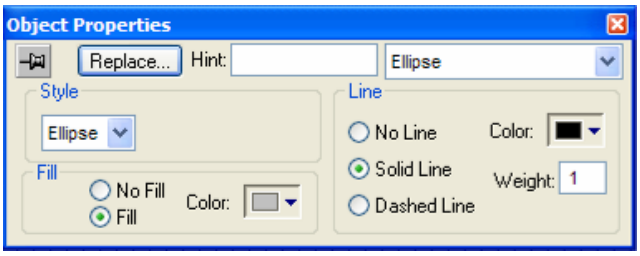
Weight: Specifies the thickness (in pixels) of the line.

Ellipse

The **Ellipse** tool can be used to draw circle, ellipse, chord, arc and ring objects. To insert an Ellipse Object:

- select the Ellipse tool icon (as shown)
- place your mouse cursor on the screen at the starting point and while holding the left mouse button down, drag the mouse cursor to create an oval shape. Release the left mouse button.
- if you want to change the shape double click on the Ellipse Object to select it, and use the Object Properties dialog box to change the shape of the object to a chord, arc or ring.

Ellipse parameters



Replace: Not applicable without one or more Dynamic Properties.

Hint: If the Hint box contains any text, the Hint will be displayed at runtime when the mouse cursor is over the Ellipse Object

Style: Specifies Ellipse, Chord, Arc, or Ring. If you specify the Chord, Arc or Ring style, an orientation combo box will appear.

Fill: Indicates if the Object is to be filled

Color: Specifies the fill color, if Fill is selected.

Line: Specifies the outside line style (No Line, Solid Line or Dashed Line)

Color: Specifies the color of the line.

Weight: Specifies the thickness (in pixels) of the line.

Notes:

- Use the Ellipse shape to draw circles and shapes.
- If you want to change the ellipse style to a Chord, Arc or Ring, double click on the Ellipse Object to select it, and use the Object Properties dialog box to change the shape.
- If you select the Chord, Arc or Ring style, you can specify the orientation of the shape; e.g. Left-Bottom, Left-Top, Right-Top and Right-bottom.

Rounded Rectangle

The **Rounded Rectangle** tool can be used to draw empty or filled rectangle objects with rounded edges. To insert an Rounded Rectangle Object:

- select the Rounded Rectangle tool icon (as shown)
- place your mouse cursor on the screen at the starting point and while holding the left mouse button down, drag the mouse cursor to create an rounded rectangle shape. Release the left mouse button.

Rounded Rectangle parameters

Replace: Not applicable without one or more Dynamic Properties.

Hint: If the Hint box contains any text, the Hint will be displayed at runtime when the mouse cursor is over the Rounded Rectangle Object

Type: Specifies the border style (None, Dashed, or Solid Line)

Color: Specifies the color of the line.

Weight: Specifies the thickness (in pixels) of the line

Fill: Indicates if the Object is to be filled

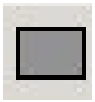
Color: Specifies the fill color, if Fill is selected.

Caption: The caption is not functional with this Object, despite being grayed out.

Notes:

- After selecting (left mouse click) the Rounded Rectangle object, an additional anchor appears that is offset from the other anchors (that are positioned on the corners and middle of the rectangle). This anchor can be adjusted (moved) to effect the shape (roundness) of the corners.

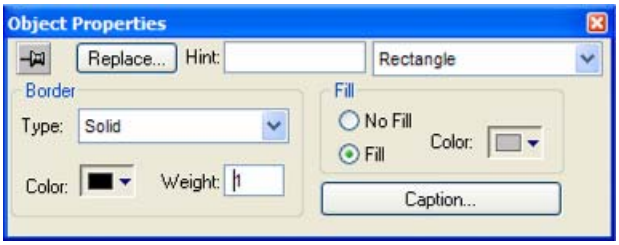
Rectangle



The **Rectangle** tool can be used to draw empty or filled rectangle objects with square edges. To insert a Rectangle Object:

- select the Rectangle tool icon (as shown)
- place your mouse cursor on the screen at the starting point and while holding the left mouse button down, drag the mouse cursor to create an rounded rectangle shape. Release the left mouse button.

Rectangle parameters



Replace: Not applicable without one or more Dynamic Properties.

Hint: If the Hint box contains any text, the Hint will be displayed at runtime when the mouse cursor is over the Rectangle Object

Type: Specifies the border type (None, Solid, Dashed, Raised or Sunken)

Color: Specifies the color of the line.

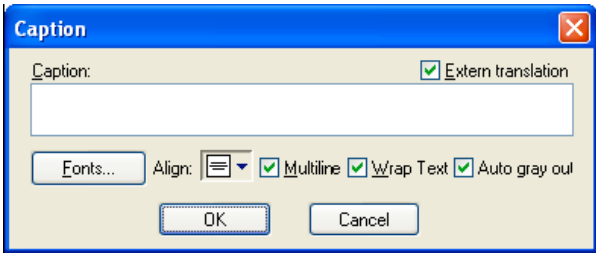
Weight: Specifies the thickness (in pixels) of the line

Fill: Indicates if the Object is to be filled or not

Color: Specifies the fill color, if Fill is selected.

Caption: Adds a caption to the rectangle Object (see below)

Caption parameters



Caption: Text than you want to display in the Rectangle Object

Font: Specifies the font to use in the Caption

Align: Specifies the alignment of the Caption in the Rectangle.

Multiline: If checked, allows Caption to be on more than one line.

Wrap Text: If checked, will automatically wrap text if necessary

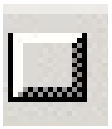
Auto Gray Out: If checked, will gray out the Caption of there is a Command Dynamic Property applied to the Rectangle and the Command Dynamic is disabled by the **Disable** field or due to the Security System (Operator not at appropriate Security Level)

Extern Translation: If checked, allows runtime translation of the Caption

Note:

- In addition to Text, the Caption can display IWS tag values. Use the curly braces { } around an IWS tag. E.g. {myTag}.

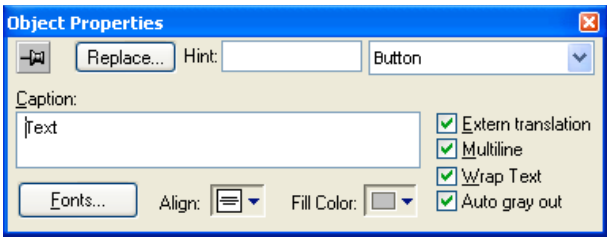
Button



The **Button** tool can be used to draw simple button objects on the Screen. To insert a Button object:

- select the Button tool icon (as shown)
- place your mouse cursor on the screen at the starting point and while holding the left mouse button down, drag the mouse cursor to create an rounded rectangle shape. Release the left mouse button.

Button parameters



Replace: Not applicable without one or more Dynamic Properties.

Hint: If the Hint box contains any text, the Hint will be displayed at runtime when the mouse cursor is over the Button Object

Caption: Adds a caption to the Button Object

Font: Specifies the font to use in the Caption

Align: Specifies the alignment of the Caption in the Button

Fill Color: Specifies the fill color. Buttons are always filled.

Extern Translation: If checked, allows runtime translation of the Caption

Multiline: If checked, allows Caption to be on more than one line.

Wrap Text: If checked, will automatically wrap text if necessary

Auto Gray Out: If checked, will gray out the Caption of there is a Command Dynamic Property applied to the Rectangle and the Command Dynamic is disabled by the **Disable** field or due to the Security System (Operator not at appropriate Security Level)

Note:

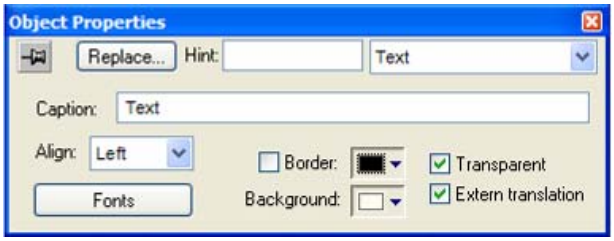
- The Button Object is frequently used to activate other Objects, navigate to other Screens or to change Tag values. This is done by adding the Command Property to the button. We will see how to add Command Properties in a later section of this manual.
- In addition to Text, the Caption can display IWS tag values. Use the curly braces { } around an IWS tag. E.g. {myTag}.

Text

The **Text** tool can be used to create text objects. To insert a text object:

- select the Text tool icon (as shown)
- place your mouse cursor on the screen where the text is to be located. A cursor will display
- type a line of text

Text parameters



Replace: Not applicable w/o one or more Dynamic Properties.

Hint: If the Hint box contains any text, the Hint will be displayed at runtime when the mouse cursor is over the Text Object

Caption: Adds a caption to the Text Object

Align: Aligns the Caption to the Left, Right or Center

Fonts: Selects the Font, Color, Size, Style, Effects

Border: If checked, specifies that the Text Object is to have a Border and allows you to select the color.

Background: Specifies the background color.

Transparent: If checked, the background color is transparent, regardless of the color selected.

Extern Translation: Enables Runtime Translation for the Caption

Note:

- With The Text Object is frequently used to display tag values, or to input tag values. This is done by adding the Text I/O Property to the Text Object. We will see how to add the Text I/O Property in a later section of this manual.
- The Caption field of the Text Object cannot be used to display a Tag value as other Caption fields do (i.e. {myTag}). Use the Text I/O Dynamic instead.

Manipulating Static Objects

Once you place a Static Object on a Screen, you may want to manipulate it further. Some examples of this include:

- **Moving an Object**
  Moving an object is straight forward using click & drag. Simply select the object by a left mouse click, and while holding the left mouse button down, drag the selected object to its new position.

  Notes:
  - When you have the clicked on the selected object and have the left mouse button held down, a small cross will be shown near the mouse cursor indicating you are dragging the object.
  - To reposition a Line Object, click in the middle of the Line so as not to just reposition it's endpoint.

- **Replicating an Object**

  To replicate an object, there are two methods you can use:

  a) Position the mouse cursor over the Object you want to copy. With the Control (**Ctrl**) key held down, click and hold the right mouse button, and drag the copied Object to its new position.

  b) Select an object and use copy & paste. To select an object, position your mouse cursor over the object and click the left mouse button. From the Main Menu Bar, select **Copy** (or press **Ctrl C**) and then **Paste** (or press **Ctrl V**). Move the new object to the desired position on the Screen.

  Note:
  - If you are going to add Dynamic Properties to an object, it is usually best to do so before replicating the object. Doing this will save you additional steps.

- **Resizing/Reshaping an Object**

  You can change the size and shape of the Static Objects, but the method of doing so depends on the specific object. After being selected, most objects will display handles on the four corners and in the middle of the object borders (e.g. ellipse, rounded rectangle, rectangle, button, text). For these objects, grab one of the handles and drag them (with the left mouse cursor held down) to a new position

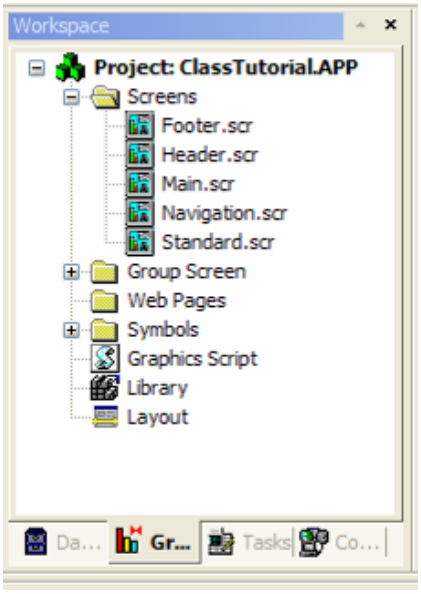
  For open polygon and closed polygon objects, after the object is selected the vertices will be displayed. Position the mouse cursor over one of the vertices, click and hold down the left mouse button. While holding down on the left mouse button, drag the vertices to the desired position on the Screen. Release the left mouse button when the vertices is correctly positioned

  Note:
  - With a closed polygon object, a connecting line between the first and last vertices is automatically drawn.
  - It is not recommend to change the height or width of a Text Object by resizing the text object. Instead, change the font of the text to keep the text characters in their proper proportions.

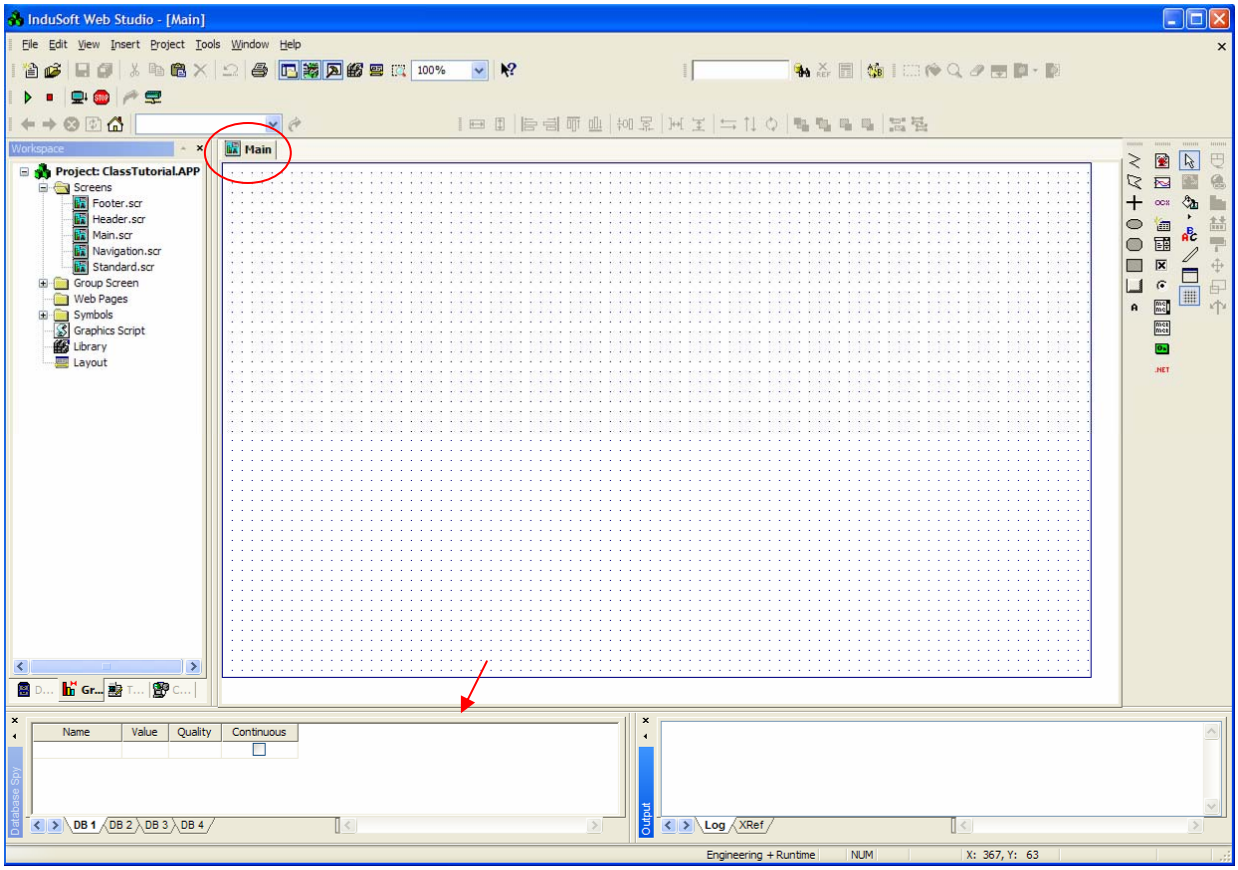
.

Exercise: Using Static Objects

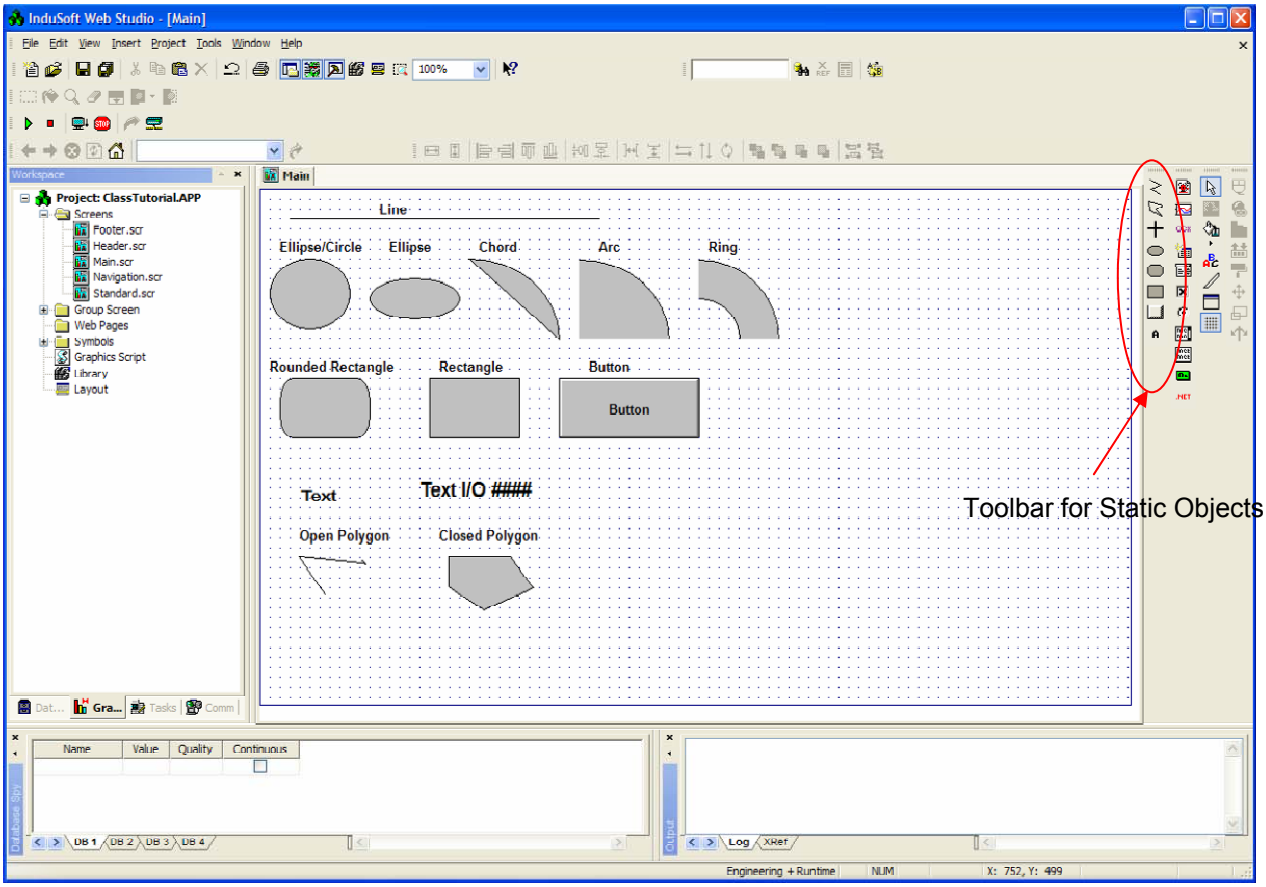
Objective: The objective of this exercise is to place various static objects on a Screen, the MDI Child Window (**Main.scr)**. we previously created The Static Objects by themselves won't do much, but in the next Exercise, we will add Dynamic Properties to these Static Objects.



- ☐ Start the IWS development environment (if it isn't already open). Be sure you have opened and are using the Project **ClassTutorial**.
- ☐ Select the **Graphics Workspace** tab.
- ☐ Double click on the **Screens** folder to expand it (or click on the expand box)
- ☐ Double click on the **Main.scr** Screen. This is a MDI Child Window that we had previously created. It will be used later as one Screen in a Group of Screens (Screen Group).
- ☐ You should now see a blank screen in the main Screen/Worksheet area. A small box above the Screen indicates that the **Main.scr** Screen is open.
- ☐ If the full screen is not viewable, grab and move the bar that separates the Database Spy and LogWin down until the full Screen is viewable (click, hold, and drag). This is not necessary, but makes development simpler when you can see the entire Screen.



Now, we are going to place a number of Static Objects on this screen. When the Screen is done, it will look as follows:

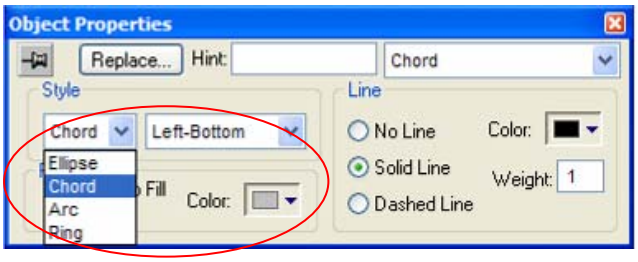


Note:

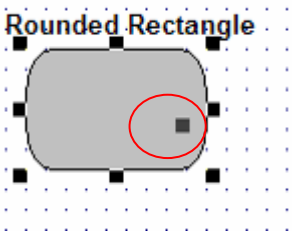
- As you move your mouse over the various icons in the Static Objects Toolbar, a ToolTip will appear indicating the Tool your mouse is over.

- ☐ Move your mouse over the Static Object Toolbar and select the **Line** tool (click it once). Position your mouse cursor near the top left corner of the Screen. Click and hold your left mouse button, and drag your mouse horizontally to the right, drawing a horizontal line. Release the left mouse button. If the line is not horizontal, position your mouse on one of the line, click, hold and drag the line end until the line is horizontal. If you position your mouse somewhat near the center of the line, you can click, hold and drag the line, moving it to a new position while maintaining its shape.
- ☐ In the Static Objects Toolbar, select the **Text** tool. Position your mouse above the line, click the left mouse button once. A small vertical cursor should be displayed on the Screen. Type **Line**, and then move your mouse away from the text and click the left mouse button.
- ☐ Select the **Ellipse** tool. Position your mouse cursor below the Line and click the left mouse button. Draw a circle, keeping the Ellipse object in a circular shape. Put a **Text** object (caption **Ellipse/Circle**) above it.
- ☐ Select the **Ellipse** tool again. Position your mouse cursor below the Line and to the right of the Circle. Click the left mouse button. Draw an ellipse, keeping the Ellipse object in an elliptical shape. Put a **Text** object (caption **Ellipse**) above it.

- Select the **Ellipse** tool again. Position your mouse cursor below the Line and to the right of the Ellipse. Click the left mouse button. Draw an ellipse. Now, double click the ellipse object you just drew. In the **Object Properties** dialog box, select the **Style** as a **Chord**, with the orientation **Left-Bottom**. By positioning the mouse over the corner or middle handles of the object and clicking & dragging, change the shape of the Chord until it matches the picture. Put a **Text** object (caption **Chord**) above it.



- Select the **Ellipse** tool again. Position your mouse cursor below the Line and to the right of the Chord. Click the left mouse button. Draw an ellipse. Now, double click the ellipse object you just drew. In the **Object Properties** dialog box, select the **Style** as a **Arc**, with the orientation **Left-Bottom**. By positioning the mouse over the corner or middle handles of the object and clicking & dragging, change the shape of the Arc until it matches the picture. Put a **Text** object (caption **Arc**) above it.
- Select the **Ellipse** tool again. Position your mouse cursor below the Line and to the right of the Arc. Click the left mouse button. Draw an ellipse. Now, double click the ellipse object you just drew. In the **Object Properties** dialog box, select the **Style** as a **Ring**, with the orientation **Left-Bottom**. By positioning the mouse over the corner or middle handles of the object and clicking & dragging, change the shape of the Ring until it matches the picture. Put a **Text** object (caption **Ring**) above it.
- Select the **Rounded Rectangle** tool. Position your mouse cursor below the Ellipse/Circle object and click the left mouse button. Draw a rounded rectangle. **Notice the extra handle that is not in a corner or the middle of the object. If you click and drag this handle, you can change the "roundness" of the rectangle.** Put a **Text** object (caption **Rounded Rectangle**) above it.



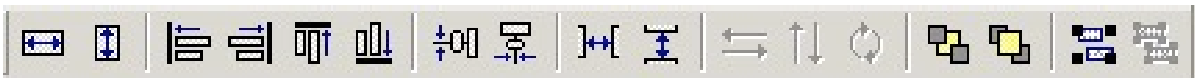
- Select the **Rectangle** tool. Position your mouse cursor to the right of the Rounded Rectangle object and click the left mouse button. Draw a rectangle. If you click and drag this hangle, you can change the "roundness" of the rectangle. Put a **Text** object (caption **Rectangle**) above it.
- Select the **Button** tool. Position your mouse cursor to the right of the Rectangle object and click the left mouse button. Draw a button. Put a **Text** object (caption **Button**) above it. **Notice that the button has a caption that is placed in the middle of the button. If you double click on the button to open its Object Properties, you can change the button caption.**
- Below the Rounded Rectangle object, we will put a **Text** object with the caption **Text.**
- To the right of the Text object, we will put another **Text** object and give it the caption **Text I/O ####** After entering the caption, click on the **Fonts** button and change the font size to 16.
- Select the **Open Polygon** tool. Position your mouse cursor below the Text object and click the left mouse button. This will define the starting point of the polygon. Move your mouse cursor and left mouse click when you want to insert another vertices. (Be sure not to make a left mouse click until you are ready – otherwise you will insert a vertices unintentionally). When finished, you can right mouse click (don't left mouse click on the last vertices). You can select the Open Polygon object, and select the handle on one of its vertices and drag it (keeping the left mouse button held down), changing the shape of the Open Polygon. By clicking (and holding) the Open Polygon on a line that is not a handle (or vertices), you can move the Open Polygon to a new position, keeping its shape.
- Select the **Closed Polygon** tool. Position your mouse cursor below the Text object and click the left mouse button. This will define the starting point of the polygon. Move your mouse cursor and left mouse click when you want to insert another vertices. (Be sure not to make a left mouse click until you are ready – otherwise you will insert a vertices unintentionally). When finished, you can right mouse click (don't left mouse click on the last vertices). You can select the Closed Polygon object, and select the

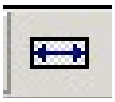
handle on one of its vertices and drag it (keeping the left mouse button held down), changing the shape of the Closed Polygon. By clicking (and holding) the Closed Polygon on a line that is not a handle (or vertices), you can move the Closed Polygon to a new position, keeping its shape.

- ☐ Save the Screen (to save our changes)

Align & Distribute Toolbar

The **Align & Distribute** toolbar allows you to edit Screen Objects. To use the tools in this toolbar, you must have one or more screen objects selected.



Resize Height	The **Resize Height** button will set the height of all the selected objects to the height of the last object selected. If only one object is selected, pressing the **Resize Height** button will set the height of the selected object to its width.	
Resize Height	The **Resize Height** button will set the height of all the selected objects to the height of the last object selected. If only one object is selected, pressing the **Resize Height** button will set the height of the selected object to its width.	
Align Left	The **Align Left** button aligns all selected objects to the left edge of the last object selected.	
Align Right	The **Align Left** button aligns all selected objects to the right edge of the last object selected.	
Align Top	The **Align Left** button aligns all selected objects to the top edge of the last object selected.	
Align Bottom	The **Align Left** button aligns all selected objects to the bottom edge of the last object selected.	
Center Vertically	The **Center Vertically** button aligns all selected objects to the vertical center of the last object selected.	
Center Horizontally	The **Center Horizontally** button aligns all selected objects to the horizontal center of the last object selected.	
Evenly Space Horizontally	The **Evenly Space Horizontally** button puts an equal amount of horizontal space between a series of two or more objects.	
Evenly Space Vertically	The **Evenly Space Vertically** button puts an equal amount of vertical space between a series of two or more objects.	
Flip Horizontally	The **Flip Horizontally** button inverts the selected object horizontally, rotating it around an imaginary line through its horizontal center. Creates a mirror image of the original object.	
Flip Vertically	The **Flip Vertically** button inverts the selected object vertically, rotating it around an imaginary line through its vertical center. Creates a mirror image of the original object.	
Rotate	The **Rotate** button rotates the selected object 90º clockwise.	

Move To Back

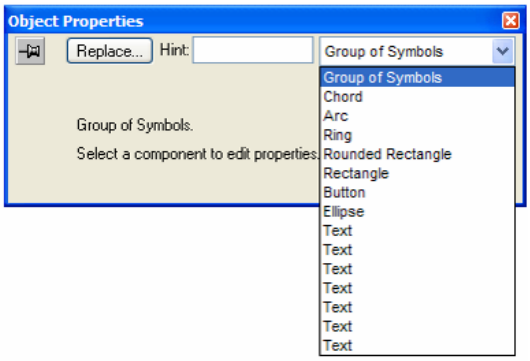
The **Move To Back** button moves a selected object (or objects) behind other objects on the Screen. IWS assigns the object the lowest ID# and moves that object behind all other objects on the Screen.

Move To Front

The **Move To Front** button moves a selected object (or objects) in front of all other objects on the Screen. IWS assigns the object the highest ID# and moves that object in front of all other objects on the Screen.

Group

The **Group** button combines multiple objects into a single object. This facilitates object selection and manipulation. To access the Object Properties of an object within the group object, double click on the group object and select the object of interest to access its properties.



Ungroup

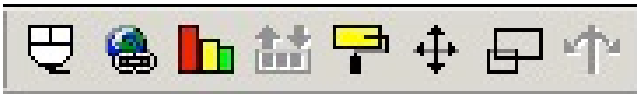
The **Ungroup** button separates multiple objects into a individual components.

Notes:

- If you use the Group tool, you can still access each of the Objects Object Properties dialog box, but you will need to select the object of interest from the Group of Objects.
- Not all Screen Objects can be used with every tools in the Align & Distribute Toolbar. For example, Linked Symbols cannot be rotated.

Dynamic Properties

Dynamic Properties allow an object to change its characteristics (color, shape, display value, etc.), generally used to provide an Operator with a visual indication of a tag's value representing the state of a machine or process.



IWS provides a number of Dynamic Properties that can be used with Static and Dynamic Objects. One or more Dynamic Properties can be added to an object. The following chart shows the objects provided by IWS and the Dynamic Properties that can be used with them.

Object\ Dynamic	Command	Bar Graph	Text I/O	Position	Resize	Rotation	Colors	Hyperlink
Line	✓			✓	✓	✓	✓	✓
Ellipse	✓	✓		✓	✓		✓	✓
Rounded Rectangle	✓	✓*		✓	✓		✓	✓
Filled Rectangle	✓	✓		✓	✓		✓	✓
Button	✓			✓	✓		✓	✓
Text	✓		✓	✓	✓		✓	✓
Open Polygon	✓			✓	✓	✓	✓	✓
Closed Polygon	✓	✓		✓	✓	✓	✓	✓
Alarm/Event Control				✓	✓			
Trend Control				✓	✓			
ActiveX Control	✓			✓	✓			
.NET Control	✓			✓	✓			
Grid Object				✓	✓			
Combo Box	✓			✓			✓	
Check Box	✓	✓		✓	✓		✓	✓
Radio Button	✓	✓		✓	✓		✓	✓
List Box				✓	✓			✓
Smart Message				✓	✓			✓
Pushbutton				✓	✓			✓

List of Objects and their available Dynamic Properties

* not available with Windows CE

Notes:

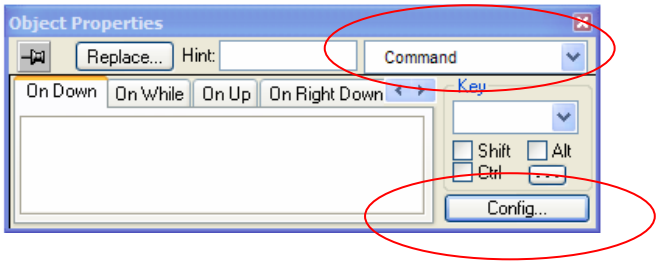
- To add a Dynamic Property to a Static Object, you should do the following:
  - First, select a Static Object on a Screen that you want to add a Dynamic Property to (left click on the Static Object)
  - The available Dynamic Properties (see chart above) will be "lit up" in the Toolbar. Any remaining "grayed out" Dynamic Properties are not available for the Static Object.
  - Select one or more Dynamic Properties to associate with the selected Static Object. The selected Dynamic Properties will be highlighted. You can disassociate a Dynamic Property from a Static Object by clicking on the highlighted Dynamic Property icon in the Toolbar.
  - Double click on the Static Object and in the upper right corner of the Object Properties will be a drop-down Combo Box that provides access to the various Dynamic Properties settings that are associated with the Object. Set these according to the application requirements.
- More than one Dynamic Property can be added to a Static Object.
  - To add additional Dynamic Properties, click on the icon for the Dynamic Property in the Tool Bar
  - Any Dynamic Property associated with a Static Object will be illuminated when the Static Object is selected
- You can remove a Dynamic Property by clicking on the icon for the Dynamic Property (it must be illuminated (enabled) first).

Command Dynamic

Command Dynamic Tool

The **Command Dynamic** is one of the most useful and commonly used properties with Static Objects. Note that the Command Dynamic can be used with all the Static Objects that IWS provides and some of the Dynamic Objects. So what can be done with a Command Property? You can run a Script (IWS Script or VBScript code segment), you can Open or Close Screens, you can set a tag (to a value of 1), reset a tag(to a value of 0) or toggle a tag (toggle between 0 and 1).

To add a Command Dynamic to an object, select the object on the Screen, and then select the Command Dynamic tool icon. If the Command Dynamic tool is not "lit", it is not available for the object.



Command Dynamic Dialog Box

To access the Command Dynamic dialog box, do the following:

- Double click on the Static Object to access the Object Properties dialog box.
- Select **Command** from the drop-down Combo Box in the upper left corner of the Object Properties dialog box.

The Command Dynamic dialog box allows you to define the following parameters:

Replace: Launches a **Replace** dialog box (see end of this section for a discussion on the Replace function).

Hint: If the Hint box contains any text, the Hint will be displayed at runtime when the mouse cursor is over the underlying object

Event: Defines the action to be taken based on the Event Type. There are multiple event types defined by the Command Dynamic:

Event Type	Description
On Down	Left mouse button click (or defined key equivalent)
On While	While the left mouse button is held down (or defined key equivalent)
On Up	When the left mouse button is released (or defined key equivalent)
On Right Down	When the right mouse button is pressed
On Right Up	When the right mouse button is released
On Double Click	A double click of the left mouse button

A dialog box below the event is used to enter the function (or Script) to be executed when the Event occurs. The default dialog box is for entering VBScript code.

Key: Allows you to define a key used to trigger the Events On Down, On While and On Up.

Shift: If checked, you need to use either shift key (left or right) with the specified Key to trigger the Events, along with the Ctrl (if checked) and Alt (if checked) keys

Ctrl: If checked, you need to use either Ctrl key with the specified Key to trigger the Events, along with the Shift (if checked) and Alt (if checked) keys

Alt: If checked, you need to use either Alt key with the specified Key to trigger the Events, along with the Shift (if checked) and Ctrl (if checked) keys

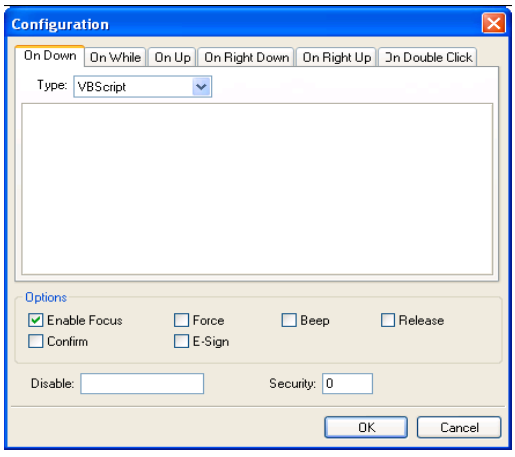
… This is the Key modifier, allowing you to specify a Key, the left, right or both Shift key, the left, right or both Ctrl key, and the left, right or both Alt key to trigger the Events.

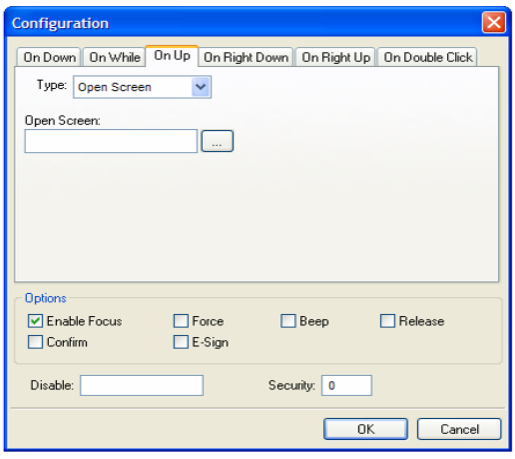
Notes:

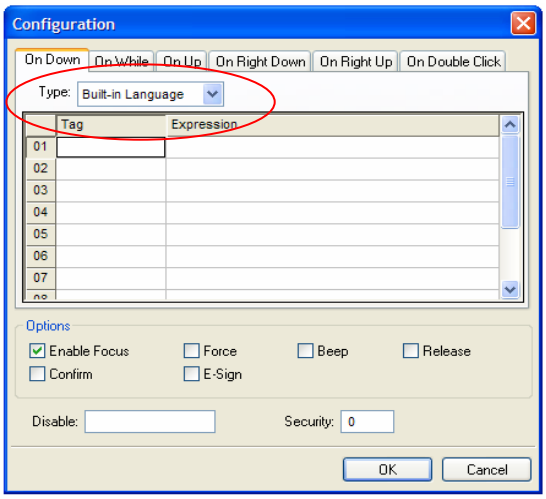
- The Hint field in the Object Properties provides a Tool Tip when the Object initially comes in focus. Any changes in the Hint field will change all the Hint fields for a given Object.
- The Key drop-down Combo Box, and the Shift, Alt & Ctrl Checkboxes allow you to define a set of keys that are the same as a left button mouse action (combination of On Down, On While, and On Up Events).
- The "…" button allows you to specify the Key plus the left, right or both combination of Shift, Ctrl & Alt keys that are the equivalent of a mouse click.
- You can define more than one Event (mouse or key sequences). E.g. you can run one VBScript code segment when the left mouse button is clicked (the On Down Event), and another when the left mouse button in released (the On Up Event).
- The Events On Right Down, On Right Up and On Double Click are not supported by CEView applications.
- Certain mouse actions (e.g. On Right Down, On Right Up) may not be available on a touch-screen device.
- The mouse action to trigger a Command Dynamic Event must take place with the mouse cursor placed on the object, while the Key has no requirement for this.
- **CAUTION: Do not use the same Key or combination of Keys to define an Event on more than one Static Object. Otherwise, both the Static Objects will execute their Command functions.**

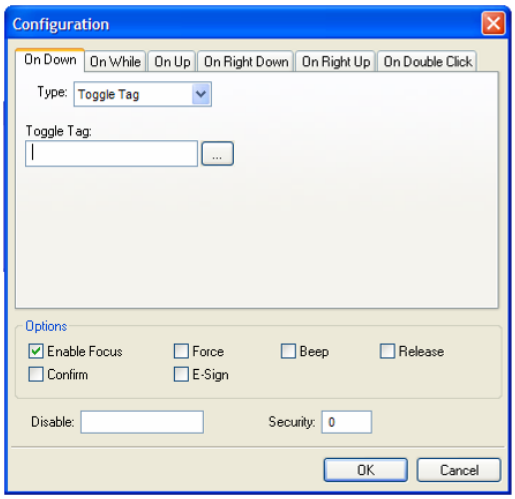
Config: Pressing the Config button in the Command Dynamic Properties dialog box launches the Command Dynamic Configuration screen. The scripting type or function to be performed is specified in the **Type** field.

Type	Description
Built-in Language	Uses the IWS built-in scripting language. You can use up to 12 expression lines, executed sequentially. The results are stored in the Tag field.
VBScript	Uses the VBScript scripting language
Open Screen	Opens a Screen. Allows you to specify the Screen to open. Can be an IWS string tag enclosed in curly braces { }
Close Screen	Closes a Screen. Allows you to specify the screen to close. Can be an IWS string tag enclosed in curly braces { }
Set Tag	Will set an IWS numeric tag to a value of 1
Reset Tag	Will set an IWS numeric tag to a value of 0
Toggle Tag	Will toggles an IWS numeric tag between 0 and 1









The Configuration dialog box has other parameters. They are:

Enable Focus When checked, the Static Object can receive focus during runtime by the navigation keys. This Option is normally checked.

Force When checked, a value will be written into the tag even if the tag currently has the correct value. For example, if the Tag value is 0 and the Command Dynamic is to reset the Tag (i.e. write a 0 to it), a 0 will be written to the Tag.
Note: This capability is useful when a Communication Driver is triggered to Write on Tag Change.

Beep When checked, a short beep is played when the Command is executed.

Release When checked, an **On Up** event is executed when an **On Down** event occurred, whether the button was released while in focus or not (e.g. button held down but mouse (or finger) moved out of focus before releasing).

Confirm When checked, User will have to confirm the Action before executing the Command.
Note: This feature is useful for preventing accidental triggering of an Action.

E-Sign When checked, User will be prompted to enter the Electronic Signature before executing the Dynamic.

Disable Disables the Command Dynamic when the result of the expression is **True** (i.e. <>0)

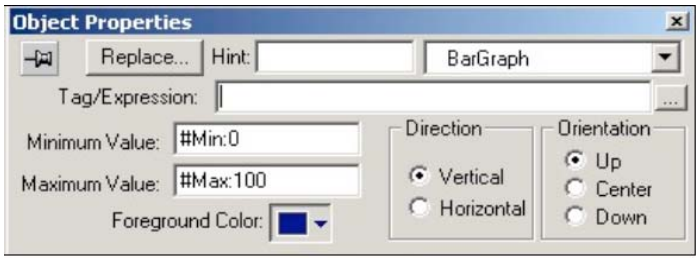
Security Defines the Security System Access Level required to execute the Command Dynamic.

Bar Graph Dynamic

The Bar Graph Dynamic Property is available for Ellipses, Rectangles (Rounded & Filled) and Closed Polygons. The Bar Graph is a simple tool that allows an Operator to quickly visualize a Tag's relative position between its Min and Max values. The Bar Graph Property can be added to any of the aforementioned Static Objects (in addition to other Dynamic Properties). The Bar Graph Property is added by selecting the Bar Graph icon in the Tool Bar.

Bar Graph icon

The Bar Graph Properties Dialog Box is shown at the right. The fields in this dialog box are:



Bar Graph Properties Dialog

Replace: Launches a **Replace** dialog box (see end of this section for a discussion on the Replace function).

Hint The **Hint** field allows you to enter a Tool Tip that shows up when the Object comes in focus.

Tag/Expression You can enter a tag or any valid expression consisting of Tags, operators, constants and IWS functions. (E.g. Mytag*5 +2, where Mytag is a numeric tag)

Minimum Value A numeric constant or tag that defines the minimum value of the Bar Graph.

Maximum Value A numeric constant or tag that defines the maximum value of the Bar Graph.

Direction & Orientation Horizontal or Vertical. Depending on the **Orientation** setting, the following applies There are three (3) directions available; Up, Down and Center. These values define the starting point for the **Minimum Value**. If Center is selected, then the starting point for Center is the middle of the Bar Graph and the end points (Top/Bottom or Left/Right) define the **Maximum Value**.

Orientation\Direction	Horizontal	Vertical
Up	Start: Left End: Right	Start: Bottom End: Top
Down	Start: Right End: Left	Start: Top End: End
Center	Start: Middle End: Left & Right	Start: Middle End: Top & Bottom

Foreground Color Specifies the color of the Bar

Notes:

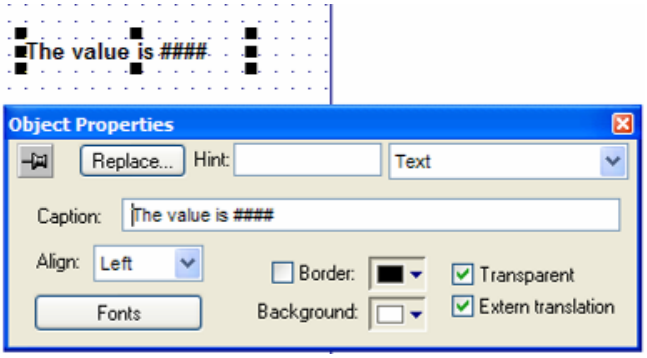
- If you change the Hint for any Dynamic Property, it will change it for all Dynamic Properties as well as the Static Object.
- If the Tag/Expression field evaluates to a value lower than the **Minimum Value** specified, the Bar Graph will display at the **Minimum Value**. If the Tag/Expression field evaluates to a value higher than the **Maximum Value** specified, the Bar Graph will display at the **Maximum Value**.
- If you selected the Bar Graph **Orientation** as **Up** and then rotate the Static Object, the **Minimum Value** will be on the left side of the rotated Static Object. If you selected the Bar Graph **Orientation** as **Down** and then rotate the Static Object, the **Minimum Value** will be on the right side of the rotated Static Object.
- In a Windows CE runtime environment, you cannot use the Bar Graph Property with a Rounded Rectangle Object.

Text I/O Dynamic

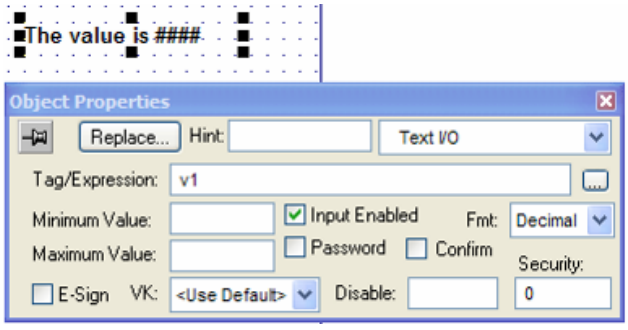
The Text I/O Dynamic Property can be used with a Static Text Object to provide numeric or alphanumeric display (e.g. tag values) and input data.

Text I/O icon

In order to use the Text I/O Dynamic, you must first put placeholder(s) in the Text object. The placeholder is the **#** character, and you need a placeholder for every alphanumeric character you want to display. Text can be placed on either side of the placeholder(s). As shown below in the example below, the Text object defines the text and placeholder(s) in the Caption field. The Caption is "**The Value is ####**". In this Caption, the text portion is "The Value is" and the placeholders are "####", providing output for 4 alphanumeric characters. In the Text I/O Dynamic, the IWS tag v1 is used to provide a value to the placeholders. The most significant four digits of the v1 tag value will be displayed.



Text Object

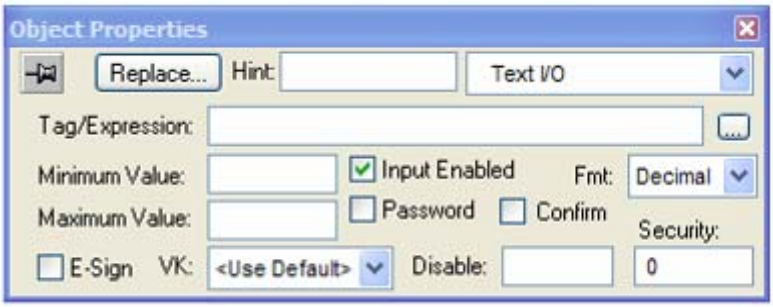


Text I/O Dynamic

Note:

- The placeholder format (in the Text Object Property Caption filed) is as follows:
  - **Boolean data:** Must include at least one # character (displays as a 0 or a 1)
  - **Integer data:** Must include one or more # characters, enough to display the range of the tag value
  - **Real data:** Follow rules for Integer if you do not want any decimal values, otherwise use ####.### in the Caption field, where the number of ### characters after the period signify the number or significant digits.
  - **String data:** Must include one or more # characters, enough to display the length of the string.

To add the Text I/O Dynamic Property, place a Text Object on a Screen, in the **Caption** field of the Text object define any text and the alphanumeric placeholder(s), then select the Text I/O Dynamic tool icon. The Object Properties will now include a dialog box for the Text I/O Dynamic Property as shown.



The parameters in the Text I/O dialog are as follows:

Replace: Launches a **Replace** dialog box (see end of this section for a discussion on the Replace function).

Hint The **Hint** field allows you to enter a Tool Tip that shows up when the Object comes in focus.

Tag/Expression Used to define an IWS tag or expression that will be displayed. The tag type can be a Numeric (Integer or Real), Boolean or String tag. Arrays or Class Members are treated based on the data type (e.g. Boolean, Integer, Real, String).

Notes:

- If the tag in the **Tag/Expression** field is:
  Boolean : Will be displayed as a 0 or 1
  Integer: Will be displayed as a whole number
  Note: If the value of the tag or expression exceeds the number of # placeholders in the Text Caption field, only the most significant digits will be displayed.
  Real: Will be displayed as a whole number or a real number, based on the placeholder format
  Note: If the value of the tag or expression exceeds the number of # placeholders in the Text Caption field, only the most significant digits will be displayed. The period can be used as a placeholder if the value of the tag or expression exceeds the number of placeholder digits. For example, if you used ###.## in the Caption Field and the tag value was 9999.99, a value of 9999 would be displayed.
  String: The ASCII value of each character in the string will be displayed, based on the placeholder format
  Note: If the number of characters in the String tag or expression exceeds the number of # placeholders in the Text Caption field, only the leftmost characters will be displayed.
- While the **Tag/Expression** field normally contains a Tag, you can enter an expression that uses the Tag as one of the elements of the expression. The result of the expression will be displayed in the Text I/O field, however the displayed value will not change the value of the Tag.
  Example: An integer tag named A is declared in the Application Tags Database. In the Text/Expression field, the expression A + 2 is placed in the field. The Input Enabled checkbox is checked. At runtime, when a value of 4 is entered into the Text I/O box, the tag A will have a value of 4 but a value of 6 will be displayed (the result of evaluating the expression A + 2). Note that the Tag and the display value in the Text I/O box are different.
  Note: Using the above example, if A were a string Tag and the expression was A+2, when a value C is entered into the text I/O box at runtime, the value of the tag A will "C2", not "E". This is not good programming practice. Instead, you should use the expression A + "2" to add the character 2 to the end. If you wanted to add the character D to the end of the string A, IWS will not allow you to do this unless the character D is enclosed in double quotes (i.e. "D"). Otherwise, if you attempt to put A + D in the Tag/Expression field, IWS will want to interpret this as adding a new tag D. If you do not define the new tag, IWS will treat this expression as a comment. If you had already defined a tag D, you might get an unexpected result.
- If more than one tag is used in the **Tag/Expression** field of a Text I/O Dynamic Property and the Input Enabled checkbox is checked, the value entered (even if an expression is used) will be stored in the left-most tag in the Expression as read from left to right.

Input Enabled If the Input Enabled checkbox is checked, the Text I/O Box can be used to input text and numeric values.

Minimum Value A numeric constant or tag that defines the minimum value for the numeric tag defined in the Tag/Expression field.

Maximum Value A numeric constant or tag that defines the maximum value for the numeric tag defined in the Tag/Expression field.

Notes:

- The Minimum Value & Maximum Value fields of the Text I/O Dynamic cannot be an expression. The fields must contain either a numeric constant or a numeric tag.
- Minimum and Maximum Values only have meaning for tags used in the Tag/Expression field that are numeric data types (Boolean, Integer, or Real).

  Note: If the Tag/Expression field has both Numeric and String data types, the Maximum and Minimum Values test will only be used only if the first tag is a Numeric data type. Otherwise, the Minimum and Maximum Values will not be used.
- If the User attempts to enter a value less than the Minimum Value or Greater than the Maximum value, it will not be accepted. The tag will retain its current value.
- Setting a Minimum Value or Maximum value in the Text I/O Dynamic only effects the data value being entered. Another object or Script can set the value the tag used in the Tex/Expression field outside of the range of the Minimum Value or Maximum Value of the Text I/O Dynamic.
- **The Minimum Value and Maximum Value** fields of the Text I/O Dynamic **ARE NOT** the same as the **Min** & **Max** properties set for the Tag in the Tags database. Be sure the Text I/O Dynamic Minimum and Maximum values are within the range specified for the Tag's Min & Max properties. Otherwise you can enter a number that will cause an error.

  Scenario 1
  A tag's Min & Max properties are set (in the Database View or by a Script) and the Text I/O Dynamic Minimum Value and Maximum Value fields are empty and a value is entered in the Text I/O Dynamic box at runtime is outside of the Tag's Min or Max values (as defined in the Tags's properties), an error message will be generated in the Output Windows and **the Tag value will be set to 0.**

  Scenario 2
  If either a) the Tag's Min property is > 0 or b) both the Tag's Min property & Tag Max property are < 0, then a value entered in the Text I/O box that is outside the valid range for the tag Min & Max properties will result in ?????? being displayed in the Text I/O field, the Tag Quality Property being set to Uncertain, and two error messages in the Output Window (1st message being that the value is outside the range (the value entered) and the 2nd message being that 0 is outside the range)

  Scenario 3
  Tag Min property = 0 ; Tag Max Property = 100
  Text I/O Dynamic Minimum Value = 10 ; Text I/O Dynamic Maximum Value = 20
  Result: Any values outside of 10 – 20 cannot be entered. No errors result

  Scenario 4
  Tag Min property = 10 ; Tag Max Property = 20
  Text I/O Dynamic Minimum Value = 0 ; Text I/O Dynamic Maximum Value = 100
  Result: Any value outside of 10 – 20 will result in a ?????? being displayed in the Text box at runtime. The Tag quality property will be set to Uncertain, and two error messages in the Output Window (1st message being that the value is outside the range (the value entered) and the 2nd message being that 0 is outside the range)

Password If the **Password** checkbox is checked, any values entered will be displayed with asterisks (*). The value entered will be stored in the tag specified in the **Tag/Expression** field.

Confirm If the **Confirm** checkbox is checked, any new values set during runtime requires confirmation. A pop-up dialog will ask the User to confirm.

Fmt The **Format** combo box specifies the format of the input/output field. Options are Decimal (the default/normal value), Hexadecimal and Binary.

Notes:
- For the Format combo box to be effective, all tags used in the Tag/Expression field must be Numeric data type.
- With Windows CE-based systems, Decimal is the only format available.

E-Sign When this option is checked, the user is prompted to enter the Electronic Signature before changing the tag value

VK Specifies the Virtual Keyboard type to be used for input with the Text I/O Dynamic. The options are:
- Numeric
- Enhanced
- AlphaNumeric
- Default

Note:
- To use the Virtual Keyboard, you need to enable the Virtual Keyboard option for the Project. Go to **Project → Settings → Runtime Desktop** and enable the Virtual Keyboard and set the default VK style.

Disable This field can be a tag or expression. When the tag or expression evaluates to 0, the data input property for the Text I/O Dynamic Property will be enabled (assuming the Input Enabled checkbox is checked). If it evaluates to a value other than 0, it will disable data input.

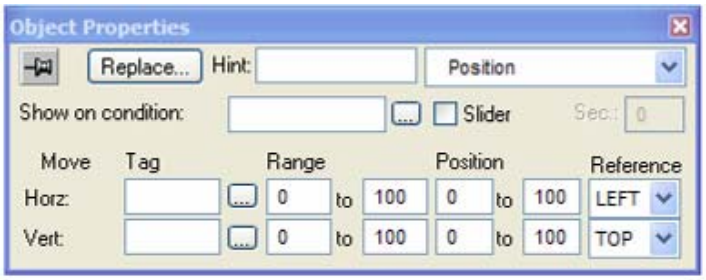
Security This is a numeric value (tags or expressions not allowed) that defines the security level required for data input to be enabled (provided the Input Enabled checkbox is checked).

Position Dynamic

The Position Dynamic property allows you to specify a horizontal and vertical offset for an object, as well determine an object's visibility

To add a Position Dynamic property to an object, select the object on the Screen, and then select the Position Dynamic tool icon. If the Position Dynamic tool is not "lit", it is not available for the object you selected.

By double-clicking on the selected object, you will open the Object Properties dialog box. When the Position Dynamic Property is added, the Position Dynamic Property dialog box becomes available from the combo box drop down list (upper right corner).



The following parameters can be specified:

Replace: Launches a **Replace** dialog box (see end of this section for a discussion on the Replace function).

Hint The **Hint** field allows you to enter a Tool Tip that shows up when the object comes in focus.

Show on Condition This is an IWS tag or expression. If the evaluated value is <=0, then the object will be hidden (not visible). If the evaluated value is >= 1, the object is visible.

Slider This is a check box that if checked, the object will act like a slider bar. This means that you can drag the object (up to the limits specified Position fields). The dragged object will update the tag values specified in the Tag fields.

Sec. A Security field. If used, requires a positive integer to be entered corresponding to the security level of the operator permitted to use the object.

Tag Specifies the IWS tags used to define the horizontal and vertical movement.

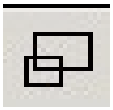
Range Defines a Min and Max value for the IWS tag values (horizontal & vertical) specified in Tag fields.

Position Determines the change in position (pixels) in the horizontal and vertical direction that the object can take. These values can be negative. The position offset (from Min to Max) will be scaled by the Range (Min to Max). For example, if the Position was 0 to 100 (for horizontal and for vertical) and the Range was 0 to 10, for each increment in the tag value (horizontal and vertical) between 0 and 10, the object would move 10 pixels.

Reference A combo box that lets you select the reference point for the object. Values are:
Horizontal: Left, Center, Right
Vertical: Top, Center, Bottom

Note:
- The Position Dynamic Property is useful to control the visibility of an object, and can be used to simulate movement.

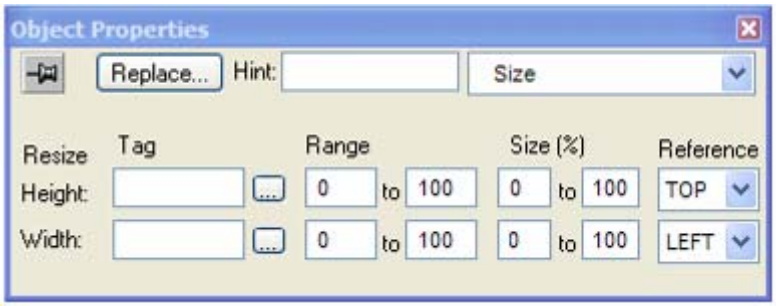
Resize Dynamic



The Resize Dynamic property allows you to increase or decrease the size of a selected object or symbol.

To add a Resize Dynamic property to an object, select the object on the Screen, and then select the Resize Dynamic tool icon. If the Resize Dynamic tool is not "lit", it is not available for the object you selected.

By double-clicking on the selected object, you will open the Object Properties dialog box. When the Resize Dynamic Property is added, the Resize Dynamic Property dialog box becomes available from the combo box drop down list (upper right corner).



The following parameters can be specified:

Replace: Launches a **Replace** dialog box (see end of this section for a discussion on the Replace function).

Hint The **Hint** field allows you to enter a Tool Tip that shows up when the object comes in focus.

Tag Specifies the IWS tags used to define the height and width scaling on the object.

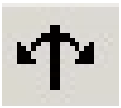
Range Defines a Min and Max value for the IWS tag values (height & width) specified in Tag fields.

Size % Determines the change in size of the object as a percentage (%) of its original value. These values can be negative. The Size % (from Min to Max) will be scaled by the Range (Min to Max). For example, if the Size% was 0 to 100 (for height and for width) and the Range was 0 to 10, for each increment in the tag value (height and width) between 0 and 10, the object would scaled by 10%.

Reference A combo box that lets you select the reference point for the object. Values are:
Height: Top, Center, Bottom
Width: Left, Center, Right

Note:
- By setting the Height & Width values to 0, the object is no longer visible. The effect is similar to controlling the visibility using the **Show Condition** in the Position Dynamic.

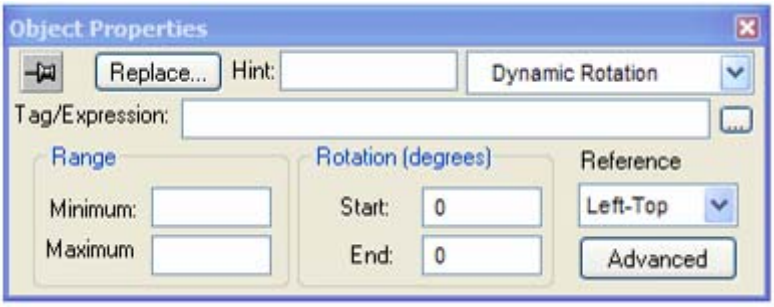
Dynamic Rotation Dynamic



The Dynamic Rotation Dynamic property allows you to increase or decrease the size of a selected object or symbol.

To add a Dynamic Rotation Dynamic property to an object, select the object on the Screen, and then select the Dynamic Rotation Dynamic tool icon. If the Dynamic Rotation Dynamic tool is not “lit”, it is not available for the object you selected.

By double-clicking on the selected object, you will open the Object Properties dialog box. When the Dynamic Rotation Dynamic Property is added, the Dynamic Rotation Dynamic Property dialog box becomes available from the combo box drop down list (upper right corner).



The following parameters can be specified:

Replace: Launches a **Replace** dialog box (see end of this section for a discussion on the Replace function).

Hint The **Hint** field allows you to enter a Tool Tip that shows up when the object comes in focus.

Tag/Expression A tag or expression to determine the rotation (in degrees) of the specified object

Range Defines a Min and Max value for the IWS tag values specified in Tag fields.

Rotation Determines the starting and ending rotation (in degrees) for the object. These values can be negative. The Rotation (from Start to End) will be scaled by the Range (Min to Max). .

Reference A combo box that lets you select a reference point (point of rotation) from which to rotate the object through the screen. Options are:

Left-Top	The upper left most point of the object
Left-Bottom	The lower left most point of the object
Center	The center (horizontal & vertical centers) of the object
Right-Top	The upper right most point of the object
Right-Bottom	The lower right most point of the object

Advanced Lets you define the rotation orientation (clockwise or counterclockwise), and an X & Y offset to the rotation reference point.

Note:

- Previous to IWS Version 6.1 SP2, there were only three reference points; Left-Top, Center and Right-Bottom. The same reference points between versions do not necessarily operate the same.
- For a Line object, the Left-Top and the Right-Top, and the Left-Bottom and Right-Bottom reference points are the same
- Window CE-based applications cannot rotate an object more than 90º. All other OS platforms can rotate an object a full 360 º.

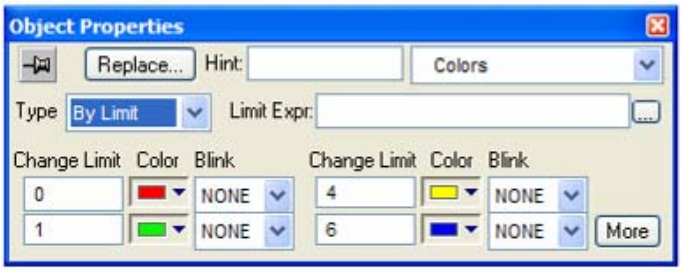
Colors Dynamic

Colors icon

The Colors Dynamic Property applies to all Static Objects and certain Dynamic Objects. This Dynamic Property allows you to change the color of the Static Object during runtime based on the evaluated value in the Tag/Expression field.

To add a Colors Dynamic property to an object, select the object on the Screen, and then select the Colors Dynamic tool icon. If the Colors Dynamic tool is not "lit", it is not available for the object you selected.

By double-clicking on the selected object, you will open the Object Properties dialog box. When the Colors Dynamic Property is added, the Colors Dynamic Property dialog box becomes available from the combo box drop down list (upper right corner).



Colors Property Dialog Box

The following options can be specified:

Replace: Launches a **Replace** dialog box (see end of this section for a discussion on the Replace function).

Hint The **Hint** field allows you to enter a Tool Tip that shows up when the Object comes in focus.

Type The **Type** field determines the mode in which the Colors Dynamic works, and can have the following types:

Type	Description
By Limit	The **By Limit** mode allows you to specify up to 16 Change Limit values, and a color for each Change Limit. In this mode, the Limit Expression field is evaluated and the color of the object is determined by the Change Limits specified
By Color	The **By Color** mode allows you to specify (in the Color Expression field).the color that must be applied to the object. With the **By Color** mode, there are no Change Limits.

Note:

- In the **By Limit** mode, the color will be determined by the evaluated value of the Limit Expression field and the Change Limit. If there is no Change Limit that corresponds to the evaluated value, then the first lower Change Limit is used. For example, if there are Change Limits of 0 (Color Red), 1 (Color Green) and 4 (Color Blue), an evaluated value of 2 would set the Color to Green. An evaluated value of 5 would set the Color to Blue.

Limit Expression This field is for an IWS tag or expression. The **Type** field will determine its use:

Type	Tag/Expression Field
By Limit	When the **By Limit** mode is used, the Limit Expression field will be evaluated and the result compared with the Change Limits specified to determine the Color of the object.
By Color	When the **By Color** mode is used, the Color Expression field defines the Color of the object.

Note:
- When using the **Type By Color** mode, you can use the IWS built-in function **RGBColor()** in the Color Expression field. This will allow you to specify the color in an R,G,B format

Change Limit The Change Limit is a numeric constant or tag that is used with the **Type By Color** mode to define the object Color based on the evaluated value contained in the Limit Expression field.

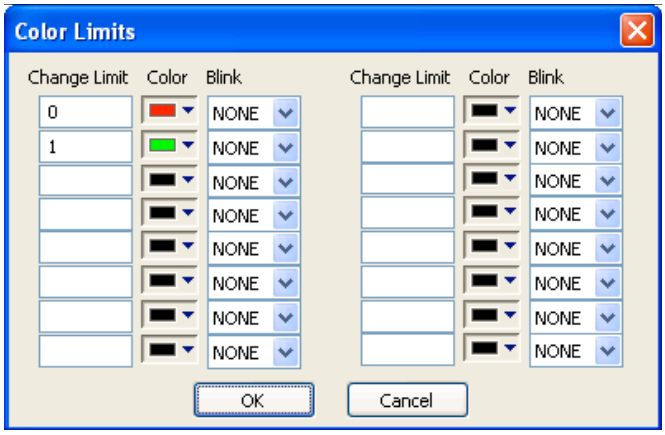
Important Note:
- The Change Limits must be entered in ascending order, starting with the lowest number in the upper left corner, down to the lower left, the upper right and down to the lower right.

Color Used to specify the Color for a Change Limit. Note that you can specify many different colors and fill effects.

Blink Allows you to specify whether the color is to blink or not. If blink is selected, the rate can be selected (fast or slow)

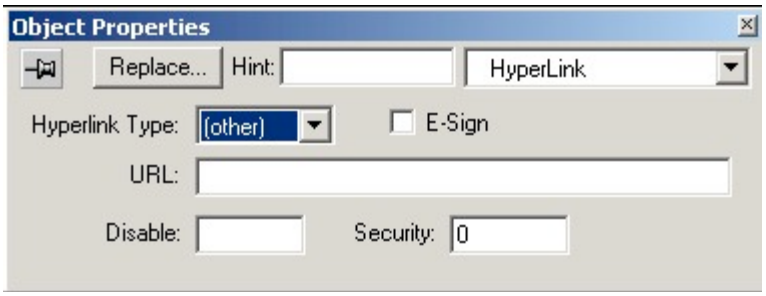
Note:
- The blink rates (BlinkFast and BlinkSlow) are system parameters set in the file **Program Settings.ini**, located in the \Bin subdirectory where the IWS Program is installed (e.g. usually c:\Program Files\InduSoft Web Studio v6.1\Bin).

More Opens a new dialog box that allows you to enter up to 16 Change Limit values (plus colors and blink settings)



Hyperlink Dynamic

The Hyperlink Dynamic Property can be added to a selected object or a group of objects. By adding this property, you can click on the object during execution and launch the default browser and load the URL specified.



The Configuration dialog box has other options available. They are:

Replace: Launches a **Replace** dialog box (see end of this section for a discussion on the Replace function).

Hint The **Hint** field allows you to enter a Tool Tip that shows up when the Object comes in focus.

HyperLink Type A combo box lets you select a URL type from the list. IWS use the appropriate protocol when it loads the URL. The types include:
- file:
- ftp:
- gopher:
- http:
- https:
- mailto:
- news:
- telnet:
- wais:

E-Sign When this option is checked, the user is prompted to enter the Electronic Signature before executing the Dynamic Property

URL Enter the URL address you want to access. The protocol type is not required to be entered in the URL field as long as it is in the Hyperlink Type list.

Disable Enter a IWS tag or expression in this field. If the value evaluates to <>0, then the Hyperlink function is disabled. If the value evaluates to 0, the Hyperlink function is enabled.

Security Enter a positive integer in this field to specify the security level required to access this object. If the current user does not have the required security level to access the object, the Hyperlink property for the object is disabled.

Mode Toolbar

The Mode Toolbar provide tools for general Screen editing. By default, this toolbar is on the right side of the screen with the Static Object, Dynamic Object and Dynamic Properties Toolbars.

Selection The **Selection** tool is used to select or move objects on the Screen. If no other tools are selected, IWS will default back to the **Selection** tool.

Bitmap Editor The **Bitmap Editor** will switch between the **Objects** editing layer (where Objects are placed) and the **Background Picture** layer (for static bitmaps). When the **Background Picture** layer becomes active, a **Bitmap** Toolbar is displayed.

Note:
- To enable the **Background Picture** layer, the **Enable Background** checkbox in the Screen Attributes dialog must be checked.

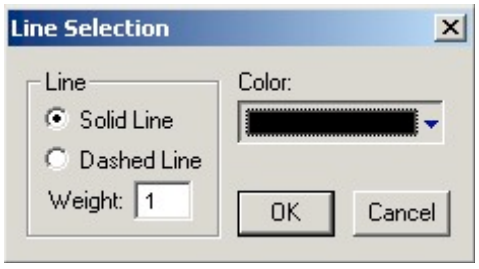
Fill Color The **Fill Color** tool allows you to specify a default fill color for the following objects:
- Closed Polygons
- Ellipses
- Rounded Rectangles
- Rectangles

Fonts The **Fonts** tool allows you to select the font, style, size, color and effects for new Text objects or for selected objects.

Note:
- The **Fonts** tool cannot be used with Grouped Text objects. Instead, you can ungroup the objects, apply the Fonts tool, then regroup the objects.

Line Color The **Line Color** tool can be used to specify the default line color for new Objects, or for selected objects. These objects include:
- Open Polygons
- Closed Polygons
- Lines
- Ellipses
- Rounded Rectangles
- Rectangles

The **Line Color** tool uses a **Line Selection** dialog box to allow you to specify the line style, weight and color.



Background Color The **Background Color** tool can be used to specify a background color for the Screen.

Grid The **Grid** tool can be used to show or hid the Grid on the Screen

Note:
- The **Grid Settings** dialog (right click on a blank portion of a Screen) can be used to set default Grid settings.

Bitmap Toolbar

The **Bitmap Toolbar** provides a set of tools to edit bitmaps placed on the **Background Picture** layer.

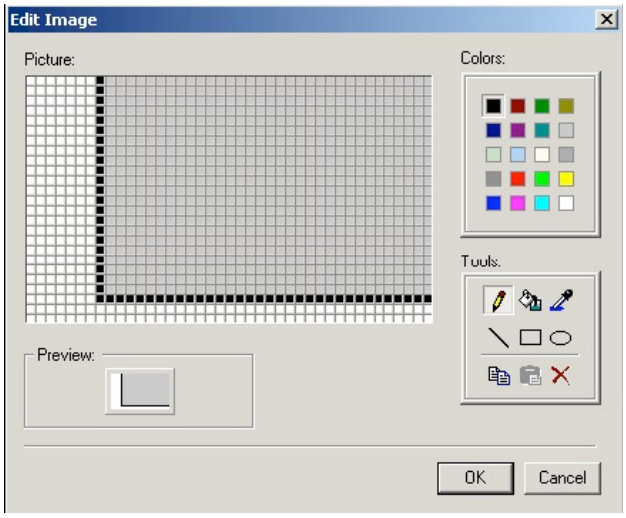
Note:

- The **Background Picture** layer can be enabled in the **Screen Attributes** dialog, accessed by pressing the right mouse button when the mouse cursor is in a blank portion of the Screen. Select **Screen Attributes** from the list of options. The Screen Attributes dialog also appears when a Screen is first created. Select the **Enable Background** checkbox.
- To access the **Background Picture** layer, select the **Bitmap Editor** tool in the Mode Toolbar

Select Area The **Select** tool is used to select an area within the Bitmap Screen Editor for further processing.

Flood Fill The **Flood Fill** tool is used to paint the surrounding area with the color you specified in the **Fill Color** tool.

Pixel Edit This is the **Edit Image** tool used to manually draw or manipulate bitmap objects, pixel by pixel.



Erase Area The **Erase Area** tool is used to remove the selected area from the Screen.

Change Colors The **Change Colors** tool is used to change the transparent fill color for a selected area.

Note:

- Before using this tool, specify a fill color using the **Fill Color** tool, select the transparent color using the **Select Transparent Color** tool, and define the area to fill using the **Select Area** tool

Select Transparent Color	This tool specifies a transparent color that is referenced by the **Change Colors** tool.	
Toggle Transparent Color	This tool will cause the color selected in the **Select Transparent Color** tool to become transparent for bitmapped objects in the Bitmap Editor.	

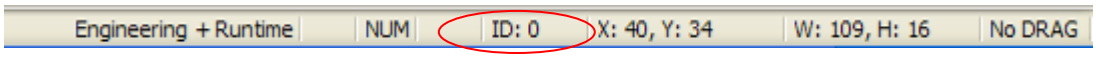
Note:

- To enable the **Background Picture** layer, the **Enable Background** checkbox in the Screen Attributes dialog must be checked.
- You can use Copy (**Ctrl C**) and Paste (**Ctrl V**) functions to exchange bitmap objects between the IWS Bitmap Editor and other programs (e.g. Microsoft Paint Brush)

Using Object IDs

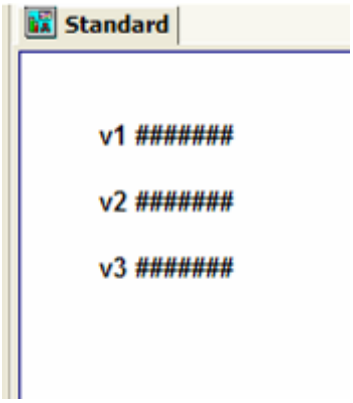
Application Screens commonly use several adjacent Text Objects (with the Text I/O Dynamic Property) for input of data. To navigate these Objects, you can use the mouse, positioning the cursor over the Object and clicking the left mouse button. But if your application has a keyboard, there is another alternative for navigating through the Text Object by use of the **Tab** key.

Every Object placed on an application Screen has an ID number. ID numbers are unique to every application Screen and always start with 0. By default, the first object placed on the application Screen has an ID = 0, the next object will be ID = 1, etc. When you select an object (i.e. place the mouse cursor over the object and click the left mouse cursor), the ID will be shown in the Status Bar located at the bottom of the IWS development screen.



When the application is placed in a runtime mode and an application Screen is first opened, the focus will be always be on the first object (i.e. the object with ID = 0). If you hit the **Tab** key on the keyboard, the focus will shift to the next object (i.e. **ID = 1**), and so on.

So for example, we could create an application Screen that has three (3) Text Objects with Text I/O Dynamic Properties with input enabled (as shown in the Screen to the right). During runtime, you might want the initial focus to be on the **v1** data field. If you enter a numeric value, it would set the value for the IWS tag **v1**. If you press the **Tab** key, you would set the focus to the **v2** data field, etc.



The problem is that the Object ID is set, by default, based on the sequence of the placement of the objects on the application Screen. For example, if the **v1** Text Object was the first object placed on the application Screen, it's ID would be equal to 0. But if it was not the first object placed on the Screen, it would have a different ID.

You can change the Object ID of an object placed on an application Screen in one of two ways:

c) Select an Object on the application Screen. Click on the **Move Forward** or **Move Backward** icons in the Toolbar area to lower or raise its Object ID number. Note that you can also use the **Move To Front** and **Move To Back** tools, but these every time you move an Object to the front or back, it will become the first object or the last object. It is better to use the Move Forward and Move Backward tools.

d) Select an Object on the application Screen. With a standard keyboard (desktop PC) or a USB keypad (for a Notebook), hold the **Ctrl** key down and press the **+** or **–** key (in the numeric keypad) to raise or lower the Object ID number.

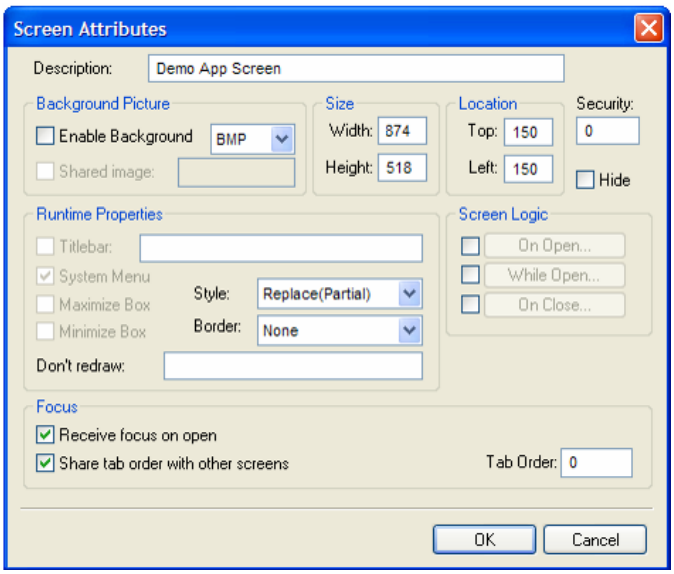
In the example above, we would want the **v1** Text Object set to ID = 0, **v2** Text Object set to ID = 1, and **v3** Text Object set to ID = 2. When we run the application, the initial focus will be on the **v1** value, and when the **Tab** key is pressed, it will move the focus to the **v2** value, etc.

Exercise: Using Static Objects with Dynamic Properties

Objective: In the previous exercises, we first created a new Project, then created some MDI screens and on one of the MDI Child Window screens (**Main.scr)**, we placed several Static Objects. In this exercise we will place Static Objects with Dynamic Properties on MDI screens to build our first working application.

Create additional MDI Screens

We will first create several new application Screens. These will be MDI Child Windows, and will be the same size as the **Main.scr** screen.

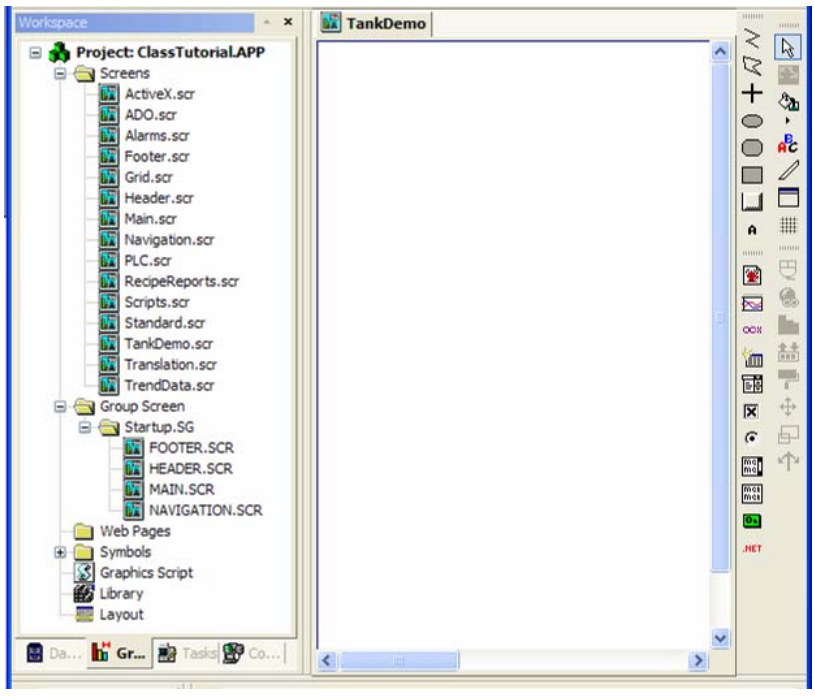


- ☐ Open the IWS development environment and open the **ClassTutorial** Project.
- ☐ Open the **Graphics Workspace**.
- ☐ Place your mouse cursor on the **Screens** folder, click the right mouse button, and select **Insert.**
- ☐ In the **Screen Attributes** dialog box, set the **Description** box to **Demo App Screen**, set the Size & Location parameters to :

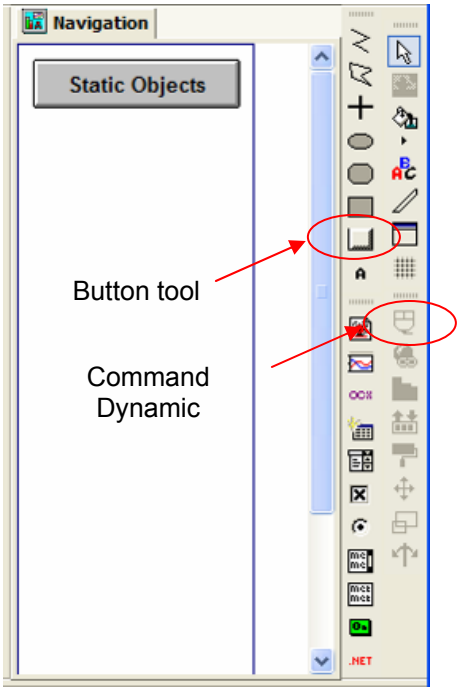
  Width = 874 Height = 518

  Top = 150 Left = 150
- ☐ Click the **OK** button
- ☐ In the Main Menu Bar, click on **File → Save As**
- ☐ In the **File name** box in the **Save As** dialog box, type **TankDemo** and press the **Save** button**.**
- ☐ Repeat the last 2 steps (i.e. click **File → Save As**, enter a file name, and press **OK**) using the following file names:

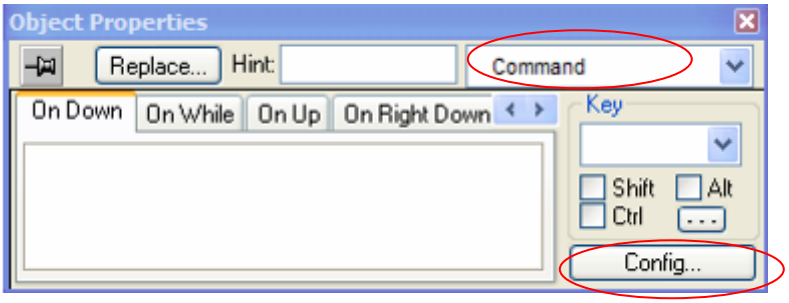
  ActiveX
  ADO
  Alarms
  Grid
  PLC
  RecipesReports
  Scripts
  Translation
  TrendData
- ☐ When finished, you should have the following Screens as shown.



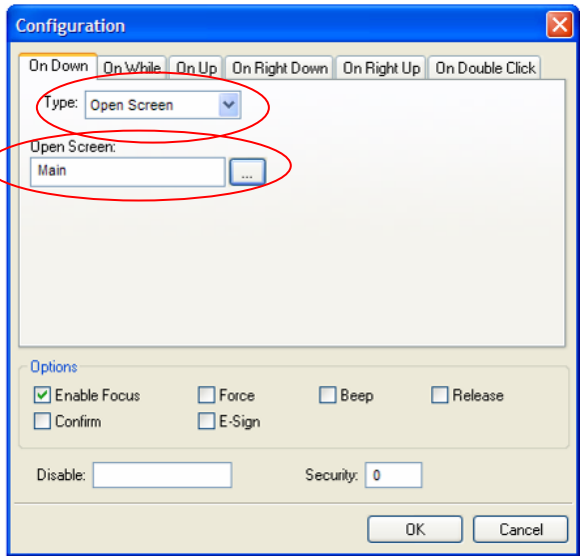
Add Navigation Buttons to the Navigation.scr Screen



- ☐ Open the **Navigation.scr** screen by double-clicking on the Screen name (or icon) in the **Screens** folder in the **Graphics Workspace**.
- ☐ Add a Button object to the **Navigation** Screen by selecting the Button tool in the Static Objects toolbar, putting your mouse cursor in the Navigation Screen, and while holding the left mouse button down, drag the mouse cursor to create the Button object. Release the left mouse button.
- ☐ If necessary, resize the Button object by selecting it (left mouse click while mouse cursor is positioned over the object), and then click and drag on the object handles. The Button object should be approximately 130 pixels wide by 30 pixels high. The **Status Bar,** located in the lower right corner of the IWS Development interface, will indicate the size (H) & (W) of the object.
- ☐ With the Button object still selected, add a **Command Dynamic** to it.
- ☐ Double-click on the Button object to get to its Object Properties.
- ☐ In the **Caption** field, type **Static Objects**.
- ☐ Select the **Command** Dynamic from the pull-down combo box. Click on the **Config…** button

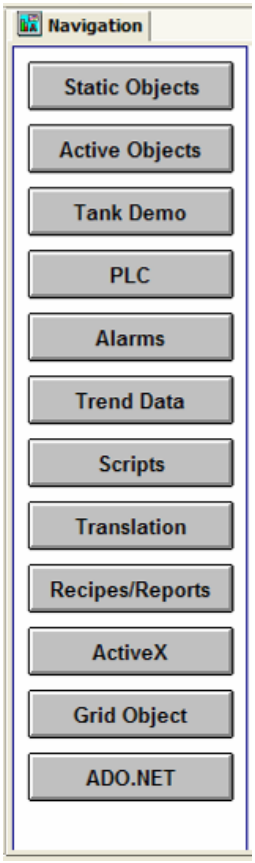


In the Configuration dialog box, choose **Type** as **Open Screen.** In the Open Screen box, click on the **…** button and select **Main** (Main.scr). Click **OK**. Close the Object Properties dialog box.



☐ Add other button objects as shown.

Button Caption	Screen to Open (Command Dynamic)
Active Objects	Standard.scr
Tank Demo	TankDemo.scr
PLC	PLC.scr
Alarms	Alarms.scr
Trend Data	TrendData.scr
Scripts	Scripts.scr
Translation	Translation.scr
Recipe/Reports	RecipeReports.scr
ActiveX	ActiveX.scr
Grid Object	Grid.scr
ADO.NET	ADO.scr



☐ Align and vertically space the Buttons using the IWS alignment tools (see the "Align and Distribute Toolbar" section).

Add a Navigation Button to the Standard.scr Screen.
When we initially created the Screens (in a previous exercise), we created one Screen that was a full-size (non-MDI) screen called Standard.scr. We will use this Screen to create a demo screen using Dynamic Objects. However, since this screen is a full-size screen, it will overwrite the Navigation Screen (Navigation.scr). So we need to have a way to return (i.e. navigate) to the startup Screen group (Startup.sg). We will do this by adding a Button Object in the Standard.scr Screen, with a Command Property that opens up the Screen **Startup.sg**.

☐ Open the Screen **Standard.scr**

☐ Select the Button tool from the Static Object Toolbar

☐ Click and drag in the Standard.scr screen, creating a button. Place in the lower left corner of the Screen.

☐ With the Button object selected, add a Command Dynamic by clicking on the **Command Dynamic** tool in the Dynamic Properties Toolbar.

☐ Double click the Button Object to open the Object Properties.

☐ In the Caption Field of the Button Object Properties, type **Startup**

☐ Open the **Command** Dynamics Object Properties by selecting Command from the combo box in the upper right corner of the Object Properties dialog box.

☐ Click on the **Config…** button

☐ In the Configuration dialog box, choose **Type** as **Open Screen.** In the Open Screen box, click on the **…** button and select **Startup** (Startup.sg). [**Note: you have to select *.sg File type in the Open dialog box**]. Click **OK**. Close the Object Properties dialog box

Testing it out.

☐ Select from the Main Menu Bar **File → Save All** to be sure all Screens are saved. Next, close all the Screens by using selecting **File → Close All** from the Main Menu Bar.

☐ Run the application by selecting the **Run Application** button.

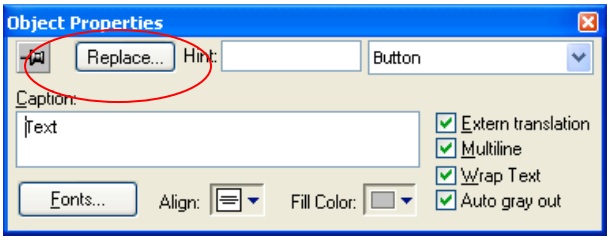
☐ Checkout all navigation buttons to make sure they function correctly.

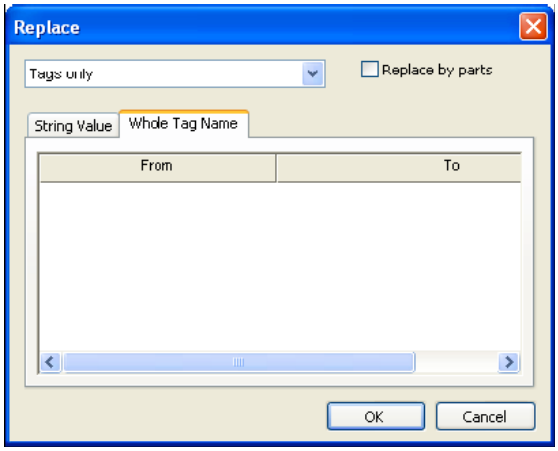
☐ Click on the Close Application box in the upper right corner of the runtime Screen.

☐ Select the **Stop Application** button.

Replace Function & Custom Properties

In several of the objects we have seen so far, there has been a **Replace** button that we have not discussed yet. The Replace button is located in the Object Properties dialog box for each of the objects that we have covered. However, since we had not used any tags or strings with the objects (except for one Button object), we have not covered the Replace function until now. The Replace function provides a fast and simple tool to replace Strings, Tags or Properties for a single object or a group of objects (Grouped Object).



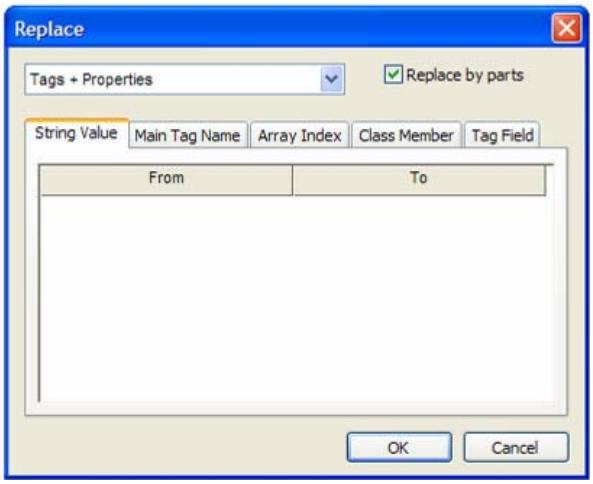


When the **Replace** button is pressed, a **Replace** dialog box is launched. The Replace dialog provides three main interface sections:

Drop Down Combo Box
The drop-down combo box allows you to select (filter) the type of information that can be replaced. The filtered information shows up when the Whole Tag Name tab is selected in Grid area below. The choices are:

- **Tags Only**
  Displays only tags which are configured directly (i.e. Application Tags and Internal Tags)
- **Tags + Properties**
  Displays both the Tags and Properties configured as Mnemonics configured as Custom Properties
- **Custom Properties**
  Displays only the Custom Properties

Replace by Parts Checkbox
The Replace by Parts checkbox is frequently used when you have a Group of Objects or a Linked Symbol. When this checkbox is checked, you can replace portions of a group of tag separately; i.e. the main tag name, the array index, a class member or a tag field. When the Replace by Parts checkbox is selected, a different From/To Grid and Tabs section of the dialog box appears.



From/To Grid with Tabs
The Grid is used to define the existing Tag Name or String Value (in the **From** column) and the replacement for the Tag Name or String Value (**To** column). This Grid is populated with all matches that meet the search criteria specified above.

Notes:
- Each Object and Dynamic Property has a **Replace** button in its Object Properties dialog box.
- The drop-down combo box defines what information shows under the **Whole Tag Name** tab in the Grid section below.
- If you select the **Replace** button from a Dynamic Property Object Properties dialog box, you will only see the tags, custom properties and strings for the specific Dynamic Property selected. Any replacements made will only effect the specific Dynamic Property. If you want to change the tag name. custom property or string value for the object, select the **Replace** button at the Object level.
- If you select a Group of Objects and then select the **Replace** button from the Object Properties dialog box, all of the Objects and Dynamic Properties can be effected by the Replace function.

Custom Properties

Custom Properties are frequently used with generic objects (e.g. Library Symbols) when the tag name, expression, or value is not determined until development occurs. So instead of creating objects with phony tag names or sting values, and later replacing the tag name (which initially could result in an undefined tag error) or string values, we can instead define a Custom Property for the tag name, expression or value and then use the **Replace** function when the object is inserted on the screen (or anytime before running the application).

The format of a custom property is as follows:

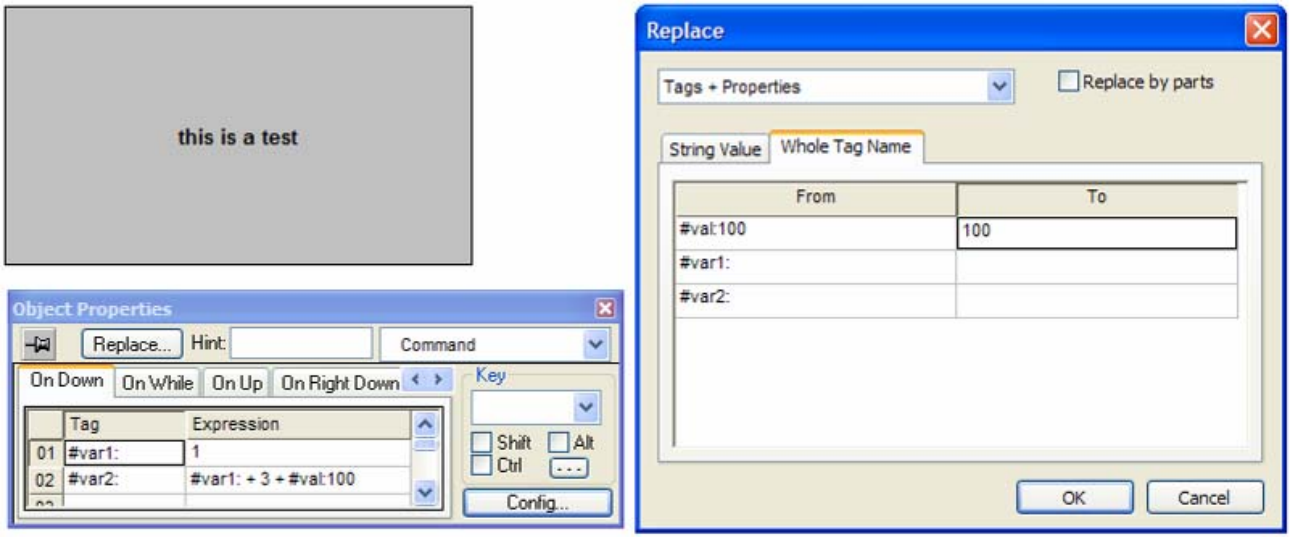
#<Label>:<Default>

where: **#** symbol used to indicate a custom property
<Label> is a name used to identify they property
: a delimiter for the optional value
<Default> us an optional default value for the property

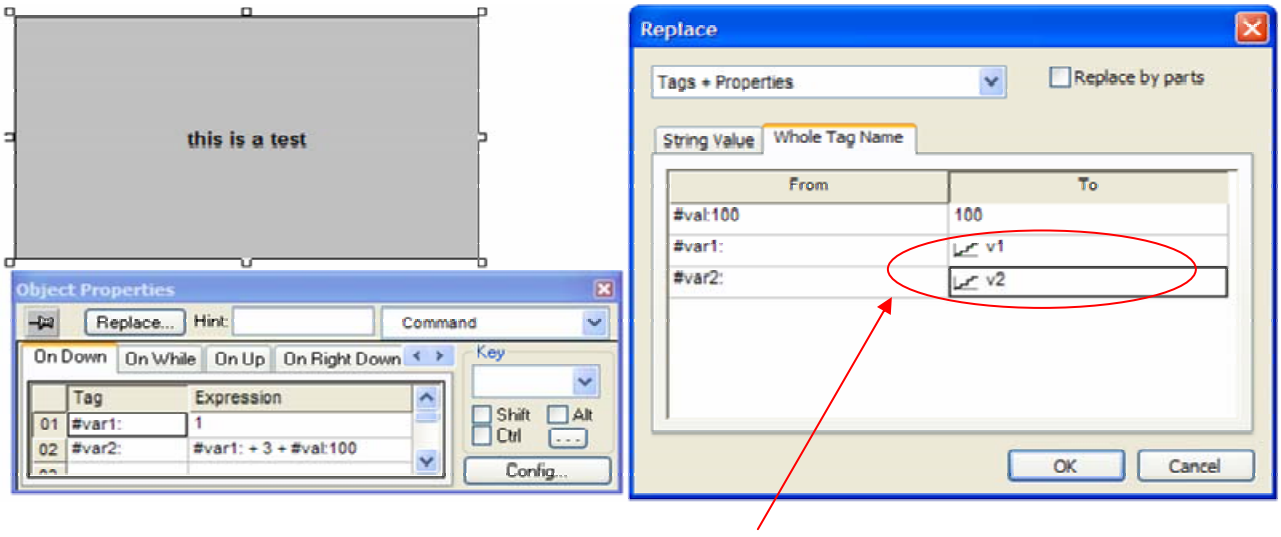
The Custom Properties can be used with Constants, IWS Tag Names, Indirect Tags, Array Tags, Boolean Tags, Numeric Tags (Integer & Real type), String Tags, and Scripting expressions. Custom Properties work for IWS Script as well as VBScript.

Notes:
- To define a Custom Property used as a Constant, replace the Numeric value with **#Label:**, where **Label** is any valid name (using alphanumeric characters). It is best not to use the name of an existing tag so as to prevent confusion. You can define an initial value for the Constant by using the following syntax **#<Label>: <Default>**, e.g. #StartPoint:100.
- The Constant name (e.g. **Label**) is not added to the application tags database.
- You should use the <Default Value> delimiter (**:**) even if there is no default value used

To give an example, we will create a simple button object (as shown below) to which a Command Dynamic Property is added. In the Command Dynamic, a Script is added (as shown). The Replace Dialog (after pressing the **Replace** button is also shown.

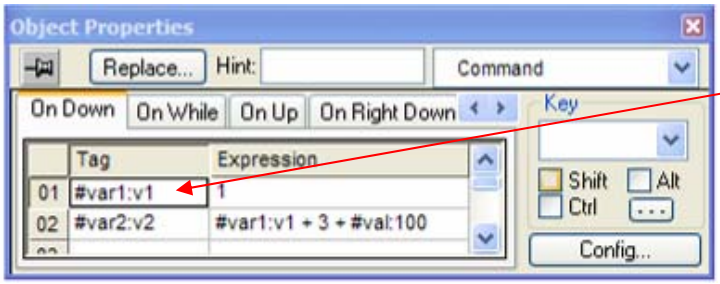


As it currently stands, although the IWS application will execute, nothing is saved and the expressions are not processed. We need to replace the Custom Properties **#var1:** and **#var2:** with real tag names since the result of an expression needs to be stored in a tag. We will replace them with tags **v1** and **v2,** respectively. These tags were already defined in the Application Tags Database, otherwise IWS would open a dialog box asking you to define them.



Enter tag names **v1** and **v2** into the **To** field. Press **OK**.

Once you have pressed the **OK** button in the **Replace** dialog box, notice that the Tag names that were entered in the **To** field of the **Replace** dialog box have been added to the Global Properties. Now, when the Screen which the Button object is placed on executes and the Button object is pressed, tag **v1** will be set to a value of 1, and tag **v2** will be set to a value of 104.

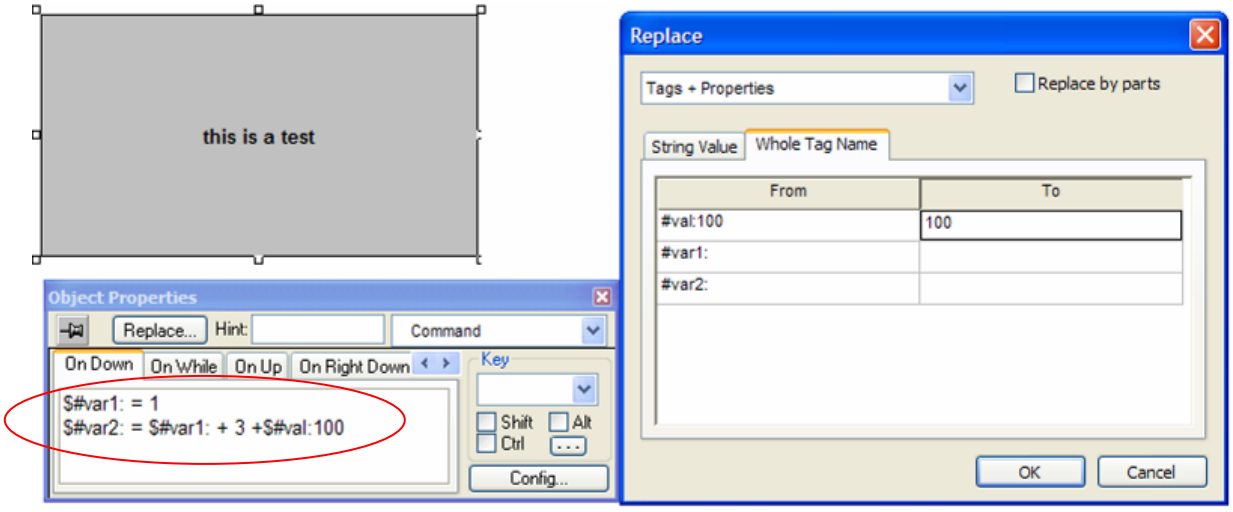


Tag names specified get entered into **<Default>** field.

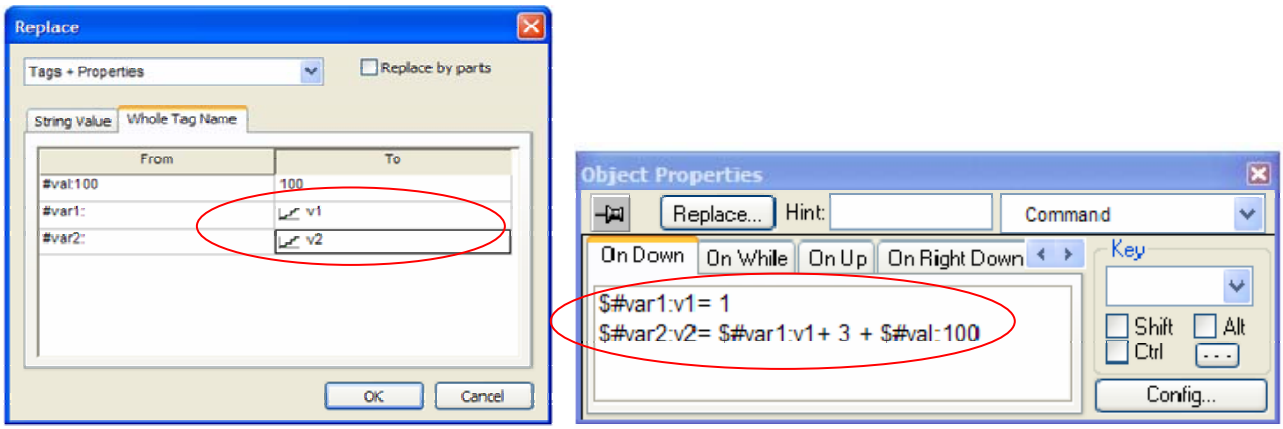
Notes:

- In some cases, an Object may require the use of a specific type of value; e.g. a String, Boolean, or Numeric value. To use the Custom Property in this manner, you must enclose the Custom Property in curly braces **{ }**. For example, the Caption field of a Rectangle can use a String to display some Text during runtime. To use a String with a Custom Property in a Caption field, the syntax would be:
  {#<Label>:"<DefaultString>"}
  If the Custom Property is used in the Caption field, you would need to press the Object's **Replace** button in order to replace the default string.

As mentioned, Custom Properties also work with VBScript. VBScript is a programming alternative to the spreadsheet oriented format of the IWS Scripting Language. The **$** operator is used to signify to the VBScript engine that a IWS Tag, Property or Function is being referenced.



The equivalent expressions are added in the VBScript area of the Object Properties dialog box for the Command Dynamic. When the **Replace** button is pressed, a **Replace** dialog box appears that is similar to the dialog box that appeared when the IWS Scripting Language was used.



When the Tags **v1** and **v2** are entered into the **To** field of the Replace dialog box and the **OK** button is pressed, the IWS tag names **v1** and **v2** will be inserted into the **<default>** field. When the application is run and the Button object on the Screen is pressed, the tag **v1** will be set to a value of 1 and the tag **v2** will be set to a value of 104.

Notes:

- VBScript requires that the Default separator (the **:** character) be used, otherwise an error will occur.
- When using a Custom Property with VBScript, be sure to do the following:
  - You must define all the **<Label>** fields with the Replace dialog box otherwise you will get an VBScript syntax error
  - Be sure that you do not leave any spaces between the $ and the #, otherwise you will get a syntax error

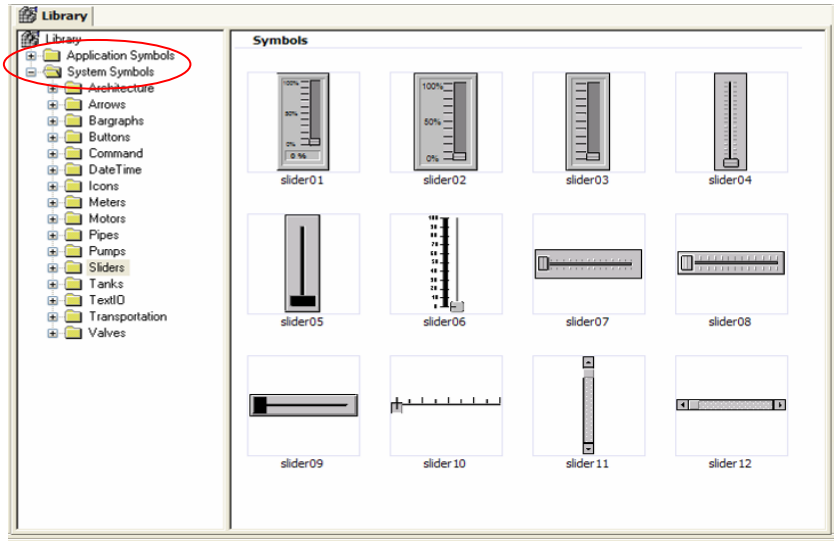
IWS Symbol Library

The IWS Symbol Library contains a group of reusable objects that have been develop for quick and easy insertion into an IWS application. You can add to the Symbol Library as well as modify the existing symbols

The Symbol Library has been revised for IWS Version 6.1 SP2. Most of the Symbols in the Library are now Linked Symbols instead of a Group of Objects. The advantage of Linked Symbols is that they can be changed once and the changes are automatically replicated everywhere the Linked Symbol is used. This makes Symbol updates simple.

The IWS Symbol Library can be accessed from the Main Menu Bar by selecting **View → Library** or selecting the Symbol Library icon in the Standard Toolbar. The Symbol Library consists of a number of Master Linked Symbols stored in the **Systems Folder** in several subfolders that correspond to a functional category. Once a Master Linked Symbol is used in an application, one copy of the Master Linked Symbol is placed in the **Application Symbols** folder of the Symbol Library.

With IWS Version 6.1 SP2, the **System Symbols** folder (containing the Master Linked Symbols that can be used for all IWS applications) is located in the \Symbol subfolder in the directory where the IWS program is installed (default is C:\Program Files\InduSoft Web Studio v6.1). The \Symbol subfolder contains a number of subfolders, corresponding to the folders as shown in the Symbol Library under the System Symbols folder. The **Application Symbols** folder (as shown in the Symbol Library, which contains a copy of the Master Linked Symbol and any custom created Linked Symbols that are used for your application) is stored in your Project directory in a \Symbol subfolder. The Linked Symbols in the Application Symbols folder can only be used by your application, and are not usable by other IWS applications.



Notes:

- Linked Symbols are simply a symbol (group of objects) that can be linked back to a Master symbol. If the Master symbol is changed, all of the copies of the Master Symbol will automatically change.
- IWS provides System-level Master Linked Symbols (**System Symbols**) and local-level Master Linked Symbols (**Application Symbols**). The Linked Symbols can be un-Linked from the Master, if required.
- Master Linked Symbols can be created and edited (modified).
- Linked Symbols cannot be rotated, unlike a Group of Symbols from the legacy Symbol Library

Inserting a Linked Symbol from the Symbol Library onto a Screen

The IWS Symbol Library contains many symbols that you can use for your application. To insert a linked symbol from the Symbol Library onto a Screen, do the following:

- Open (or create) an application Screen
- Open the Symbol Library by
  a) selecting **View → Library** from the Main Menu Bar, or
  b) selecting the Symbol Library icon in the Standard Toolbar, or
  c) double-clicking the Symbol Library icon from the Graphics Workspace.

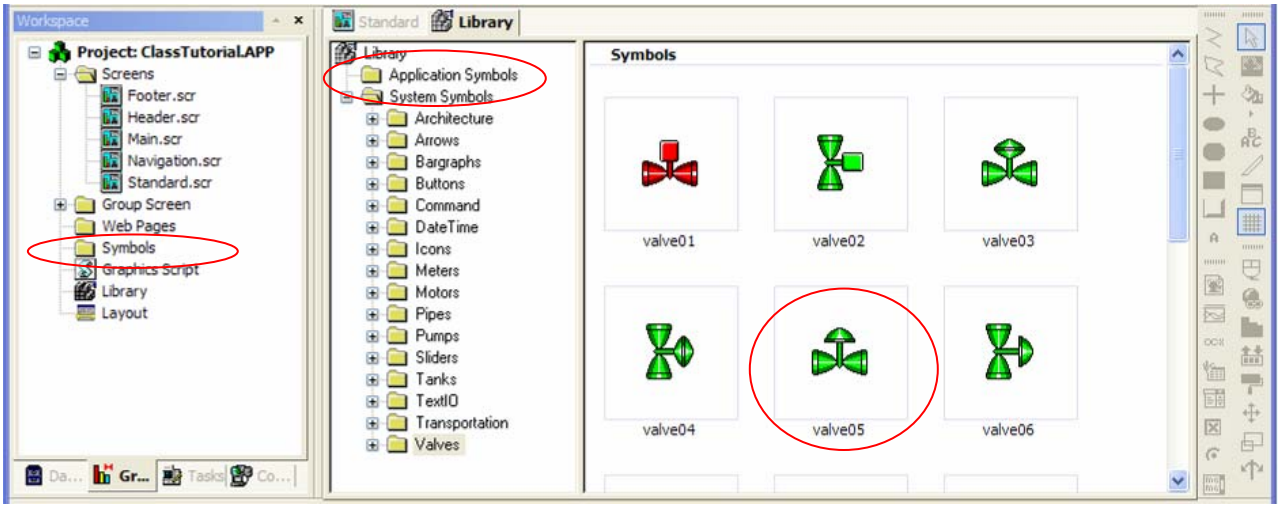
- View the Master Linked Symbols in the Symbol Library by opening the **System Symbols** folder. Left click on the subfolder that contains the category of the symbol you are interested in. You should see all the Symbols in the selected subfolder.
- Left click on Symbol of interest. The Symbol should now be highlighted.
- Go back to your application Screen. Your mouse cursor should have a small rounded rectangle attached to it. Position your mouse cursor where you want to insert the Symbol and left click the mouse.
- You have now inserted a Linked Symbol on your application Screen.

Notes:

- After inserting a Linked Symbol from the Symbol Library (**System Symbols** folder), a copy of the Symbol will be placed in the **Application Symbols** folder in a subfolder identical to the **System Symbols** subfolder from where the Symbol was located. The **Application Symbols** folder contains **Linked Symbols** that are only used for your current application.
- If you make any changes to a Linked Symbol that has been placed on your application Screen, by default the changed Linked Symbol will be saved in the **Application Symbols** subfolder, not in the **System Symbols** folder. You have the option to save the changes to the **System Symbols** subfolder, however.

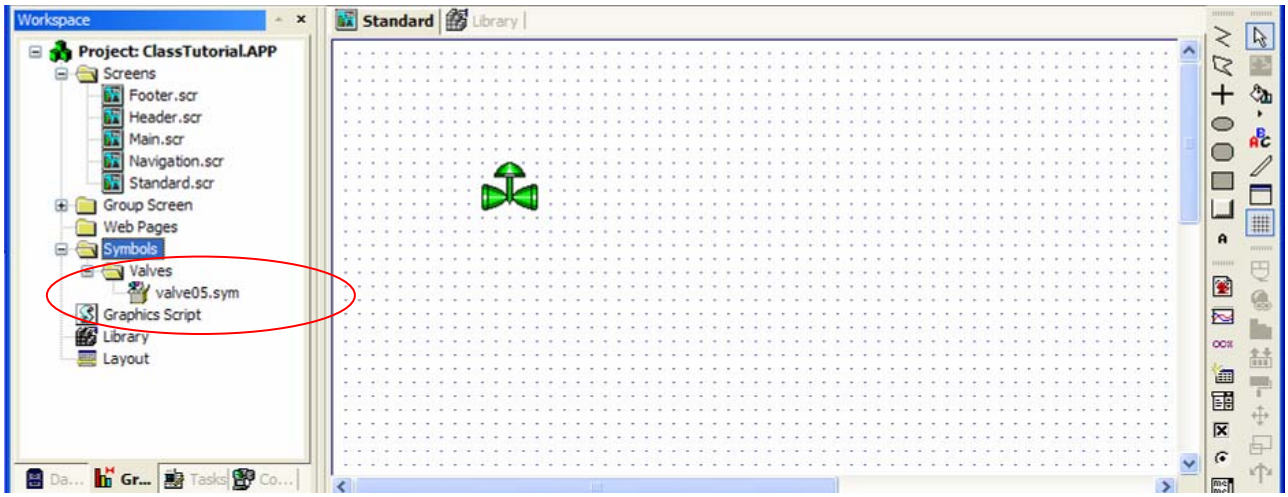
Adding additional Linked Symbols to your Application

Once you have inserted a Symbol from the System Symbols folder onto your application Screen, you may want to insert another identical Linked Symbol on the same Screen or another Screen. You can repeat the above procedure ("Inserting an existing Linked Symbol from the System Library on a Screen") which will insert a Linked Symbol from the **System Symbols** folder, but for reasons to be pointed out shortly, the better way to insert another Linked Symbol on your application Screen is by using the Symbol(s) in the **Application Symbols** subfolder of the Symbol Library.



Notice in the above picture, there are currently no files (or subfolders) shown in the **Application Symbols** folder, shown in the Symbol Library or the **Symbols** folder in the Graphic Workspace (the **Symbols** folder in the Graphic Workspace is the same as the **Application Symbols** folder in the Symbol Library).

If you now select the **valve05** symbol (after selecting it, the border of the Symbol will be highlighted), open your application Screen, position your mouse cursor and left click, the **valve05** symbol will be placed on your application Screen. Note in the picture below that the inserted symbol (**valve05.sym**) is copied into the **Symbols** folder in the Graphic Workspace (the **Symbols** folder in the Graphic Workspace is the same as the **Application Symbols** folder in the Symbol Library).



To insert another **valve05** symbol, you simply place your mouse cursor on the **valve05** icon in the **Symbols\Valves** folder (as shown), right mouse click and select insert. A new copy of the **valve05** symbol will be placed in the upper left corner of your application screen. Select (click) the symbol on your application screen and drag it to the desired location.

The reason that you want to use the existing symbol (from the **Symbols** folder of the Graphics Workspace, or the **Application Symbols** folder in the Symbol Library) to insert another Symbol in your application is that if you make changes to the Linked Symbol, you want to use the modified Linked Symbol, not the original (Master) Linked Symbol contained in the **System Symbols** folder.

Notes:

- The **Symbols** folder in the **Graphics Workspace** is the same as the **Application Symbols** folder in the Symbol Library. The contents of this folder are stored in your Project directory under the subfolder **\Symbol**
- To insert a duplicate symbol, use symbols contained in the **Symbols** folder of the Graphics Workspace, or the **Application Symbols** folder in the System Library. This will preserve this symbol as the Master Linked Symbol for your current application.

Creating a new Linked Symbol (Application or System)

Creating a new Master Linked Symbol is a straight forward process. The steps to do this are as follows:

- Create a new (empty) application Screen, or use a currently opened screen.
- Insert any number of Static Objects, Active Objects, and Dynamic Properties that will be used for the Linked Symbol. You do not need to group these Objects together.
- You may want to insert Custom Properties in one or more of the Objects or Dynamic Properties used. See previous section on how to add Custom Properties.
- Using the Selection Tool in the Mode Toolbar, select the Objects that are to be used for the Linked Symbol
- With the Objects selected that are to be part of the new Master Linked Symbol, click on the right mouse button and select **Create Linked Symbol**.
- A **Save As** dialog box will open, prompting you to enter a unique file name for the symbol. The file will be stored in the **\Symbol** subfolder of your Project folder.
- An icon for the new Symbol will now appear under the **Symbols** folder in the Graphic Workspace. You can select this icon, click the right mouse button and:
  a) Select **Insert** to insert a copy of the Application Linked Symbol in a Screen, or
  b) Select **Send to System Symbols** to copy the Application Master Linked Symbol to the System Symbols folder, making this new Linked Symbol available to all IWS developers.

Notes:

- When you create a new Linked Symbol, you are creating a Linked Symbol for **your Application**. This Linked Symbol, in its current state, is not available to other IWS applications.
- To make the new Linked Symbol to other IWS applications, you must save it into the **System Symbols** folder of the IWS Symbol Library. You can do this through the **Send to System Symbols** option.
- While the Application-level Linked Symbol can have the same symbol name as a Linked Symbol in the **System Symbols** folder, it is highly recommended to uniquely name your new (created) Linked Symbol so as to avoid confusion.

Editing a Linked Symbol

You can edit Linked Symbols. But the catch is that you can only edit an Application Linked Symbol. If you want to change a Linked Symbol in the **System Symbols** folder of the Symbol Library, you must first copy (insert) a System Symbol into your application (making it an Application-level Linked Symbol), edit the Application-level Linked Symbol, then select **Send to System Symbols** to send the updated Linked Symbol to the **System Symbols** folder. When the updated Linked Symbol is stored in the System Symbols folder, other IWS applications can access it.

There are a couple methods to edit a Linked Symbol:

- If the Linked Symbol is inserted on your Screen, then:
  - Select a Linked Symbol (left click on the Linked Symbol)
  - With the Linked Symbol selected, click the right mouse button
  - Select the **Edit Linked Symbol** option
- If the Linked Symbol is not inserted on your active Screen, then:
  - Open the Graphics Workspace.
  - Open the **Symbols** folder if it is closed
  - Click (left mouse click) on the Linked Symbol you want to edit
  - With the Linked Symbol (in the **Symbols** folder) selected, click the right mouse button
  - Select the **Edit** option

After doing one of the above steps, you will now have the Linked Symbol Editor opened. Using the Linked Symbol Editor, you can do any of the following:

- Add, modify or delete Objects and Object Properties
- Add, modify or remove Custom Properties
- Add, modify or remove Dynamic Properties

After changes are made to the Linked Symbol, you need to save the Linked Symbol. This is done by either selecting from the Main Menu Bar **File → Save**, or by clicking the right mouse button on the Linked Symbol name above the Screen and selecting **Save**, or pressing **Ctrl S**.

After saving the linked Symbol, you will need to re-verify the application to insure all Linked Symbols are updated. This is done by saving and closing all Screens, then selecting **Tools → Verify Application** from the Main Menu Bar.

Notes:

- The Linked Symbol Editor is used to edit Linked Symbols. For the most part, it looks and works like a Screen
- You do not need to Group the Objects before opening the Linked Symbol Editor. The Editor will do this for you. However, objects in a Linked Object can be Grouped.

Creating Tooltips for Custom Properties used with Linked Symbols

Custom Properties can be used with regular objects, but when using a Custom Property with a Linked Symbol, you can create ToolTips to remind a developer what information needs to be entered into a Custom Property field.

To create a Tooltip for a Custom, property, perform the following steps:

- Either create a new Linked Symbol or select a Linked Symbol that has been used with your application.
- With the Linked Symbol selected, open the Linked Symbol Editor (reference the "Editing a Linked Symbol" section)
  Note: The object must already have Custom Properties, or you need to add the Custom Properties to the Object. Reference the "Custom Properties" section.
- Position your mouse cursor in an area of the Linked Symbol Editor Screen where there are no objects. Click on the right mouse button and select **Edit Symbol Properties**.
- In the **Description** field of the **Symbol Properties** dialog box, type a Tooltip message for the various Custom Properties.
- When finished, click on the **OK** button, save the updated Linked Symbol and close the Linked Symbol Editor.
- Save and close all open application Screens.
- Select **Tools → Verify Application** from the Main Menu Bar to update all Linked Symbols
- When you double-click on the Linked Symbol in an application Screen and open the Object Properties for the Linked Symbol, moving your mouse over the Custom Properties will now show the Tooltip.

Notes:

- Tooltips are displayed when the mouse cursor passes over the Custom Properties in an open Linked Symbol Object Properties dialog box.
- You open the Linked Symbol Object Properties dialog box by double clicking on the Linked Symbol.

Unlinking a Linked Symbol

You can easily unlink a Linked Symbol by selecting (position you mouse cursor on the Linked Symbol and click the left mouse button) the Linked Symbol in your application Screen, click the right mouse button, and select the **Unlink Linked Symbol** option. Unlinking a Linked Symbol is used to separate the Symbol from the Master Linked Symbol, so that when changes are made to the Master Linked Symbol, the will not update the Symbol.

Note:

- Once an Linked Symbol is un-linked, there is no re-link tool. You can, however, select all of the objects, click the right mouse button and select the **Create Linked Symbol** option.

Deleting a Linked Symbol

You can delete a Linked Symbol by selecting the Application-level Linked Symbol in the **Symbols** folder of the **Graphics** Workspace, clicking the right mouse button and selecting the **Delete** option. You will be prompted to confirm the deletion.

Note that this will only delete Application-level Linked Symbols. It will not delete Linked Symbols that are in the **System Symbols** folder of the Symbol Library.

Paste Link

IWS includes a **Paste Link** tool. This tool allows you to take a source picture file, in multiple image file formats, and paste it in an application. If the source file changes, the picture in the application will change as well. The linked object is not part of the Screen, however.

To use the Paste Link function, select **Edit → Paste Link** from the Main Menu Bar. Paste Linked objects are not Linked Symbols, in that they do not have Objects and Properties associated with them.

Notes:

- The **Paste Link** function is not available for Windows CE-based platforms.
- The **Paste Link** function supports multiple file formats
- It is recommended to place the source picture in Project folder.

Legacy Symbol Library

The Legacy Symbol Library (previous to IWS Version 6.1 SP2) consisted of symbols that were really a Group of Symbols (or objects). If you need to access these legacy symbols, this is possible to do.

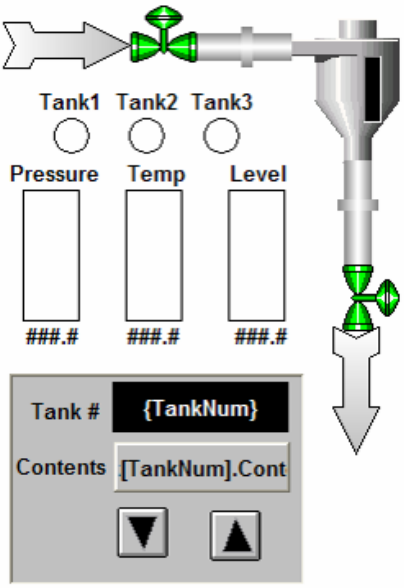
- Be sure a previous version of IWS is installed on your development system in a unique folder from IWS Version 6.1 SP2.
- From the Main Menu Bar, select **File → Open File**.
- In the **Open File** dialog box, point to the **\Lib** folder where the legacy IWS program was installed. This is usually in the Program Files folder of the C:\ drive (e.g. **C:\Program Files\InduSoft Web Studio v6.0\Lib)**
- Select the file type as **All Files** (***.***)
- There will be files that represented the Symbol categories. Click on of them, and click on **Open**. (e.g. Arrows, Bargraphs, Buttons, Common Controls, Conveyors, Date & Time, FileBrowser, Frames, Icons, Links for NT, Meter, Motors, PC_PLC, PID, Pipes, Pumps, Pushbtn, Sliders, Tanks, Trucks, Valves)
- A new Screen will appear in your application containing the set of legacy Symbols in the selected category. These symbols (Group of Symbols) can be copied and pasted into your application.

Notes:

- Symbols from a Legacy Symbol Library can be used in a IWS Version 6.1 SP2 application, but only as a Group of Symbols.
- You can convert the Group of Symbols into a Linked Symbol.

Exercise: Creating a Demo (TankDemo)

This Exercise builds on the previous Exercises. In this Exercise, we will create a simple demo screen that uses Linked Symbols from the Symbol Library and Class Tags. We will create three (3) tanks with Temperature, Pressure and Fill Level Properties. The demo will become functional in a later Exercise. When we complete this Exercise, the **TankDemo** Screen will look as follows:



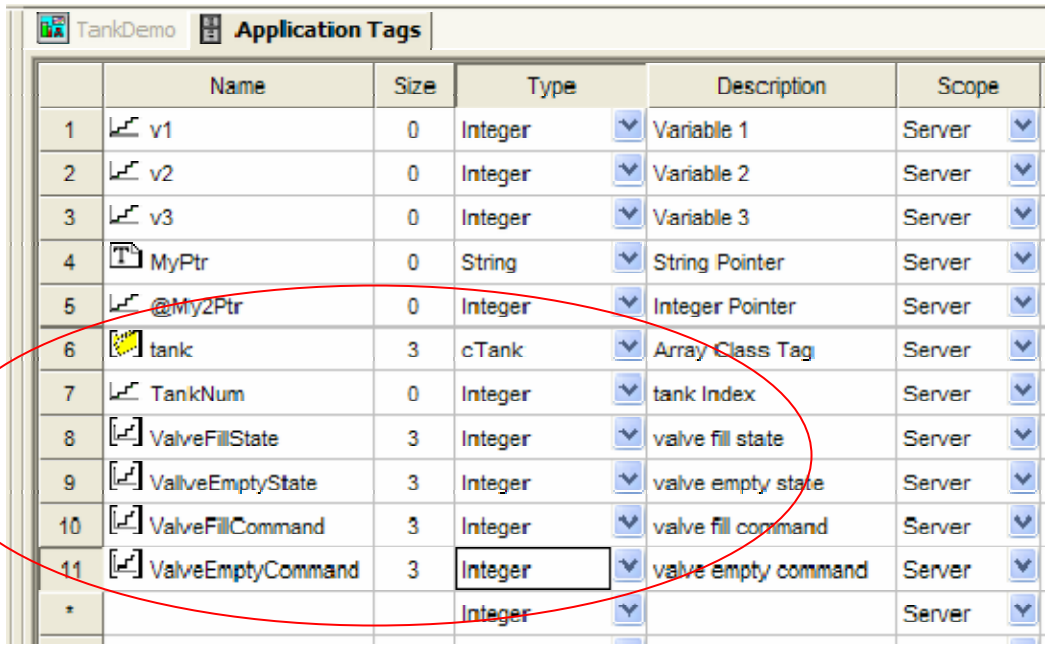
Add Class Members and Tags to the Database

- ☐ Open the **Database** Workspace.
- ☐ Open the **Classes** folder by placing your mouse cursor over the folder and clicking the left mouse button. Click on the folder **cTank** and open (left click) on the **Datasheet View** for Class **cTank** (this Class definition was created in a previous Exercise)
- ☐ We will add descriptions to the Class Members and add six additional Class Members (**Contents, State, StateP, StateT, StateL, InspDate**) as follows shown below:

TankDemo | Class: cTank

	Name	Type	Description
1	Name	String	Tank Name
2	PhysicalLocation	String	Tank Location
3	Capacity	Real	Capacity (gallons)
4	Level	Real	Level (gallons)
5	Temperature	Real	Temperature (F)
6	Pressure	Real	Pressure (psi)
7	Contents	String	Tank Contents
8	State	Integer	Overall State Indicator for Tank
9	StateP	Integer	State of Pressure
10	StateT	Integer	State of Temperature
11	StateL	Integer	State of Level
12	InspDate	String	Last Inspection Date
*		Integer	

- ☐ Right mouse click on the **Class: cTank** icon above the Screen, and select Close.
- ☐ In the **Database** Workspace, open the **Application Tags** folder and double click on **Datasheet View**.
- ☐ The Application Tags database should have an Array Class tag named **tank.** Set its **Size** argument to a value of 3.
- ☐ Add a new integer tag called **TankNum** with a Startup value of 1
- ☐ Add a new integer Tag array **ValveFillState,** size 3
- ☐ Add a new integer Tag array **ValveEmptyState,** size 3
- ☐ Add a new integer Tag array **ValveFillCommand,** size 3
- ☐ Add a new integer Tag array **ValveEmptyCommand,** size 3



	Name	Size	Type	Description	Scope
1	v1	0	Integer	Variable 1	Server
2	v2	0	Integer	Variable 2	Server
3	v3	0	Integer	Variable 3	Server
4	MyPtr	0	String	String Pointer	Server
5	@My2Ptr	0	Integer	Integer Pointer	Server
6	tank	3	cTank	Array Class Tag	Server
7	TankNum	0	Integer	tank Index	Server
8	ValveFillState	3	Integer	valve fill state	Server
9	ValveEmptyState	3	Integer	valve empty state	Server
10	ValveFillCommand	3	Integer	valve fill command	Server
11	ValveEmptyCommand	3	Integer	valve empty command	Server
*			Integer		Server

- ☐ Right mouse click on the **Application Tags** icon above the Screen, and select Save and then Close.

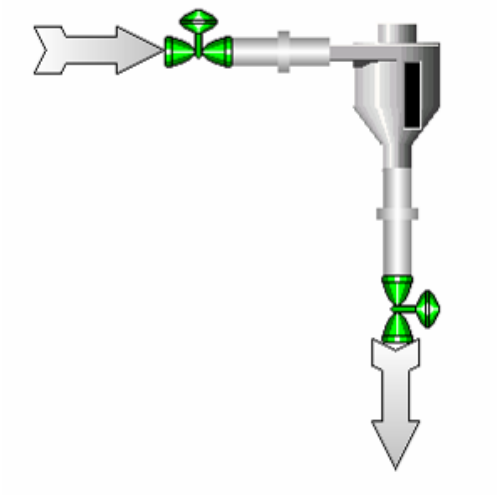
Draw Tank, Valves, Arrows and Piping

- ☐ Open the **TankDemo.scr** Screen by opening the **Screens** folder in the **Graphics Workspace** and double-clicking on **TankDemo**.
- ☐ Open the **Symbol Library** by clicking on the Symbol Library icon in the **Graphics Workspace** or by clicking on the Symbol Library in the Main Menu Bar.

- ☐ In the Symbol Library, select the following Symbols from their respective subfolder and place them, one at a time, in the **TankDemo** Screen. We will arrange them in the next step, but for now just place them on the **TankDemo** Screen.

Symbol Subfolder	Symbol Name
Tanks	tank05
Valves	valve03 and valve04
Arrows	Arrow05 and Arrow07
Pipes	pipe01, pipe02, pipe06 and pipe07

- ☐ Arrange the linked symbols in the following manner on the **TankDemo** Screen:



Notes:

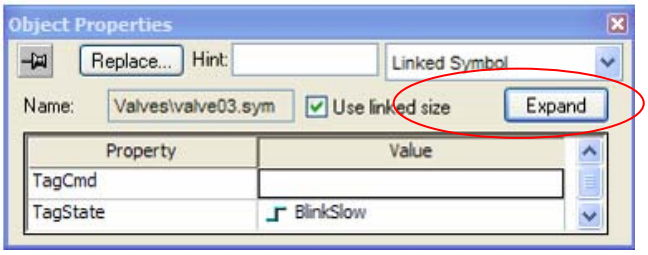
- You can move an object by holding down on the Shift key and pressing the left, right, up and down arrows on your keyboard.
- You can turn off the Snap-to-Grid feature by moving your mouse cursor to a blank area of the screen, click the right mouse button, and select Grid Settings. Uncheck the **Snap to Grid** checkbox.
- In the Grid Settings dialog box, you can change the size of the Screen Grid. The Grid Setting (H) & (V) can be as small as 5 pixels.
- To help draw fine detail objects, you can do the following:
  a) Turn off the Snap to Grid feature (see previous Note)
  b) Make Grid Size 5 (H) x 5 (V) (this is the finest granularity allowed)
  c) In the Standard Toolbar, change Zoom setting to 200-500% as required.

☐ Save the changes you made to the **TankDemo.scr** by positioning your mouse cursor on the **TankDemo** screen icon located just above the Screen, clicking the right mouse button and selecting Save. Alternatively, you can left click on the Save icon in the Standard Toolbar.

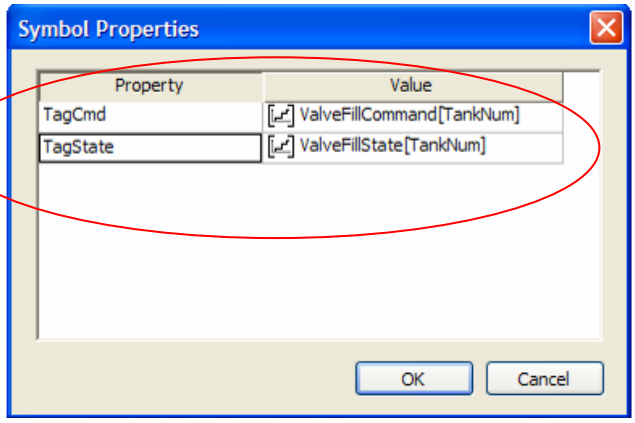
Set Valve Command and State Indicators

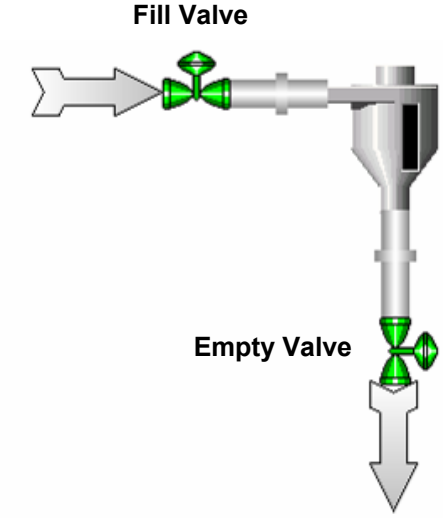
In this step, we will specify the Fill and Empty Valve Commands and State Indicators. The Valves are Linked Symbols.

- ☐ Double click on the **Fill Valve**
- ☐ Click on the **Expand** button
- ☐ Type **ValveFillCommand[TankNum]** in the Value field for the TagCmd Property
- ☐ Type **ValveFillState[TankNum]** in the Value field for the TagState Property
- ☐ Click **OK**

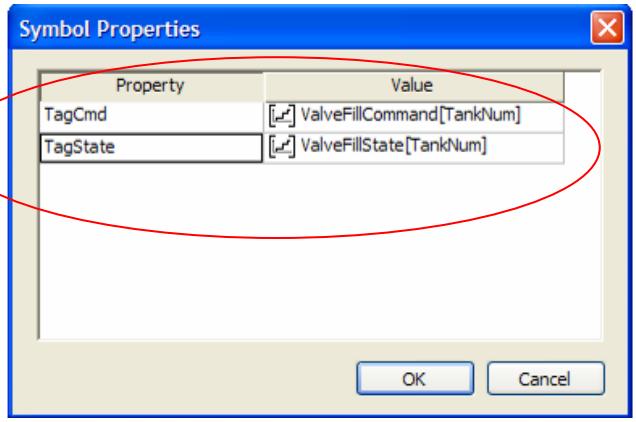


- ☐ Click on the Arrow next to the **Fill Valve.**
- ☐ In the Value field for the State Property, type **ValveFillState[TankNum]**





- ☐ Click on the **Empty Valve**
- ☐ Click on the **Expand** button
- ☐ Type **ValveEmptyCommand[TankNum]** in the Value field for the TagCmd Property
- ☐ Type **ValveEmptyState[TankNum]** in the Value field for the TagState Property
- ☐ Click **OK**

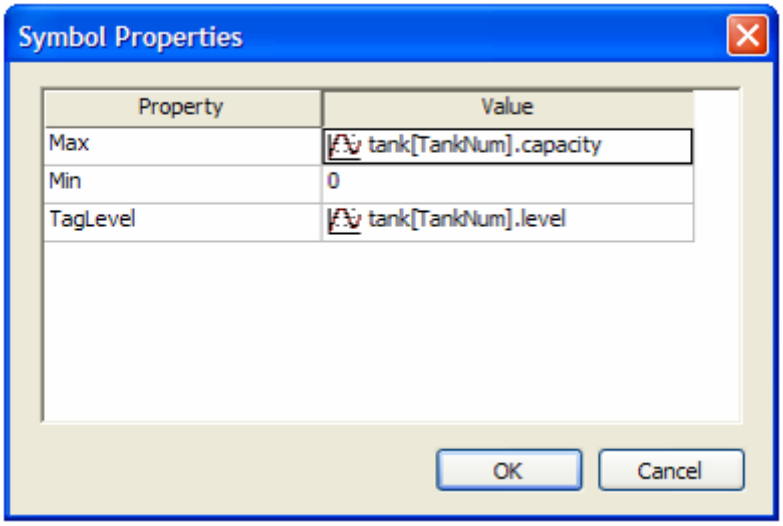


- ☐ Click on the Arrow next to the **Fill Valve.**
- ☐ In the Value field for the State Property, type **ValveFillState[TankNum]**
- ☐ Close the Object Properties dialog box.

Define the Level Indicator for the Tank Symbol

In this step, we will set the parameters that define the Tank Level indicator.

- ☐ Double click on the Tank Symbol
- ☐ Click on the Expand button
- ☐ Specify the following values in the **Value** field as shown below



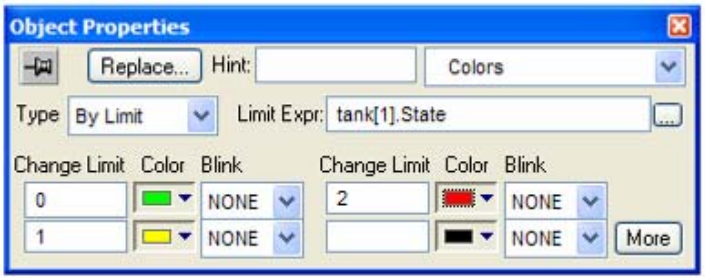
- ☐ Click on **OK**
- ☐ Close the Object Properties dialog box

Draw Tank State Indicators

In this step, we will draw Tank State Indicators. These indicators will indicate if any state (Pressure, Temperature, or Fill Level) is in a warning state.

- ☐ Using the Ellipse Tool in the Static Objects Toolbar, create an Ellipse Object (small circle) on the **TankDemo** Screen that is approximately 22 (W) by 22 (H) pixels in sze. The Ellipse Object properties are as shown:

- ☐ With the Ellipse Object selected, click the Colors Dynamic Property icon. **The Limit Expression** field for the first object will be **tank[1].State** and the **Change Limits** need to be set as shown.

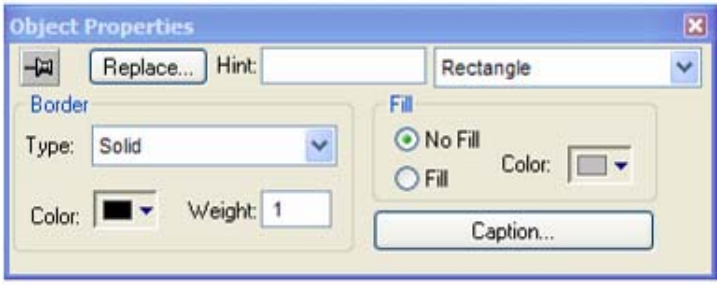


- ☐ Above the Ellipse Object (the circle), insert a Text Object and type **Tank 1**
- ☐ Using the Selection Tool, select the Text Object and the Ellipse Object. You can group them if you want.
- ☐ Using copy (Ctrl C) and paste (Ctrl V), add additional State Indicators for Tanks 2 & 3. You will need to change the Text Object to **Tank 2** and **Tank 3**. You will also need to change the Colors Dynamic **Limit Expression** to **tank[2].State** and **tank[3].State**.
- ☐ Close any open Object Properties dialog boxes.

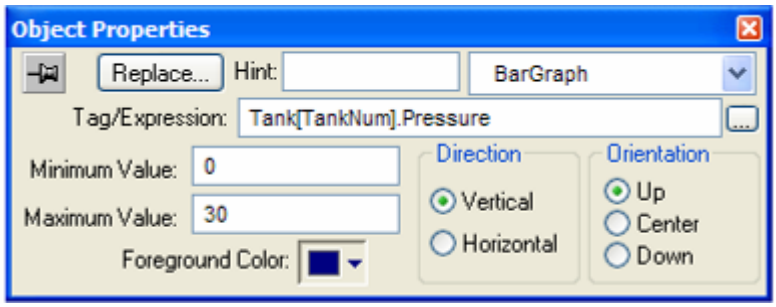
Draw Pressure, Temperature and Level Bar Graphs

In this step, we will create three bar graphs that represent the Pressure, Temperature and Fill Level of the Tank selected.

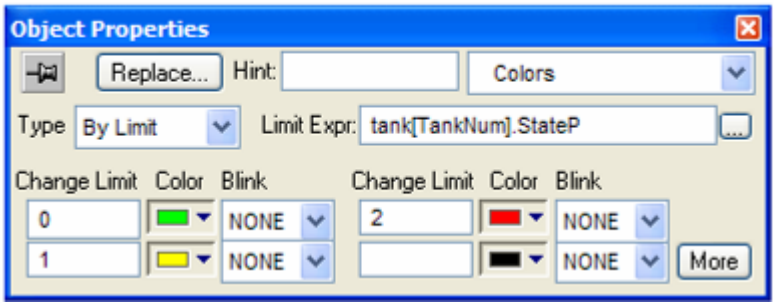
- ☐ Using the Rectangle tool in the Static Objects toolbar, create a Rectangle Object that is approximately 35 (W) by 80 (H) pixels in size, and has the following properties:



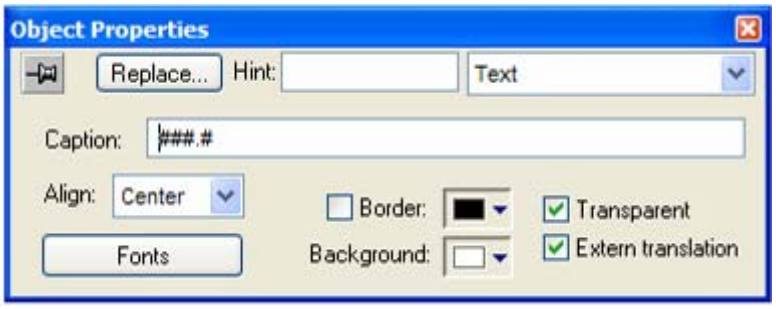
- ☐ With the Rectangle Object selected, click on the **BarGraph** Dynamic Property to add it to the Rectangle Object. Fill in the Properties as shown below.



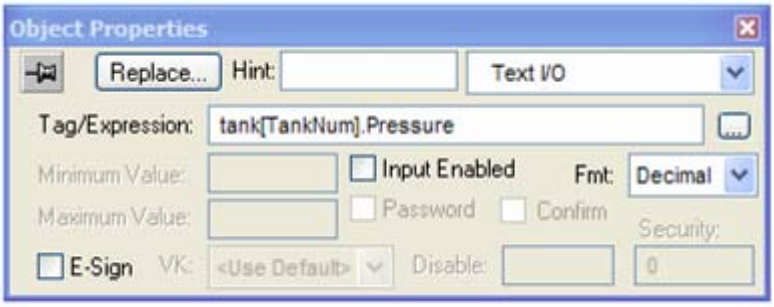
- ☐ With the Rectangle Object selected, click on the **Colors** Dynamic Property to add it to the Rectangle Object. Fill in the Properties as shown below.



- ☐ Above the Rectangle Object, insert a **Text Object** and type **Pressure**.
- ☐ Below the Rectangle Object, insert a **Text Object** and type **###.#** in the Caption field.



☐ With the Text Object selected (the Text Object below the Rectangle Object), click on the **Text I/O** Dynamic Property. Fill in the Properties as shown below.



☐ Group the the two Text Objects and the Rectangle Object.

☐ Copy (Ctrl C) and Paste (Ctrl V) the grouped Object twice (one will be for Temperature, the other for Level. Position them side by side, under the Tank State Indicators.

☐ Ungroup the 2nd Grouped Object and specify the Properties for the various Objects as follows:

Object	**Property**	**Value**
Text Object above	Caption	Temp
Text Object below	Caption	###.#
Text I./O Dynamic Property	Tag/Expression	tank[TankNum].Temperature
BarGraph Dynamic Property	Tag/Expression	taml[TankNum].Temperature
BarGraph Dynamic Property	Minimum Value	0
BarGraph Dynamic Property	Maximum Value	150
Colors Dynamic Property	Limit Expr	tank[TankNum].StateT

☐ Ungroup the 3rd Grouped Object and specify the Properties for the various Objects as follows:

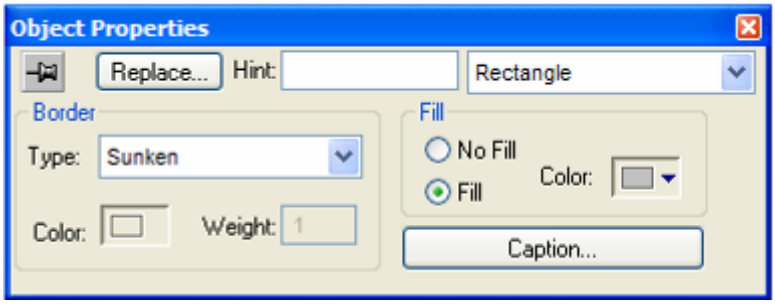
Object	**Property**	**Value**
Text Object above	Caption	Temp
Text Object below	Caption	###.#
Text I./O Dynamic Property	Tag/Expression	tank[TankNum].Level
BarGraph Dynamic Property	Tag/Expression	tank[TankNum].Level
BarGraph Dynamic Property	Minimum Value	0
BarGraph Dynamic Property	Maximum Value	tank[TankNum].Capacity
Colors Dynamic Property	Limit Expr	tank[TankNum].StateL

☐ Close all open Object Properties dialog boxes.

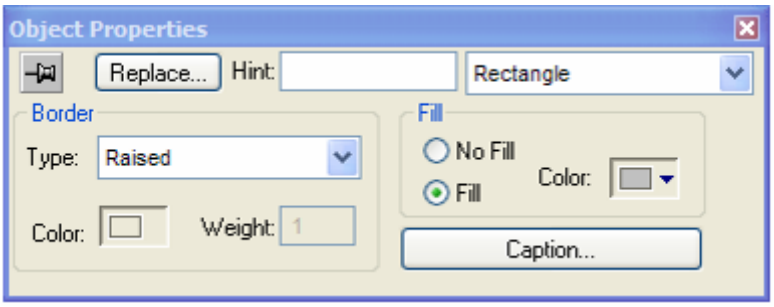
Draw Tank Selector

In this step, we will create a Tank Selector that has up/down buttons selecting select the Tank values to be displayed. The Tank number and its contents will also be displayed

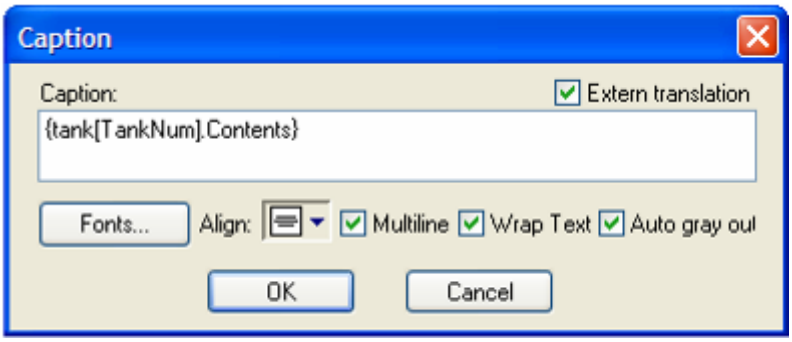
- ☐ Using the Rectangle tool in the Static Objects toolbar, create a Rectangle Object that is approximately 180 (W) by 130 (H) pixels in size. This Object will be used as a background frame. This Object has the following properties:



- ☐ Using the Rectangle tool in the Static Objects toolbar, create a second Rectangle Object that is approximately 110 (W) by 30 (H) pixels in size. This Object will be used as a Frame to display the Tank Contents. This Object has the following properties:

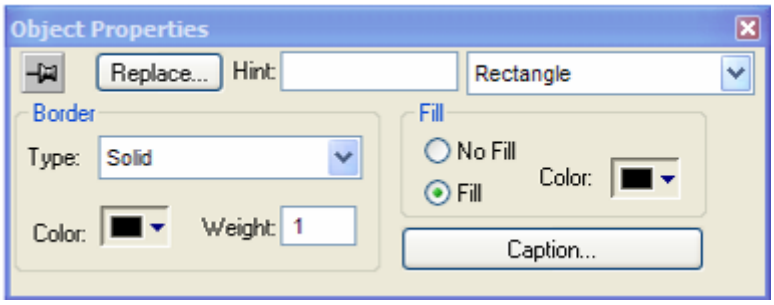


- ☐ In the Caption field, insert the following: **tank[TankNum].Contents**

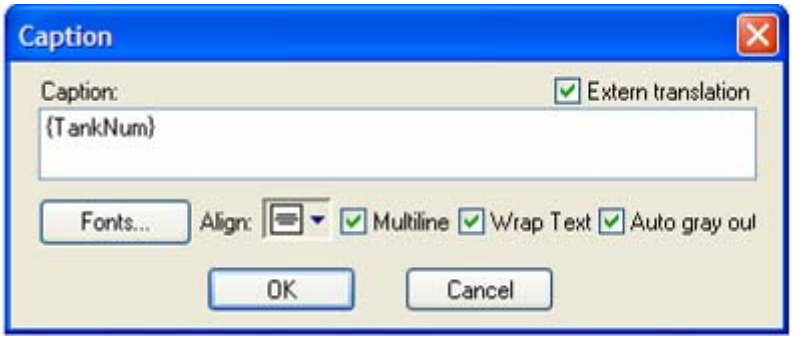


- ☐ Select **OK** and close the Object Properties dialog box.
- ☐ To the left of this Rectangle Object, insert a Text Object with a Caption **Contents.**
- ☐ Close the Object Properties dialog box.

☐ Using the Rectangle tool in the Static Objects toolbar, create a third Rectangle Object that is approximately 110 (W) by 30 (H) pixels in size. This Object will be used as a Frame to display the Tank Number. This Object has the following properties:



☐ In the Caption field, insert **{TankNum}**



☐ Select **OK** and close the Object Properties dialog box.

☐ To the left of this Rectangle Object, insert a Text Object with a Caption **Tank #**

☐ Close the Object Properties dialog box.

Next, we will add up and down command arrows from the Symbol Library.

☐ Insert the following arrows from the **Command** folder:

Symbol Subfolder	Symbol Name
Command	pushbutton_smallup
Command	pushbutton_smallup

- ☐ Right click on the **pushbutton_smallup** symbol and select **Edit Linked Symbol**
- ☐ In the Linked Symbol Editor screen, double click on the symbol (now a Group of Symbols).
- ☐ Click on the combo-box (upper left corner of the Object Properties dialog box) and select the Command Dynamic properties of the Button object.
- ☐ Click the **Config…** button
- ☐ In the **On Down** tab, replace the code **$#TagCmd: = 1** with **If $Tanknum < 3 Then $TankNum = $TankNum +1**.
- ☐ Select the **On Up** tab. Delete any code in this segment. Press the **OK** button.
- ☐ Position your mouse on the **pushbutton_smallup** icon above the screen edit area, right mouse click and select **Save** and then **Close**.

- ☐ Right click on the **pushbutton_smalldown** symbol and select **Edit Linked Symbol**
- ☐ In the Linked Symbol Editor screen, double click on the symbol (now a Group of Symbols).
- ☐ Click on the combo-box (upper left corner of the Object Properties dialog box) and select the Command Dynamic properties of the Button object.
- ☐ Click the **Config…** button
- ☐ In the **On Down** tab, replace the code **$#TagCmd: = 1** with **If $Tanknum > 1 Then $TankNum = $TankNum -1**
- ☐ Select the **On Up** tab. Delete any code in this segment. Press the **OK** button.
- ☐ Position your mouse on the **pushbutton_smalldown** icon above the screen edit area, right mouse click and select **Save** and then **Close**.
- ☐ Save the **TankDemo** screen

- ☐ In the Datasheet View, give the tag **TankNum** a startup value of 1

Notes:

- The above steps were done to eliminate the **TagCmd** custom property, and to insert the VBScript that will increment (or decrement) the TankNum index

Notes

Chapter 9. Active Objects

InduSoft provides a number of Active Objects that can be used with Screens. These include the following Objects::

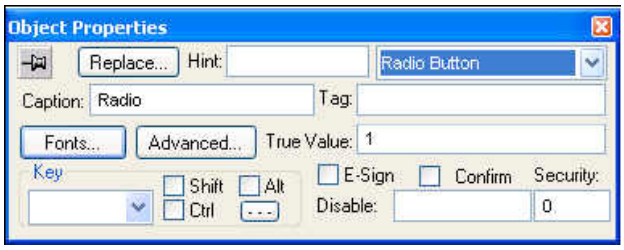
- Radio Button
- Check Box
- Combo Box
- List Box
- Smart Message
- Push Button

In addition to these Active Objects, there are other Active Objects (i.e. Alarm/Event Control, Trend Control, and Grid Object) that will be covered in later Chapters.

Radio Button

The Radio Button icon is used to create a Radio Button Object on an application screen. The Radio Button allows users to select one option from multiple options.

Radio Button parameters



Caption: Allows you to specify a caption

Tag: Tag specified in this field is updated when the user clicks the radio button during runtime

True Value: The value to be written to the Tag specified in the Tag Field when the radio button is clicked.

Fonts: Opens a dialog box allowing you to specify a font style

Advanced: Opens a dialog box that allows you to specify a Tri-State value and Button feedback tag.

Key Field: Specifies a key and any modifiers that will be an equivalent to the radio button being clicked.

E-Sign: If checked, user will be prompted to enter an Electronic Signature before the command executes

Confirm If checked, requires the user to confirm the action at runtime

Disable This field can be a tag or expression. When the tag or expression evaluates to 0, the data input property for the Text I/O Dynamic Property will be enabled (assuming the Input Enabled checkbox is checked). If it evaluates to a value other than 0, it will disable data input.

Security This is a numeric value (tags or expressions not allowed) that defines the security level required for data input to be enabled (provided the Input Enabled checkbox is checked).

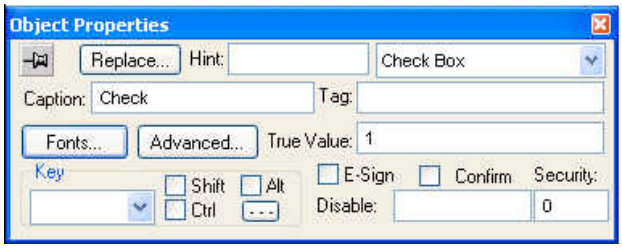
Notes:

- More information on the Radio Button Object can be found in Application Note AN-00-0003

Check Box

The Check Box icon is used to create a Check Box Object on an application screen. The Check Box Object is commonly used to enable or disable an option.

Check Box parameters



Caption: Allows you to specify a caption

Tag: Tag specified in this field is updated when the user clicks the check box during runtime

True Value: The value to be written to the Tag specified in the Tag Field when the Check Box is clicked.

Fonts: Opens a dialog box allowing you to specify a font style

Advanced: Opens a dialog box that allows you to specify a Tri-State value and Button feedback tag.

Key Field: Specifies a key and any modifiers that will be an equivalent to the radio button being clicked.

E-Sign: If checked, user will be prompted to enter an Electronic Signature before the command executes

Confirm If checked, requires the user to confirm the action at runtime

Disable This field can be a tag or expression. When the tag or expression evaluates to 0, the data input property for the Text I/O Dynamic Property will be enabled (assuming the Input Enabled checkbox is checked). If it evaluates to a value other than 0, it will disable data input.

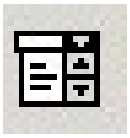
Security This is a numeric value (tags or expressions not allowed) that defines the security level required for data input to be enabled (provided the Input Enabled checkbox is checked).

Notes:

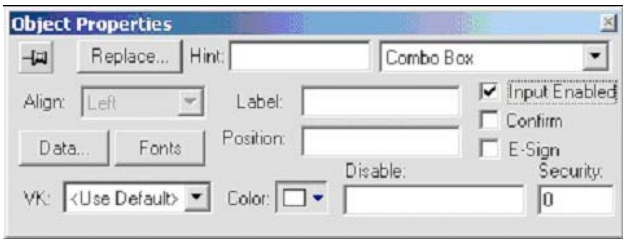
- One difference between the Radio Button and the Check Box is that the Check Box can be unselected.
- If the Tag specified in the **Tag** field is a String Tag, and the **True Value** field contains a valid string, the Tag will be toggled between an empty string and the **True Value**. If the **True Value** field is left blank, the Tag will be toggled between "**Unselected** " and an empty string.
- More information on the Check Box Object can be found in Application Note AN-00-0003

Combo Box

The Combo Box icon is used to place a Combo Box Object on an application screen. Combo Boxes provide a drop down list of labels. The user can select from this list, and both the label number and label string can be returned to Tags.



Combo Box parameters



Align: Specifies label alignment

Label: Used with a string Tag. Receives the label selected by the user.

Position: Used with an integer Tag. Receives the position of the label (zero-based) selected by the user. Can also be used to change the label being displayed.

Data: Opens a **Combo Data** dialog box (more on this below)

Fonts: Opens a dialog box allowing you to specify a font style

VK: Specifies a virtual keyboard to be used

Color: Specifies a background color for the Combo Box

Input Enabled: If checked, allows a user to select a label by typing the contents of that label into a Tag specified in the Label field..

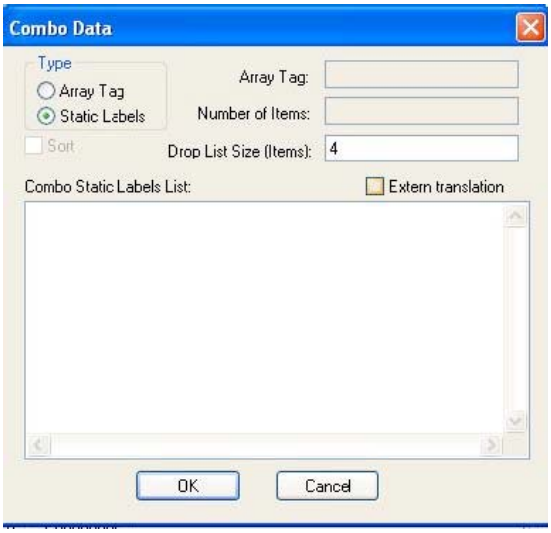
Confirm If checked, requires the user to confirm the action at runtime

E-Sign: If checked, user will be prompted to enter an Electronic Signature before the command executes

Disable This field can be a tag or expression. When the tag or expression evaluates to 0, the data input property for the Text I/O Dynamic Property will be enabled (assuming the Input Enabled checkbox is checked). If it evaluates to a value other than 0, it will disable data input.

Security This is a numeric value (tags or expressions not allowed) that defines the security level required for data input to be enabled (provided the Input Enabled checkbox is checked).

Combo Data parameters



Type: Specifies whether an array of labels or a static list of labels is to be used.

Array Tag: If an array of labels is used, specifies that string tag array name.

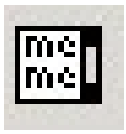
Number of Items: Integer or Tag that defines the number of labels to be displayed. (Zero-based)

Drop List Size: Integer or Tag that defines the number of labels to be viewed in the Combo Box at one time. If (Number of Items) > (Drop List Sze), then a scroll bar will appear.

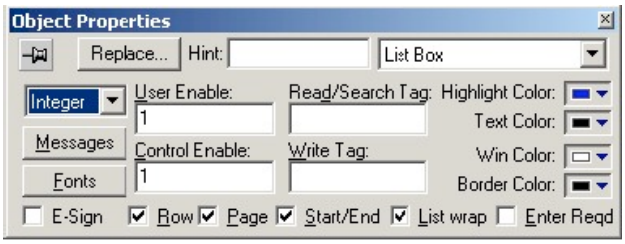
Combo Static Labels List: Used to specify Static Labels. The first item is position 0.

List Box

The list box icon is used to create a List Box Object on an application screen. It is used to display a list of predefined messages during runtime.



List Box parameters



Value: A drop-down list that allows the following tag values to be used for the index to the List Box messages; **Boolean**, **Integer** (default), **LSB** (least significant bit).

User Enable: A tag or expression that allows the user (or application) to select a message at runtime (Default value is 1 (enabled)).

Read/Search Tag: An Integer or Boolean tag to point to a selected Message.

Control Enable: A tag or expression to enable selecting a message in the runtime application depending on the runtime application.

Write Tag: An optional string Tag to receive the selected **Message** string.

Fonts: Opens a dialog box allowing you to specify a font style

E-Sign: If checked, user will be prompted to enter an Electronic Signature before the command executes

Row: A check box that, if checked, includes set up and set down arrows with the scroll bar

Page: A check box that, if checked, includes page up and page down arrows with the scroll bar

Start/End: A check box that, if checked, includes home and end arrows with the scroll bar

List Wrap: A check box that, if checked, will scroll to the beginning of the Message list when the end is reached, or will scroll to the end of the Message list when then beginning is reached.

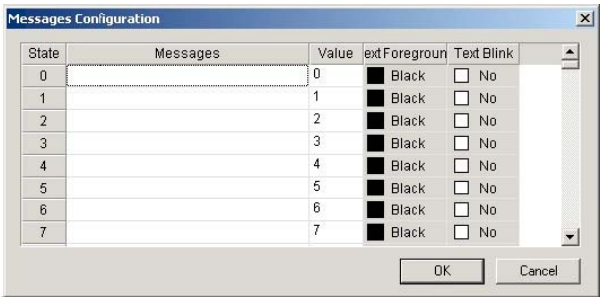
Enter Reg'd: A check box that, if checked, allows selecting messages using the **Enter** key only, and disables the **Tab** key for this Object.

Color boxes: Selects colors for the highlighted message, text, background and border

Disable This field can be a tag or expression. When the tag or expression evaluates to 0, the data input property for the Text I/O Dynamic Property will be enabled (assuming the Input Enabled checkbox is checked). If it evaluates to a value other than 0, it will disable data input.

Security This is a numeric value (tags or expressions not allowed) that defines the security level required for data input to be enabled (provided the Input Enabled checkbox is checked).

Message Configuration parameters



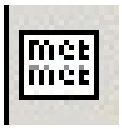
Messages: A field used to enter the message

Value: A value used to identify a specific message.

Text Foreground: Specifies a color for the message text

Text Blink: If checked, will cause the text to blink once per second when the text is displayed.

Smart Message

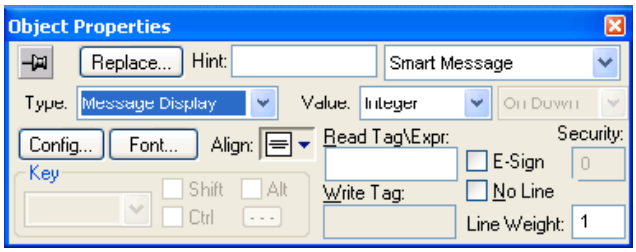


The Smart Message icon is used to create a Smart Message Object on an application Screen. Smart Messages can be used to display messages and graphics based on a Tag value at runtime. A Smart Message Object can be used to display:

- **Message Display**
  Displays one of several messages based on a Tag value
- **MultiState Indicator**
  Similar to the Message Display option but can also display bitmap images with the message
- **MultiState Pushbutton**
  Functions as a multi-position pushbutton that toggles between messages (message and bitmap images).

Smart Messages all can read a Tag value to determine which message to display, but vary in their display ability and ability to write to a Tag.

Smart Message parameters



Type: Combo box that determines Smart Message type:
Message Display
Multistate Indicator
Multistate Pushbutton

Value: Combo box that determines the type of Tag value used to index the Message. Values are:
Boolean – Provides 2 valid states
Integer – Provides 500 valid states (16 for Multi-state Pushbutton)
LSB – Provides 32 valid states (16 for Multi-state Pushbutton)

Read Tag/Expr: Integer or Boolean Tag used to specify the Message to be displayed at runtime.

Write Tag: An optional Integer or Boolean Tag (only for Multistate Pushbutton) returns the message number being displayed.

Config: Launches a **Configuration** dialog box that allows you to specify messages

Fonts: Opens a dialog box allowing you to specify a font style

Align: Specify Message alignment

Key Field: Specifies a key and any modifiers that will be used at runtime for Smart Message selection.

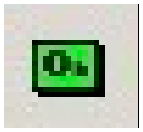
E-Sign: If checked, user will be prompted to enter an Electronic Signature before the command executes

No line: A check box that, if checked, will hide the line border of the Smart Message. If unchecked, you can enter a Line Weight to be used.

Security This is a numeric value (tags or expressions not allowed) that defines the security level required for data input to be enabled (provided the Input Enabled checkbox is checked).

Notes:
- Reference the Users Manual for a complete description of the **Config** dialog box settings.

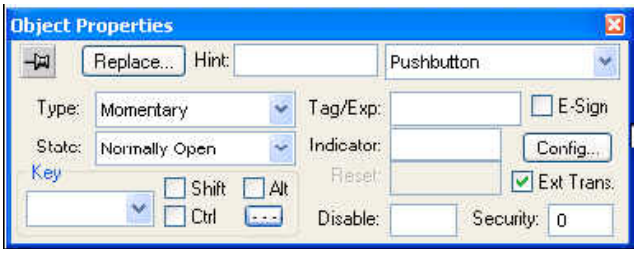
Push Button



The Pushbutton icon is used to create Push Button Objects on an application Screen. The type of pushbuttons that can be created are:

- **Momentary**
- **Maintained**
- **Latched**

Pushbutton parameters



Type: Combo box that determines the Pushbutton:

Momentary
Maintained
Latched

State: Determines the default state for the pushbutton (Normally On or Normally Closed).

Tag/Expr: A Tag or Expression.

If a Tag: Receives the **Write Value** from the appropriate state area (Open or Closed) in the **Configuration dialog.**

If an Expression: Executes when the **On Down** (button press) occurs

Indicator: A Tag that causes the button to change to a specified color when the Tag value matches one of the two specified values configured in the **Configuration** dialog.

Config: Launches a **Configuration** dialog box that allows you to specify style and State parameters for the Pushbutton Object

Key Field: Specifies a key and any modifiers that will be used at runtime for Smart Message selection.

E-Sign: If checked, user will be prompted to enter an Electronic Signature before the command executes

Ext. Trans.: Allows external translation

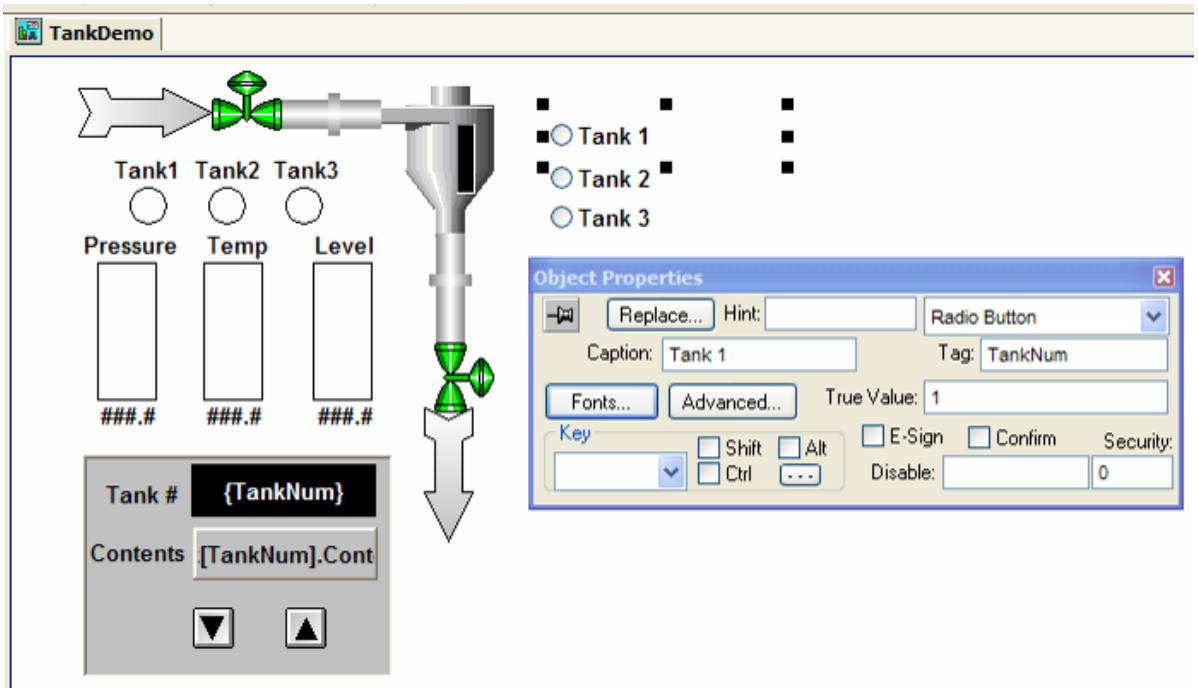
Reset: Can specify a Tag to control button's latched state. 0 = remain latched after button is pressed, Non-zero = unlatched after the button is pressed.

Disable This field can be a tag or expression. When the tag or expression evaluates to 0, the data input property for the Text I/O Dynamic Property will be enabled (assuming the Input Enabled checkbox is checked). If it evaluates to a value other than 0, it will disable data input.

Security This is a numeric value (tags or expressions not allowed) that defines the security level required for data input to be enabled (provided the Input Enabled checkbox is checked).

Exercise: Using Active Objects

In this exercise, we will add 3 Radio Buttons to the TankDemo screen. These Radio Buttons can be used as an alternative to the Up and Down arrow buttons for tank selection.



- ☐ Create 3 Radio Buttons as shown in the graphic above
- ☐ Set each Object's properties as listed below

Object 1:	Caption	Tank 1
	Tag	TankNum
	True Value	1
Object 2:	Caption	Tank 2
	Tag	TankNum
	True Value	2
Object 3:	Caption	Tank 3
	Tag	TankNum
	True Value	3

- ☐ Close the Object Properties dialog box
- ☐ Save the TankDemo Screen

Notes

Chapter 10. Working with Scripting Languages

IWS provides you with two different scripting languages; the IWS Scripting Language and VBScript. This Chapter is an introduction to the IWS Scripting Language and the various built-in functions that IWS provides. VBScript will be covered in a later Chapter. IWS built-in functions are also accessible from VBScript.

One question might be; which scripting language do I use (IWS Scripting Language or VBScript)? The simple answer is that you can use either. But more likely, the answer depends on what you are doing and where in the application you are doing it. If you need just a few lines of Script or want to call a built-in function, the IWS Scripting Language works great. If you need a several lines of complex logic or have nested conditional processing, VBScript might be a better choice.

Overview of the IWS Scripting Language

The IWS Scripting Language is very straight forward to use. All IWS Scripting Language is used in conjunction with a Worksheet, that consists of three fields; **Line Number**, **Tag Name** and **Expression**.

	Tag Name	Expression
1	v3	v1 + v2
*		
*		

Example Script

By default, the IWS Script will execute sequentially beginning with the first line. Any conditional logic contained in the Script can alter the execution sequence. Other than an indication of the sequence of execution, **Line Numbers** serve no real purpose; i.e. you do not have the ability to jump to a line number.

The **Expression** field contains any mix of one or more variables, constants, operators, or functions to be evaluated when the **Line Number** containing the Expression is executed. The **Tag Name** field specifies an IWS tag name used to store the result of the Expression contained in the Expression field. For example, if you wanted to add the values of IWS Tags **v1** and **v2** together and store the result into an IWS Tag **v3**, this could be done by entering the Tags in a Script Worksheet as shown above.

Generally, the IWS Script will contain a Tag in the **Tag Name** field. However, if you want to execute an IWS built-in function (e.g. Open a specified Screen), and do not care about any returned values, you can leave the **Tag Name** field blank.

Worksheet examples include:

	Tag Name	Expression
1	v3	v1 + v2 //add 2 numbers
2	TankNum	1
3	tank[TankNum].Contents	"Fructose"
4	v2	v2 *4 + v1 + 6
5	MyPtr	"v1"
6	v2	@MyPtr + 5 //v2 set to v1 +5
7		Trace("Finished with Worksheet 1")
8		Open("Startup.sg") //Open Startup Screen
9		// Finished
*		

Notes:

- Comments can be added in the **Expression** field following the *//* characters
- The **Tag Name** field does not need to have a Tag name if the **Expression** field contains an IWS function.
- As shown in the **Example Script**, the expression v3 = v1 + v2 is implemented by placing v3 in the **Tag Name** field, and placing the expression v1 + v2 in the **Expression** field. In this example, v1 + v2 is the Expression to be evaluated, and v3 is the result of the Expression.
- Using the expression v3 = v1 + v2, the **Expression** field is for everything to the right side of the equals sign, while the **Tag Name** field is for everything to the left side of the equals sign (a single Tag Name only in this field). The equals sign (**=**) is not used. If an equals sign (**=**) is contained in the **Expression** field, this is treated as a comparison operator.
- The IWS Scripting Language is not Case-Sensitive

IWS Scripting Language Syntax

The IWS Scripting Language is included with the standard IWS development environment, and allows the developer to manipulate IWS tags through the use of Arithmetic, Logical and String Operators, and call any of the built-in IWS Functions. The IWS Scripting Language is a little different than other traditional programming languages (e.g. VBScript) in that certain Operators are implemented in the IWS Scripting Language as Functions. An example of this is the Exponentiation Operator (**^**).

IWS Supported Tag Data Types

IWS Scripting Language supports the following IWS Tag Data Types:

- **Boolean**
  A one-bit value (0 or 1) used to indicate a False or True state
- **Integer**
  A signed 32-bit value.
  Integers can be entered in Decimal or Hexadecimal format (e.g. 0xBC4) format
- **Real**
  An 8-bye double-precision floating point value.
- **Strings**
  A character string up to 255 characters in length. Can be ASCII or Unicode. Strings are enclosed in double quotes (e.g. “abc”).
- **Arrays**
  Set of tags with the same name but unique indexes.
- **Classes**
  Compound Tags that consist of user-defined data types (Boolean, Integer, Real or String type only), The Class elements are called members
- **Pointers** (Indirect Tags)
  Pointers provide indirect access to another tag type (including Arrays and Class Tags)

More about Boolean Tags

The memory storage element for a Boolean Tag is not a single bit but is instead a 32-bit (4-byte) memory cell. This is normally irrelevant, as long as Boolean tags are manipulated individually or using Logical Operators (e.g. **And, Or,** etc.). But it is possible to manipulate a Boolean Tag so that its value is not 0 or 1. In fact, VBScript uses values of **0** for **False**, and **-1** for **True**.

How IWS handles the values for Boolean **True** depends on the settings in the <Application>.APP file. Using Microsoft Notepad or some other text editor, open the <Application>.APP file (be sure IWS is not running this file). In the **[Options]** section, there is a setting for **BooleanTrueAboveZero**. If **BooleanTrueAboveZero=0**, then Boolean Tags will assume the value of 1 (logical **True**) whenever they receive a value different than 0 (i.e. a positive or negative value). If **BooleanTrueAboveZero=1**, then Boolean Tags assume the value of 1 (logical **True**) only when they receive a value greater than 0 (i.e. must be a positive integer).

Notes:

- The **BooleanTrueAboveZero** parameter is found in your Application File <Application>.APP, found in your application folder. To access this file, open your application folder, position your mouse cursor over the <Application>.APP file and click the right mouse button. Select the **Open With** option and select Microsoft Notepad. Be sure the IWS Application (Development and Runtime) is closed.
- By default, the **BooleanTrueAboveZero** parameter is set to 0.
- If you will be developing your application using VBScript, be sure to set the parameter **BooleanTrueAboveZero** to 0 (default setting), since VBScript uses the value of -1 for logical True. IWS uses the value of 1 for logical True.

Arithmetic Operators and Functions

Arithmetic Operators are used to calculate a numeric value, and can be used in conjunction with a Comparison Operator. The Subtraction Operator (**-**) can also be used as a unary operator to indicate a negative number or the negative of a Tag or Expression. The Addition Operator (**+**) can be used for the addition of two numbers or to concatenate strings.

Operator	Name	Description
+	Addition	Adds two numbers together
-	Subtraction	Subtracts one number from the other
*****	Multiplication	Multiplies two numbers
/	Division	Divides one number by the other

Addition Operator (+)

Description Adds two numbers
Usage *number1 + number2*
Result The result type is numeric or string, depends on the data type in the **Tag Name** field
Remarks The Addition Operator evaluates in floating point format and converts to the data type of the Tag used in the **Tag Name** field. The Addition Operator (**+**) is also used for String Concatenation.

Number 1	Number 2	Tag Name Type	Notes
Boolean	Boolean	Boolean	It is not advised to use Boolean Tags in this manner. If Number 1 = 1 (True) and Number 2 = 1 (True), then the Boolean Tag in the Tag Name field will equal 2. However, if this value is used to set another Boolean Tag, the result will be a 1 (Boolean True).
Boolean	Integer	Integer	Will add a 0 or a 1 to the Integer value.
Boolean	Real	Real	Will add a 0 or a 1 to the Real value.
Integer	Boolean	Boolean	Will evaluate expression in floating point format. If evaluation result <>0, then Boolean Tag in Tag Name field is set to 1, otherwise 0.
Integer	Integer	Integer	Will evaluate expression in floating point format. Result is in a Integer format (i.e. any fractional result removed).
Integer	Real	Real	Will evaluate expression in floating point format. Result is in a Real format.
Real	Boolean	Boolean	Will add a 0 or a 1 to the Real value.
Real	Integer	Integer	Will add Integer to the Real, and return a Real result.
Real	Real	Real	Will produce a real result

Subtraction Operator (-)

Description Finds the difference between two numbers or indicates the negative value of a numeric expression (unary operator)
Usage *number1 - number2*
Result The result type is numeric, with the data type depending on the Tag in the **Tag Name** field.
Remarks As with the Addition Operator, the expression is evaluated in floating point and the Tag in the **Tag Name** field will determine the Data Type (format) of the result.

Multiplication Operator (*)

Description	Multiplies two numbers
Usage	*number1 * number2*
Result	The result type is numeric, with the data type depending on the Tag in the **Tag Name** field.
Remarks	As with the Addition Operator, the expression is evaluated in floating point and the Tag in the **Tag Name** field will determine the Data Type (format) of the result.

Division Operator (/)

Description	Divides two numbers
Usage	*number1* / *number2*
Result	The result type is numeric, with the data type depending on the Tag in the **Tag Name** field.
Remarks	As with the Addition Operator, the expression is evaluated in floating point and the Tag in the **Tag Name** field will determine the Data Type (format) of the result. For example, if you had an Integer numerator (*number1*) = 3, and integer denominator (*number2*) = 2, the Integer result of *number1*/*number2* + *number1*/*number2* (i.e. 3/2 + 3/2) is equal to 3. If the denominator is 0, an error will occur.

Comparison Operators

Comparison operators are used to compare Tags (all IWS Tag Data Types), Expressions and Constants (numeric values) against other Tags, Expressions and Constants. The result of the comparison is either a 1 (logical **True)** or a 0 (logical **False**).

Operator	Name	Example	Description
<	Less than	a < b	Returns 1 (True) if a < b
<=	Less than or equal	a <= b	Returns 1 (True) if a is not greater than
>	Greater than	a > b	Returns 1 (True) if a is greater than b
>=	Greater than or equal	a >=b	Returns 1 (True) if a is not less than b
=	Equals	a = b	Returns 1 (True) if a is equal to b
<>	Not equal	a <> b	Returns 1 (True) if a is not equal to b

String Concatenation Operators

The String Operator is used to concatenate (combine) strings.

Operator	Name	Example	Description
+	String Concatenation	“AB” + “CD”	Concatenates two strings

Notes:

- If you concatenate a string Tag and a Numeric Tag, the value of the Numeric Tag will be converted to ASCII or Unicode format and concatenated with the String tag.
- The String Concatenation operator is the same symbol as the Arithmetic Operator. If the Tag in the **Tag Name** field is a String Data Type, the String Concatenation Operator will be used. If the Tag in the **Tag Name** field is a numeric (Boolean, Integer or Real) Data Type, the Arithmetic Operator will be used.
- String Tags or String Values cannot be used as arguments if the Tag in the **Tag Name** field is a numeric Data Type (Boolean, Integer, or Real). The result will be 0.

Logical Operators

Logical Operators are used to perform logical operations on numeric Tags (Boolean, Integer and Real), Expressions and Constants. A numeric Tag, Expression or Constant is Logical True if its value is <>0 (Depends upon **BooleanTrueAboveZero** parameter).

Operators	Function	Example	Returns
And	Logical And	a And b	1 (Logical True) only if a and b are both True
Not	Logical Not	a Not b	1 (Logical True) if a is false; False if a is True
Or	Logical Or	a Or b	1 (Logical True) if a or b is true, or both are True
Xor	Logical Exclusive Or	a Xor b	1 (Logical True) if a or b is True, but not both

Logical Bit Operators

Bit Operators are used to perform bit-wise operations on numeric Tags (Boolean, Integer and Real), Expressions, and Constants. The result of the bit-wise Operator is based on the values of the operands and the Data Type of the Tag in the **Tag Name** field. These functions would normally be used on Integer Tags.

Operators	Function	Example	Note
&	Bitwise And	a & 0xF	Performs bitwise And operation
|	Bitwise Or	a | 0x77	Performs bitwise Or operation
~	Bitwise Not	~ a	Performs bitwise Not
^	Bitwise Xor	a ^ 0xF	Performs bitwise Exclusive Or
>>n	Rotate **n** bits to right	a >> 3	Shift bits right by **n** bits
<<n	Rotate **n** bits to left	a << 2	Shift bits left by **n** bits

Operator Precedence

When numeric Expressions contain Operators from more than one category, there is a precedence to the order by which the Operators are executed. Arithmetic operators are evaluated first, comparison operators are evaluated next, and logical operators are evaluated last. Comparison Operators all have equal precedence, as do Logical Operators; that is, they are evaluated in the left-to-right order in which they appear. Arithmetic Operators are evaluated in the following order of precedence:

Arithmetic	Comparison	Logical	Priority
Negation (**-**)	Equality (**=**)	Not	
Multiplication and division (*****, **/**)	Inequality (**<>**)	And	
Addition and subtraction (**+**, **-**)	Less than (**<**)	Or	
	Greater than (**>**)	Xor	
	Less than or equal to (**<=**)	&	
	Greater than or equal to (**>=**)	|	
		~	
		^	
		>>n	
		<<n	

Highest Priority → Lowest Priority

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 precedence of Expression evaluation can be altered by enclosing portions of the Expression in parentheses. The portion of the Expression enclosed inside of a parentheses will be evaluated first. If multiple parentheses are used, they will be evaluated from left to right.

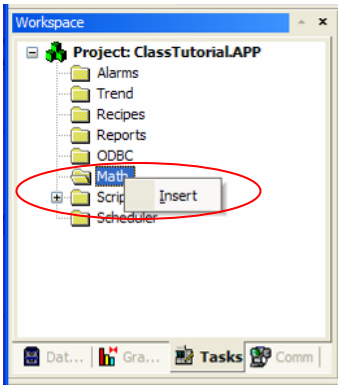
IWS Scripting Language Usage

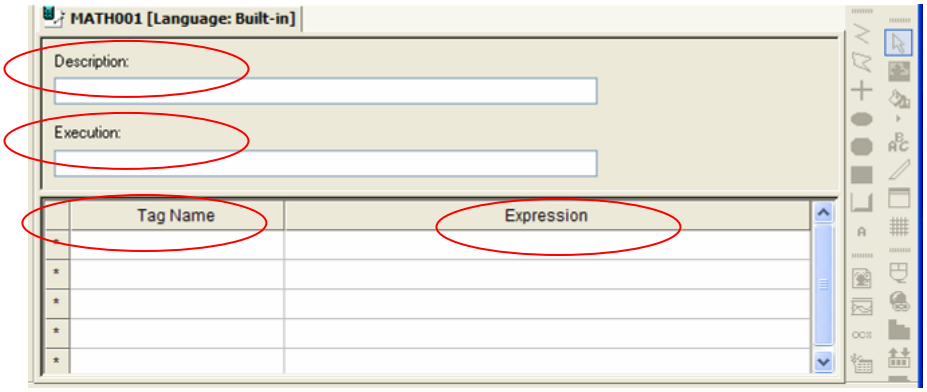
There are distinct areas within an IWS application where the IWS Scripting Language can be used. These include:

- **Math Worksheets**
- **Command Dynamic** for Static and Dynamic Objects
- **Screen Logic**
- **Scheduler**

Math Worksheets

Math Worksheets are located in the **Math** folder in the **Tasks** Workspace. You can select an existing Math Worksheet from this folder, or you can insert a new Math Worksheet by placing your mouse cursor on the **Math** folder and clicking the right mouse button. Select the **Insert** option.





A new Math Worksheet will be generated as shown above. Each Math Worksheet is uniquely numbered and has the **Tag Name** and **Expression** fields as previously described. Math Worksheets also have two additional fields; the **Description** and **Execution** fields.

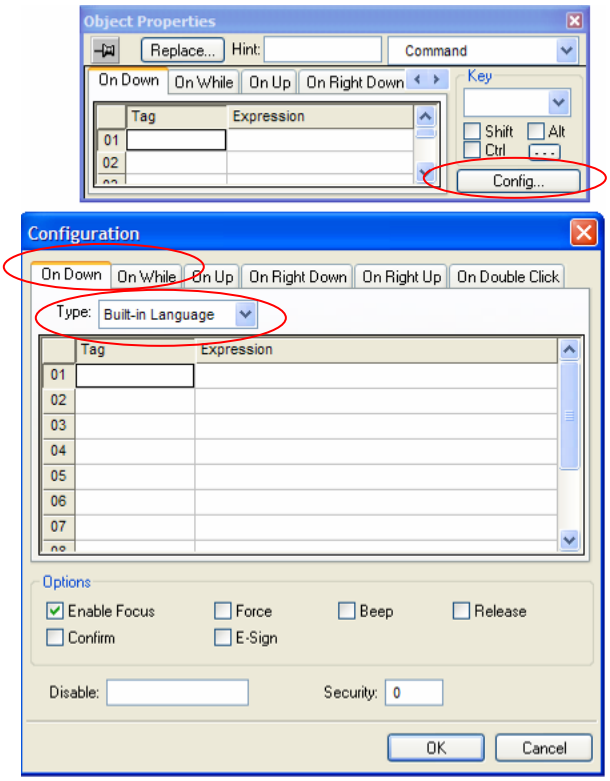
- The **Description** field is optional and is used only for documentation purposes.
- The **Execution** field controls the execution of the Math Worksheet. All Math Worksheets execute in a background mode (i.e. the Math Worksheet will execute whether or not a Screen is open), and the **Execution** field controls whether this particular Math Worksheet executes or not. If the **Execution** field contains a value of <>0 (Logical True), or evaluates to a value of <>0, then the Math Worksheet will execute. If the value or Expression in the **Execution** field evaluates to 0, the Math Worksheet will not execute. The Execution field can contain any of the following:
  - A Constant (i.e. a value of 1)
  - An IWS Tag
  - Calls to IWS built-in functions
  - An Expression (consisting of Tags, Constants and numeric Operators)

Notes:

- The **Execution** field must be set to, or evaluate to, a logical True in order for the Math Worksheet to execute.
- Logical True is defined by the parameter **BooleanTrueAboveZero**
- The Math Worksheet Execution field is evaluated periodically and, if evaluated to be a Logical True, then the Math Worksheet will be executed. This period is defined in the **[Period]** section of the **Program Settings.ini** file located in the \Bin folder in the directory where the IWS program is installed (e.g. **C:\Program Files\InduSoft Web Studio v6.1\Bin\Program Settings.ini.**
- You can have multiple Math Worksheets for your application. Each is uniquely numbered, and runs as a background task (i.e. is not associated with a Screen).

Command Dynamic

The Command Dynamic, as discussed in a previous Chapter of this Training Guide, is a Dynamic Property that can be associated with a Static Object and several Active Objects. The Command Dynamic allows certain predefined, as well as programmable, actions to be taken when a Static or Dynamic Screen Object is selected during runtime operation.



Executing a segment of the IWS Scripting Language is one of the options for the Command Dynamic to execute when the Static or Dynamic Screen Object is selected at runtime. The Command Dynamic is currently limited to 12 lines of IWS Scripting Language.

The Command Dynamic is added to a Static or Dynamic Screen Object by selecting the Screen Object and selecting the Command Dynamic from the Dynamic Properties Toolbar. Then, when you double click on the Screen Object, the Object Properties for the Object's Command Dynamic is accessible (as shown to the right). Click on the **Config…** button.

Next, you will need to do two things:

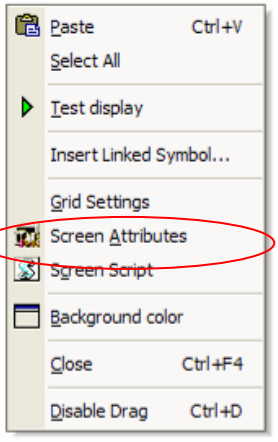
1) Select the event that will trigger the execution of the IWS Script (e.g. On Down, On While, On Up, etc.)
2) From the **Type** combo-box, select **Built-in Language**

A Configuration Worksheet will appear that you can enter the IWS Scripting Language into. The **Expression** field defines one or more Tags, Constants, Operations and/or calls to IWS built-in Functions to be executed). The **Tag** field is the same as the **Tag Name** field in the **Math Worksheet** (i.e. this is where the name of the Tag that stores the result of the Expression is defined). When finished, click on the **OK** button and close the Object Properties dialog box.

IWS Scripting Language entered into the Configuration Worksheet will be executed when the Command Dynamic event (On Down, etc.) occurs.

Screen Logic

An IWS Scripting Language code segment can be executed when a Screen is first opened, when it closes, or run while it is opened. This allows for some programmatic action(s) to be taken based on the Screen status.



To implement Screen Logic, you need to first open up an existing Screen (from the **Screens** folder in the **Graphics** Workspace). Position your mouse cursor to a blank area of the Screen and click the right mouse button. A list of options as shown to the right will appear. Select (left mouse click) on the **Screen Attributes** option.

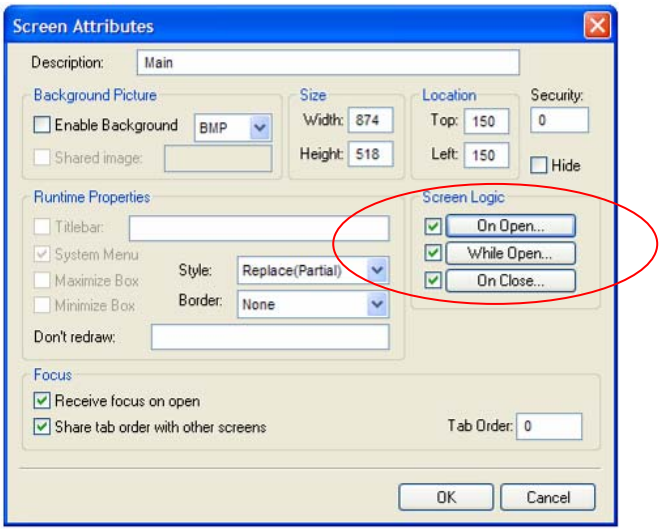
Next, a **Screen Attributes** dialog box will appear. Check the check boxes for the

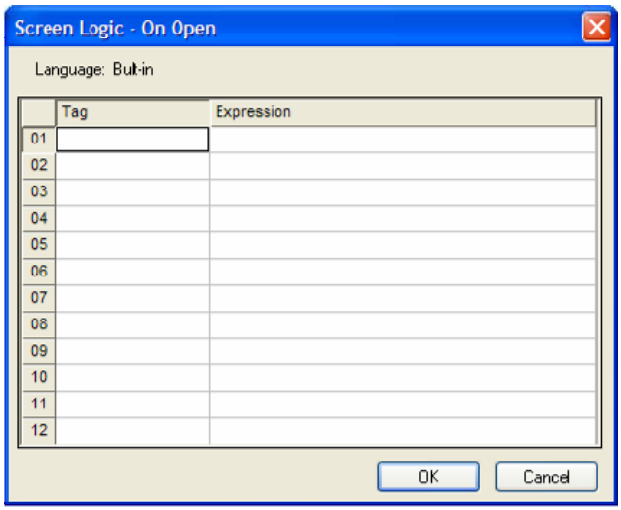
Screen Logic events you want to configure. Options are: On Open, While Open, and On Close.

To enter the IWS Script Language code for the event you selected, left click on the **On Open, While Open**, or **On Close** button(s) next the check boxes.

When any of the **On Open, While Open,** or **On Close** buttons are pressed, a **Screen Logic** Configuration Worksheet will open. You can enter a IWS Scripting Language code segment here, but as with the Command Dynamic, you are limited to a total of 12 lines.

The **Expression** field defines one or more Tags, Constants, Operations and/or calls to IWS built-in Functions to be executed). The **Tag** field is the same as the **Tag Name** field in the **Math Worksheet** (i.e. this is where the name of the Tag that stores the result of the Expression is defined). When finished, click on the **OK** button and close the Screen Attributes dialog box by clicking on the **OK** button.





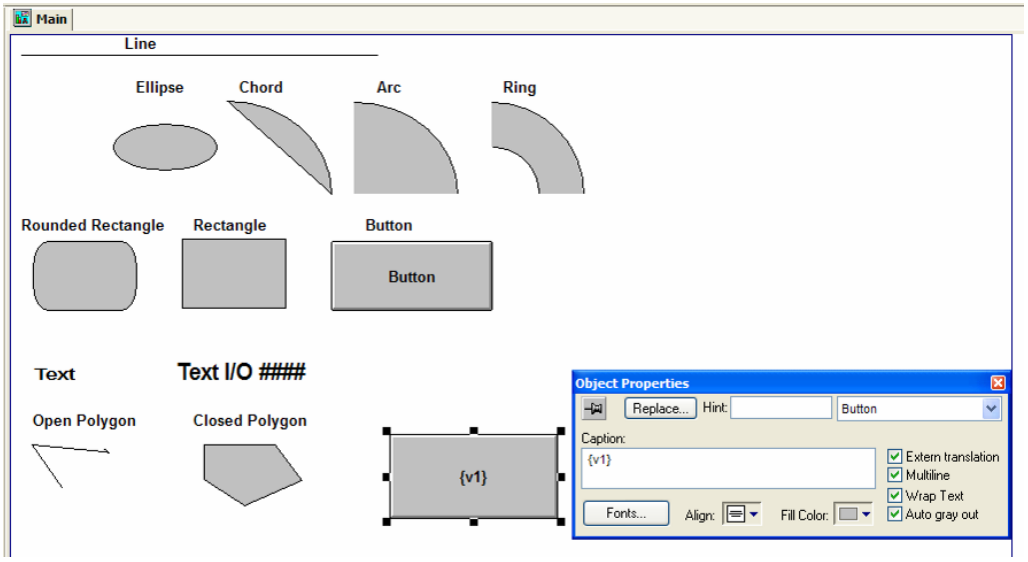
Scheduler

The Scheduler is a Task found in the **Tasks** Workspace. Its purpose is to perform some action based on an event, such as elapsed time, a date, or a tag trigger. The IWS Scripting Language can be used with the Scheduler. However, this topic will be discussed in a later section with the Scheduler.

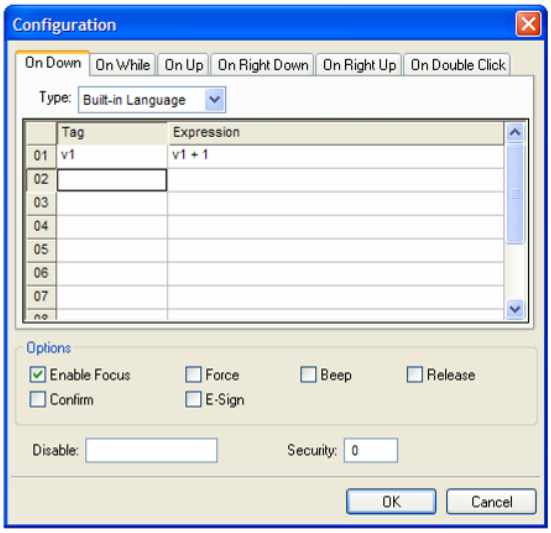
Exercise: Using IWS Scripting Language with Screens

In this Exercise, we will create a Button Object and attach a Command Dynamic property to it. Then, we will use the IWS Scripting Language with the Command Dynamic to perform count up when the left mouse button is clicked, and count down when the right mouse button is clicked. The IWS Scripting Language will be used to set a Tag when the Screen opens. This application builds on previous Exercises.

- ☐ Open the IWS development environment and open your Application (**ClassTutorial**).
- ☐ In the **Graphics** Workspace, open the **Screens** folder.
- ☐ Double click on **Main.scr**
- ☐ Select the **Button** Tool from the **Static Objects** Toolbar.
- ☐ With the **Button** Tool selected, draw a Button Object on the Screen. (Click, Drag & Release).
- ☐ Double click on the Button Object to access the Object Properties.
- ☐ In the Caption field of the Button Object, type **{v1}** into the Caption field as shown

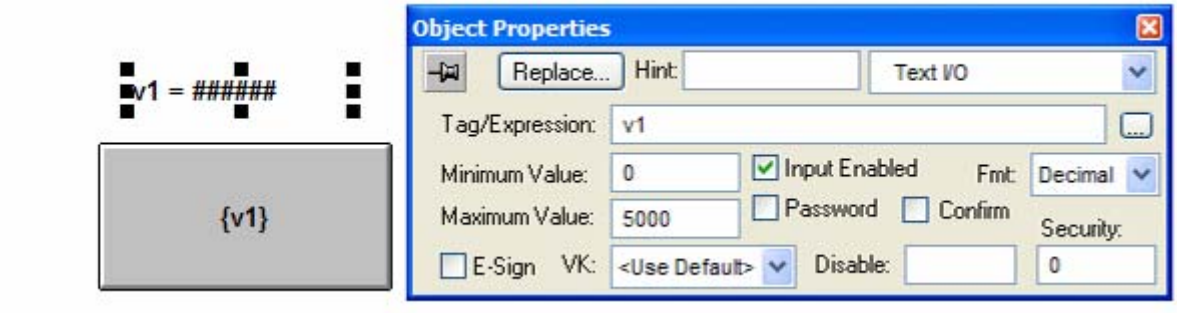


- ☐ With the Button Object selected (it is ok if the Object Properties dialog box is still open), select the **Command Dynamic** from the **Dynamic Properties** Toolbar.
- ☐ In the Command Dynamic Object Propertied dialog box, click on the **Config…** button.
- ☐ Make sure the **On Down** event is selected
- ☐ Make sure the **Type** is **Built-in Language**
- ☐ In line 1, type **v1** in the **Tag** field
- ☐ In line 1, type **v1+1** in the **Expression** field
- ☐ Select the **On-Right Down** event
- ☐ Make sure the **Type** is **Built-in Language**
- ☐ In line 1, type **v1** in the **Tag** field
- ☐ In line 1, type **v1-1** in the **Expression** field
- ☐ Click the **OK** button to close the Configuration dialog box.
- ☐ Close the Object Properties dialog box



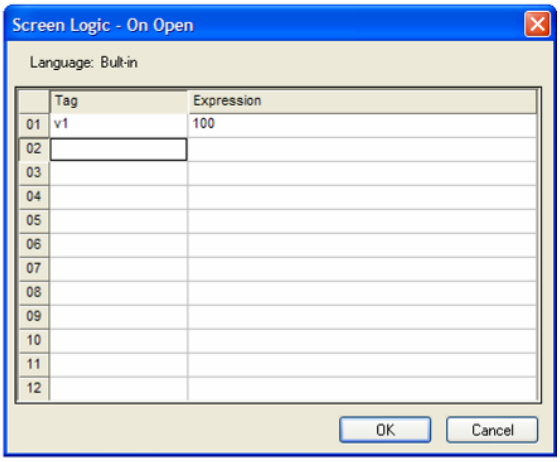
Next, we will add a Text Object with Text I/O properties

- ☐ Directly above the Button Object, add a **Text** Object by selecting the **Text** Tool from the **Static Objects** Toolbar.
- ☐ Position the cross-hair above the Button Object and type **v1 = ######**
- ☐ Click the left mouse button once. The Text Object should be selected
- ☐ With the Text Object selected, select the **Text I/O** Dynamic Property
- ☐ In the **Tag/Expression** field, type v1 (this is the Tag v1).
- ☐ Check the **Input Enabled** check box
- ☐ Enter **0** into the **Minimum Value** field
- ☐ Enter **5000** into the **Maximum Value** field
- ☐ Close the Object Properties dialog box



Finally, we will use the IWS Scripting Language in Screen Logic to provide an initial value for the Tag **v1**.

- ☐ Position the mouse cursor In a blank area of the Screen and click the right mouse button.
- ☐ Select the **Screen Attributes** option from the list of Options.
- ☐ In the Screen Attributes dialog box, check the **On Open** check box in the **Screen Logic** section.
- ☐ Click on the **On Open** button
- ☐ A **Screen Logic** Configuration dialog will open. In Line 1, type **v1** in the **Tag** field, and type **100** in the **Expression** field.
- ☐ Click the **OK** button to close the Screen Logic Configuration.
- ☐ Click the **OK** button to close the Screen Attributes dialog.
- ☐ Close any open Object Properties dialog boxes
- ☐ Save the Screen **Main**
- ☐ Close the Screen **Main**
- ☐ Run the Application and verify the operation of the button that was just created.
- ☐ When finished, stop the Application runtime.

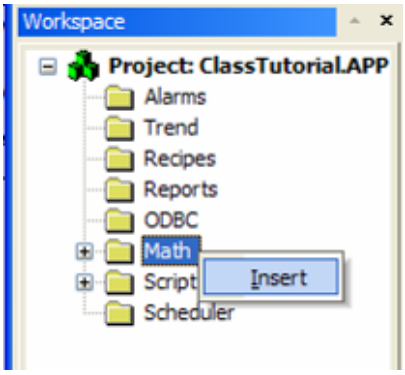


Exercise: IWS Scripting Language & Math Worksheets

In this Exercise, we will create a Button Object and attach a Command Dynamic property to it. Then, we will use the IWS Scripting Language with the Command Dynamic to perform count up when the left mouse button is clicked, and count down when the right mouse button is clicked. This application builds on previous Exercises.

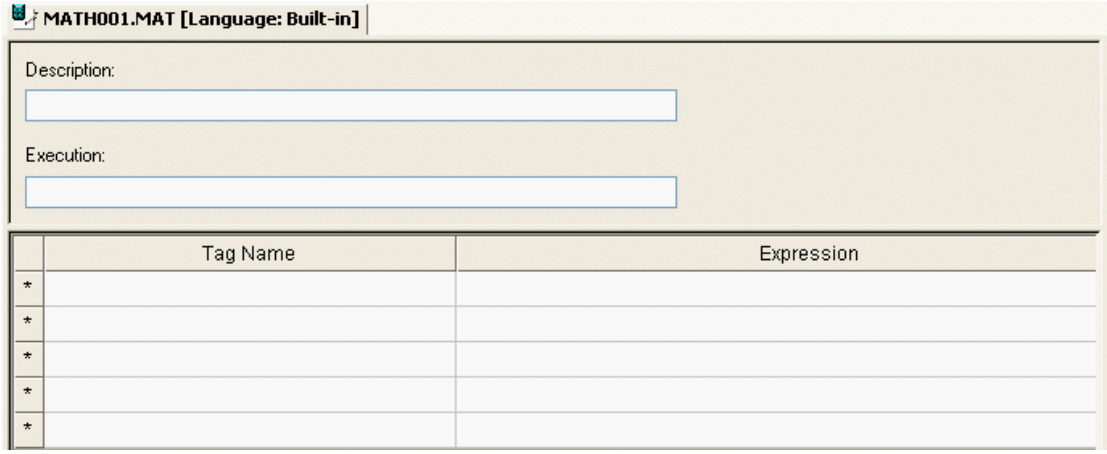
Use the following procedure to configure a Math worksheet:

- ☐ In the **Workspace**, select the **Tasks** tab and then right-click on the **Math** folder.
- ☐ When the pop-up menu displays, select the **Insert** option:



Opening a Math Worksheet

A blank **Math** worksheet displays:



Blank Math Worksheet

The header portion of the worksheet contains the following fields:

Description field: Provides space for an *optional* description of the worksheet.

Execution field: Controls the math execution. You can type a full expression here, a simple condition, a tag name, or a value. The math will execute while it is TRUE.

The body section consists of the following fields:

Tag Name: Receives the result of the expression in the **Expression** column.

Expression: Any expression written using the built-in IWS scripting language.

- In the **Description** field, type **Field Process Simulator**.
- In the **Execution** field, type **1**. This enables continuous execution of this Math worksheet; the value 1 is always a TRUE condition.
- Next, to generate simulated process data that can be used by the various screens, we must set the following variables:

  Valve status (according to the command given)

  Temperature, pressure, and level for three tanks

  Simulated valve status (just transfer the value from the command tags to the status tags).

  Simulated temperature and pressure properties for each tank (configure these properties using *sine* and *cosine* trigonometric functions)

  Simulated level properties for each tank (remember that both the Fill and Empty valves allow for the same flow).

- Given this information, complete the body of the *Math* worksheet as follows:

Description:

Field Process Simulator

Execution:

1

	Tag Name	Expression
1	ValveEmptyState[1]	ValveEmptyCommand[1]
2	ValveEmptyState[2]	ValveEmptyCommand[2]
3	ValveEmptyState[3]	ValveEmptyCommand[3]
4	ValveFillState[1]	ValveFillCommand[1]
5	ValveFillState[2]	ValveFillCommand[2]
6	ValveFillState[3]	ValveFillCommand[3]
7	tank[1].Temperature	(sin ((Second/10) *pi ()) +1) *50
8	tank[2].Temperature	(sin ((Second/20) *pi ()) +1) *50
9	tank[3].Temperature	(sin ((Second/30) *pi ()) +1) *50
10	tank[1].Pressure	(cos ((Second/10) *pi ()) +1) *50
11	tank[2].Pressure	(cos ((Second/20) *pi ()) +1) *50
12	tank[3].Pressure	(cos ((Second/30) *pi ()) +1) *50
13	J	For (1,3,1)
14	tank[J].level	if (ValveEmptyState[J]> ValveFillState[J] AND tank[J].Level>0, tank[J].Level-1)
15	tank[J].level	if (ValveEmptyState[J]< ValveFillState[J] AND tank[J].Level<100, tank[J].Level+1)
16	Next	
*		

Note:

- In the preceding example, the Math worksheet is executed continuously. In a real world application, we strongly recommend that the execution of each Math worksheet be carefully controlled to improve system performance.

IWS Built-in Functions

IWS provides a large number of built-in Functions that can be called from the IWS Scripting Language and from VBScript. It is beyond the scope of these materials to cover all these Functions and the reader is referred to Appendix A of the IWS Users Manual (Users Guide and Technical Reference Manual) for a complete listing and description of these Functions. In this section, we will cover a few of the more commonly used Functions.

The IWS built-in Functions are organized into the following categories:

- Arithmetic Functions
- Trigonometric Functions
- Logarithmic Functions
- Statistical Functions
- Logical Functions
- Looping Functions
- String Functions
- Date & Time Functions
- Opening & Closing Windows Functions
- Security Functions
- Module Activity Functions
- File Functions
- Graphics & Printing Functions
- Multimedia Functions
- Translation Function
- System Information Functions
- Database (Tags Database) Access Functions
- ODBC Functions
- Email Functions
- FTP Functions
- Dial-Up Functions
- ActiveX Functions
- Event Logger Functions
- Log Message Functions

IWS Built-in Function Argument Syntax

The IWS built-in Functions generally contain one or more arguments enclosed in parentheses following the Function name, although this is not always the case. In the documentation contained in Appendix A of the IWS Users Manual, these arguments follow the syntax below:

- **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

where **Name** is the Tag name.

With a few built-in Functions, the Tag name must be enclosed in double quotations. In several cases, either a Tag name or an Expression can be used. .

IWS Built-in Function Usage

You will need to check the IWS Users Manual Appendix A documentation (or the On-Line Help System) to determine the environment that any particular IWS built-in Function will support. The options are Windows XP (includes NT, 2000, Server 2003 & Vista), Windows CE, and Web Thin Client. Not all built-in Functions are supported under every runtime environment.

IWS built-in Functions are used in the Expression field of Math Worksheets, Command Dynamics, Screen Logic and the Scheduler. With VBScript, the built-in Functions are simply functions that are used in a statement (proceeded by the **$** operator to let the VBScript interpreter know that the function being called is an IWS built-in function).

Commonly used IWS Built-in Functions

Some commonly used IWS built-in Functions are show below. Of course, the built-in Functions that you use will depend on your application

Logical Functions

Function **If()**
Execution Synchronous
Environments Win XP, WinCE, Web Thin Client
Description A logical If statement that is used with an IWS Scripting Language code segment.
Usage If(**numExpression, numThen, optNumElse**)
Argument(s) **numExpression**
A Tag or Expression used as the condition in the If Function
numThen
A Tag or Expression used if the condition is logically True
optNumElse
A Tag or Expression used if the condition if logically False
Return Value numThen
Used if the numExpression is logically True
optNumElse
Used if the numExpression is logically False
No value returned
If numExpression is logically False and there is no optnumElse argument
Remarks You can nest arguments by using additional **If** functions (see example below). You can use Logical Operators in the numExpression argument. While the **If** Function does return values, a Tag in the return field (**Tag Name** field) is not required.
Example(s)

Tag Name	Expression
Tag	If (10 > 2, 5, 4) //returns a value of 5 since expression is True
	If (v1>10, v2+5, v2-5) //If v1 >10 then add 5 to v2, else subtract 5
Tag	If (v1>v2, If (v3<v2, 1, 2), 3)
Tag	If (v1>v2 AND v3=5, 1,0)

Looping Functions

Function **For() and Next**
Execution Synchronous
Environments Win XP, WinCE
Description Implements the **For …Next** loop within an IWS Script Language code segment. The Loop begins with the **For** statement and ends with the **Next** statement.
Usage For (**numInitialValue, numFinalValue, numStep**) ….. **Next**
Argument(s) **numInitialValue**
A Numerical Tag or Expression containing the initial value of the **For** loop
numFinalValue
A Numerical Tag or Expression containing the final value of the **For** loop. The loop will continue until the value exceeds the **numFinalValue** value
numStep
A Numeric Tag or Expression containing the step (increment) of the **For** loop.
Return Value Returns the step on which the loop is currently runing
Remarks The **Next** statement works with the previous **For()** statement. For every **For()** statement, there must be a **Next** statement.
Example(s)

Tag Name	Expression
Tag	For (0, 100, 2)
	//statements go here
Next	

Opening and Closing Windows Functions

Function **Close()**
Execution Asynchronous
Environments Win XP
Description Closes an Open Screen.
Usage Close (strScreen)
Argument(s) **strScreen**
A string Tag or Expression containing the name of the Screen to close
Return Value None
Remarks You do not need do use .scr extension. If you Open a Screen with the Replace style, the Open function automatically closes the Screens with Replace or Popup attributes that are overlapped by the new Screen. In this case, you do not need to call the Close Function.
Example(s)

Tag Name	Expression
	Close (“main”)
	Close (strScreenName) //strScreenName is a string Tag

Function **Open()**
Execution Asynchronous
Environments Win XP, WinCE, Web Thin Client
Description Opens a specified Screen
Usage Open(**strScreen, optNumX1, optNumy1, optNumX2, optNumY2**)
Argument(s) **strScreen**
A string Tag or Expression containing the name of the Screen to be opened
optNumX1
Optional Integer Tag or Expression containing the X coordinate for the upper-left corner of the Screen (in pixels)
optNumY1
Optional Integer Tag or Expression containing the Y coordinate for the upper-left corner of the Screen (in pixels)
optNumX2
Optional Integer Tag or Expression containing the X coordinate for the lower-right corner of the Screen (in pixels)
optNumY2
Optional Integer Tag or Expression containing the Y coordinate for the lower-right corner of the Screen (in pixels)
Return Value 0 if the Function executed successfully
1 if the Function did not execute successfully
Remarks Some Web Servers are case sensitive. **Only use lowercase letters for Screen names if you are going to use Web Thin Client(s).**
Example(s)

Tag Name	Expression
	Open (“main”)
	Open (“alarms”, 50, 50, 550. 530)

Function **OpenPrevious()**
Execution Asynchronous
Environments Win XP, WinCE, Web Thin Client
Description Re-opens the last Screen that was closed.
Usage OpenPrevious(**optNumX1, optNumy1, optNumX2, optNumY2**)
Argument(s) **optnumX1**
Optional Integer Tag or Expression containing the X coordinate for the upper-left corner of the Screen (in pixels)
optnumY1
Optional Integer Tag or Expression containing the Y coordinate for the upper-left corner of the Screen (in pixels)
optnumX2
Optional Integer Tag or Expression containing the X coordinate for the lower-right corner of the Screen (in pixels)
optnumY2
Optional Integer Tag or Expression containing the Y coordinate for the lower-right corner of the Screen (in pixels)
Return Value 0 if the Function executed successfully
1 if the Function did not execute successfully
Remarks
Example(s)

Tag Name	Expression
	OpenPrevious()
	OpenPrevious (50, 50, 550. 530)

Module Activity Functions

Function **ExitWindows()**
Execution Asynchronous
Environments Win XP
Description Closes the Windows OS in a specified manner.
Usage ExitWindows(**numExitCode**)
Argument(s) **numExitCode**
Integer tag containing a number specifying how the Windows OS will be exited.
0 = Reboot Windows
1 = Log Off Windows
2 = Shutdown Windows
Return Value No returned value
Remarks
Example(s)

Tag Name	Expression
	ExitWindows(0)

Function **LogOff()**
Execution Asynchronous
Environments Win XP, Win CE, Web Thin Client
Description Logs off the current User and logs on as the Guest user
Usage Logoff()
Argument(s) none
Return Value No returned value
Remarks
Example(s)

Tag Name	Expression
	Logoff()

Function **LogOn()**
Execution Asynchronous
Environments Win XP, Win CE, Web Thin Client
Description Logs on as the specified User using the specified Password.
Usage LogOn (**optStrUser, optStrPassword**)
Argument(s) **optStrUser**
Optional string Tag or Expression containing the name of the User to log on.
optStrPassword
Optional string Tag or Expression containing the User's log on Password
Return Value No returned value
Remarks If no User name is specified, a LogOn dialog box will be opened. If a Password is enabled but not specified in the LogOn Function, a LogOn dialog box will be opened.
Example(s)

Tag Name	Expression
	Logon()
	Logon ("Albert", "EMC2")

Function **Recipe()**
Execution Asynchronous
Environments Win XP, Win CE, Web Thin Client
Description Activates a specified Recipe Function
Usage Recipe (**strFunction**)
Argument(s) **strFunction**
String Tag or Expression specifying the Operation to be performed and the Recipe Worksheet to be used in the Operation. The format of the **strFunction** is "[Operation] : [Recipe Worksheet]".
Valid Operations are:
Save :Saves data to a Recipe data file
Load : Loads data from a Recipe data file
Delete : Deletes a Recipe data file
Init : Initializes a Recipe data file with all the Tags being set to 0
Recipe Worksheet
Name of the Recipe Worksheet to be used
Return Value **0** : No Error
1 : The **strFunction** Tag or Expression is numeric (not a String)
2 : Expression does not contain a colon **":"**
3 : Invalid Operation
4 : Recipe Task was not found (does not exist, or **Background Task** is not running)
5 : Disk Error (e.g. Disk full, Read-only File, or invalid path)
Remarks You must be running the **Background Task** to execute the Recipe Function. The **Background Task** is enabled by default, and is accessed from the Development environment by selecting from the Main Menu Bar **Project → Status.**
When the Recipe() Function is invoked on a Web Thin Client, the command is sent to the (IWS) **Server** to execute the Recipe task on the Server. Therefore, any Tag in the Tag Name field will be updated, even if the Tag has a Scope of Local (instead of set to a Scope of Server).
Example(s)

Tag Name	Expression
Tag	Recipe("Save:Recipe1")
Tag	Recipe("Load:Recipe5")

Function **Report()**
Execution Synchronous
Environments Win XP, Win CE
Description Saves a Report to a Disk File or sends a Report to the Printer
Usage Report (**strFunction, optNumOrientation**)
Argument(s) **strFunction**
String Tag or Expression specifying the Operation to be performed and the Report Sheet to be used in the Operation. The format of the **strFunction** is "[Operation] : [Report Worksheet]".
Valid Operations are:
Disk : Saves the Report to a data file
Prn : Prints the Report to a Printer
Report Worksheet
Name of the Report Worksheet to be used. File name must use a **.REP** extension
optNumOrientation
Sets the paper orientation. This argument is used only with the **Prn** operation. Valid values are:
0 = Portrait (default)
1 = Landscape
Return Value **0** : No Error
1 : The **strFunction** Tag or Expression is numeric (not a String)
2 : Expression does not contain a colon **":"**
3 : Invalid Operation
4 : Report Task was not found (does not exist, or **Background Task** not running)
5 : Disk Error (e.g. Disk full, Read-only File, or invalid path)
Remarks The **optNumOrientation** argument is not supported when using a Windows CE-based runtime platform. The Report Function does not support RTF Reports when using a Windows CE-based runtime platform.

You can use the **PrintSetup()** Function to set up the Printer.

You must be running the **Background Task** to execute the Recipe Function. The **Background Task** is enabled by default, and is accessed from the Development environment by selecting from the Main Menu Bar **Project → Status.**

Example(s)

Tag Name	Expression
Tag	Report ("Disk: Report1.rep")
Tag	Recipe ("Prn : Report2.rep", 1)

Function	**ShutDown()**
Execution	Synchronous
Environments	Win XP, Win CE
Description	Shuts down all of the active Application Modules
Usage	Shutdown ()
Argument(s)	None
Return Value	No returned values
Remarks	The **Shutdown()** Function does not close the development environment, Database Spy or LogWin.
Example(s)	

Tag Name	Expression
	Shutdown()

Function **WinExec()**
Execution Asynchronous/ Synchronous
Environments Win XP, Win CE, Web Thin Client
Description Executes a Windows Command Line (or Program)
Usage WinExec (**strCommand**, **optNumState**, **optNumSync**, **"optTagReturnOrHandle")**
Argument(s) **strCommand**

String Tag or Expression containing the Command Line to execute

optNumState

Optional numeric Tag or Constant containing an Integer that defines the initial state of a new Application. Valid values are:

- **0** : Activates but hides the Windows Program. IWS Application is still active.
- **1** : Activates and displays the Windows Program (**default)**
- **2** : Activates the Windows Program and displays it as an Icon
- **3** : Activates the Windows Program and maximizes it
- **4** : Activates and shows the Windows Program with its most recent size. IWS Application is still active
- **7** : Shows the Windows Program as an Icon. IWS Application is still active

optNumSync

Optional numeric Tag or Constant that specifies if the Function will execute in a Synchronous or Asynchronous mode. When executing Synchronously, the Function will return when the Windows command or Application finished execution. When executing Asynchronously, the Function will return immediately. To verify if a Program that was started Asynchronously has finished, you should use the **"optTagReturnOrHandle"** parameter and the **WinExecIsRunning** Function.

- **0** : Executes in Asynchronous mode (**default**)
- **1** : Executes in Synchronous mode (i.e. next line of Script will not be executed until the Function is completed its execution)

"optTagReturnOrHandle"

Optional argument that specifies an IWS Integer Tag name using a String Constant. The Tag specified in the String Constant will receive either the Return Code (if the Function is running in a Synchronous mode), or the Windows Program Handle number (if the Function is running in an Asynchronous mode). The Windows Program Handle number can be used with the **WinExecIsRunning** Function to determine whether the Windows Program is still running

Return Value **0** : Command was not executed successfully
1 : Command was executed successfully

Remarks You should specify the full path name of the Windows Program you intend to execute.

Microsoft NotePad	"C:\Windows\NotePad.exe"
Microsoft WordPad	"C:\Program Files\Windows NT\Accessories\WordPad.exe"
Microsoft Paint	"C:\Windows\System32\MSPaint.exe"
Microsoft Command	"C:\Windows\System32\cmd.exe"

Be sure to leave a space between the end of the Windows Program file name if you intend to pass any parameters the Windows Program.

Example(s)

Tag Name	Expression
s1	"C:\Windows\Notepad.exe " \\s1 is a string tag
Tag	WinExec (s1 + "Myreport.rep") \\opens report file
Tag	WinExec ("C:\Windows\System32\cmd.exe") \\opens CMD dialog
Tag	WinExec ("C:\Windows\System32\MSPaint.exe", 4)
Tag	WinExec ("C:\MyBatch.bat", 0,1,"Tag")
	//the above Function starts a batch file, runs hidden, runs synchronously, status returned in Tag

File Functions

Function **Hst2Txt()**

Execution Asynchronous

Environments Win XP, Win CE

Description Exports information from the IWS proprietary binary trend history file(s) (*.hst) into a Text file (*.txt) or CSV (*.csv) format

Usage Hst2Txt (**strStartDate, strStartTime, numDuration, numGroupNumber, optStrTargetFile, optStrSeperator, optNumMilliseconds, optStrFormat)**

Argument(s)

strStartDate
String Tag or Expression containing the start date of the data

strStartTime
String Tag or Expression containing the start time of the data

numDuration
Numeric Tag or Expression containing the duration of the data in hours

numGroupNumber
Numeric Tag or Expression containing the Trend Task Group Number

optStrTargetFile
Optional String Tag or Expression containing the path and file name of the .TXT (or .CSV) file. If omitted, the Function will create a file with the same name as the proprietary binary file but with a **.txt** file extension

optStrSeparator
Optional String Tag or String Constant containing the data separator character for the **.txt** file. If omitted, the **Tab** character (**\t**) will be used to separate the values in the text file.

optNumMilliseconds
Optional Numeric Tag or Numeric Constant. If the value is set to 0 (logical False), the text file created will not show millisecond precision on the timestamp of each history sample. Otherwise, the millisecond precision on the timestamp of each history sample will be included.

optStrFormat
Optional String Tag or Expression that specifies the order of the Month (M), Day (D) and Year (Y) for the timestamp format exported to the text file. If omitted, the default is "DMY" format. Valid values are:
"DMY" : Day, Month, Year
"MDY" : Month, Day, Year
"YMD" : Year, Month, Day

Return Value
-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 HstTxt Functin has not yet completed

Remarks Use the comma (,) character as the **optStrSeparator** argument to create a **CSV** file (Comma Separated Values). This will allow the Trend history data to be converted to a file format that can be opened with Microsoft Excel.

Example(s)

Tag Name	Expression
Tag	Hst2Txt ("01/05/2007", "07:00:00", 1.5, 2, "C:\IWS\data1.csv", ",")
Tag	Hst2Txt ("01/01/2007", "07:00:00", 0.2, 1, "C:\IWS\data2.txt", "\t", 0, "MDY")

Function **Print()**
Execution Asynchronous
Environments Win XP, Win CE
Description Prints a text file
Usage Print (**strFilePath, optNumOrientation)**
Argument(s) **strFilePath**
String Tag or Expression specifying the path and name of the text file that will be printed.
optNumOrientation
Optional Numeric Tag or Constant specifying the orientation of the paper. This function is not supported under Windows CE.
0 : Portrait (**default**)
1 : Landscape
Return Value None
Remarks This Function can only be used to print text files. Any other information contained in the file (e.g. Pictures, Binary Data) cannot be printed using this Function.
Example(s)

Tag Name	Expression
	Print (“C:\IWS\data2.txt”)
	Print (“C:\IWS\data2.txt”, 1)

Function **RDFileN()**
Execution Synchronous
Environments Win XP, Win CE, Web Thin Client
Description Launches a File Browser window allowing you to select a file.
Usage RDFileN (**“tagSelectedFile”, strSearchPath, strMask, optNumChangeDir)**
Argument(s) **“tagSelectedFile”**
Name of a String Tag that will receive the name (and path) of the selected file. The Tag name must be enclosed in double quotes and must be a valid Tag.
strSearchPath
String Tag or Expression containing the file path (directory) to search
strMask
String Tag or Expression containing the mask used to filter the files.
optNumChangeDir
Optional Numeric Tag or Constant that specifies whether the operator will be able to change the browsing directory. If this argument is missing or set to 1 (logical True), then the window opened by this Function will allow the operator to navigate to different directories. If this argument is set to 0 (logical False), then the window will be restricted to the path (directory) specified by the **strSearchPath** argument.
Return Value **0** : Function executed successfully
1 : One of the arguments that is supposed to be a String is not a String
2 : The first argument contains an invalid Tag name
3 : The user canceled the operation
Remarks This Function is helpful when selecting a file (e.g. for a Batch operation)
Example(s)

Tag Name	Expression
Tag	RDFileN(“s2”, “C:\iws\”, “*.txt”,1)

Translation Functions

Function **SetTranslationFile()**

Execution Synchronous

Environments Win XP, Win CE, Web Thin Client

Description Sets the active translation file and translates all enabled text within the Application.

Usage SetTranslationFile (**strFileName, optStrColumnName)**

Argument(s) **strFileName**

String Tag or Expression containing the name of the Translation File to be used..

optStrColumnName

Optional String Tag or Expression containing the name of the column from the Translation file to be used to translate all enabled text within the Application. When this argument is omitted, the second column for the translation file will be used by default.

Return Value **0** : Function executed successfully

1 : Invalid number of arguments

2 : Wrong argument type

3 : Translation file could not be opened or could not be found

Remarks The Translation Option check box must be enabled (selected) for this Function to work. The Translation Option check box can be found in the Main Menu Bar in **Project → Settings → Options**. The older style translation files supported only one translation language and the files ended in a **.tra** file extension. Newer style translation files support multiple languages per file and are usually **.csv** file types (comma separated values).

Example(s)

Tag Name	Expression
Tag	SetTranslationFile(“MyTranslation.csv”, “German”
Tag	SetTranslationFIle(“English.tra”)

System Information Functions

Function **GetAppPath()**
Execution Synchronous
Environments Win XP, Win CE
Description Returns the directory (path) of the current Application
Usage GetAppPath()
Argument(s) None
Return Value Returns the directory (path) of the current Application as a String
Remarks This Function returns the current directory (path) of the Application, including a "\" at the end of the path. Must include the empty parentheses.
Example(s)

Tag Name	Expression
Tag	GetAppPath()

Function **GetCompterIP()**
Execution Synchronous
Environments Win XP, Win CE, Web Thin Client
Description Returns the first IP address of the local station.
Usage GetComputerIP()
Argument(s) None
Return Value Returns the first IP address of the local station as a String.
Remarks Function must include the empty parentheses.
Example(s)

Tag Name	Expression
Tag	GetComputerIP()

Function **GetComputerName()**
Execution Synchronous
Environments Win XP, Web Thin Client
Description Returns the local Computer name
Usage GetComputerName()
Argument(s) None
Return Value Returns the name of the local Computer as a String.
Remarks Function must include the empty parentheses.
Example(s)

Tag Name	Expression
Tag	GetComputerName()

Event Logger Functions

Function **SendEvent()**
Execution Synchronous
Environments Win XP, Windows CE, Web Thin Client
Description Used to send an Event Message and Comment to the Event Log file.
Usage SendEvent (**strEvent, optBooFlag, optStrComment)**
Argument(s) **strEvent**
String Tag or Expression containing the text of the Event Message to be saved in the Event Log file.
optBooFlag
Optional Boolean Tag or Constant used to indicate if the **SendEvent** Function has a Comment associated with the Event Message. If **0** (logical False), there is no Comment (default). Otherwise, the is a comment associated with the Event Message.
optStrComment
String Tag or Expression containing the Comment for the Event Message saved in the Event Log file. If omitted, the User is prompted with a Standard Dialog box where the comment can be typed.
Return Value **0** : Function executed successfully
1 : Event Logger is disabled in the Event Settings
2 : Event Logger is enabled but Custom Messages are disabled in the Event Settings dialog.
Example(s)

Tag Name	Expression
Tag	SendEvent(“Valve Open – Line 1”)
Tag	SendEvent(“High Temp – Temp is” + strTemp) //temp is string val
Tag	SendEvent(“Valve Open – Line 1”, 1) //will prompt for comments
Tag	SendEvent(“Valve Open – Line 1”, 1, “Stuck Valve”)

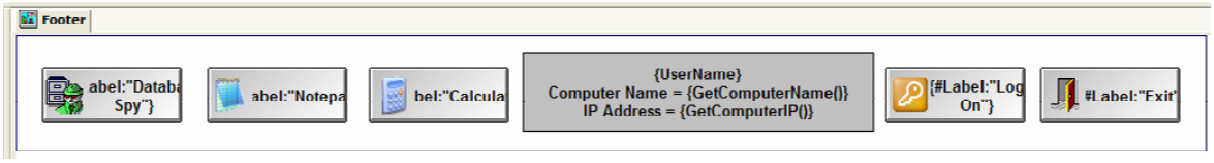
Log Message Functions

Function **Trace()**
Description Displays the contents of **strOutputMessage** in the LogWin window
Execution Synchronous
Environments Win XP, WinCE
Usage Trace(strOutputMessage)
Argument(s) The strOutputMessage argument can be a string constant, a string tag or an Expression.
Return Value None
Remarks You must check the **Trace Messages** check box in the Log Settings **Log Options** for the Log Output Window for Trace Messages to be displayed.
The **Trace** Function is a good diagnostic tool for IWS Scripting or with VBScript.
Example(s)

Tag Name	Expression
	Trace (“At stage 3”)
	Trace (“The date is “ + Date)
	Trace (“Tag v1 = “ + v1)

Exercise: Using the IWS Built-in Functions

In this Exercise, we will add some buttons from the Symbol Library that call IWS Built-In Functions, as well as retrieve the current User Name, Computer Name and Computer IP address. These will be added to the **Footer.scr** Screen and will be displayed at runtime.



- ☐ Open the **Footer.scr** Screen
- ☐ From the **Buttons** folder in the Symbol Library, add the following Buttons to the Screen
  - **button_DatabaseSpy** (opens the Database Spy at runtime)
  - **button_Notepad** (open Notepad at runtime)
  - **button_Calculator** (opens Calculator at runtime)
  - **button_logon2** (opens the security logon dialog box at runtime)
  - **button_exit** (will shutdown the Application, if confirmed)
- ☐ Using the **Rectangle** tool in the Static Object Toolbar, add a Rectangle Object to the Screen
- ☐ Double click on the Rectangle Object to open the Object Properties dialog box
- ☐ Click on the **Caption…** button
- ☐ In the Caption field, type the following
  {UserName}
  Computer Name = {GetComputerName()}
  IP Address = {GetComputerIP()}
- ☐ Click **OK**, and close the Object Properties dialog box
- ☐ Save and Close the Footer Screen

Note:
- In the Rectangle Object Caption field, Tags and IWS Built-in Functions must be enclosed in curly braces **{ }**
- **UserName** is an IWS internal tag that contains the name of the User currently logged on.

Notes

Chapter 11. VBScript

Visual Basic Script Language (VBScript) is one of Microsoft's scripting languages that is commonly associated with Server-side and Client-side web applications. However, Microsoft has opened up VBScript to developers and now VBScript can be found in a variety of applications. InduSoft has standardized on VBScript since it provides a significant subset of Microsoft Visual Basic's functionality, and VBScript supports all of Microsoft's operating system platforms including Windows CE, unlike VBA (Visual Basic for Applications) which cannot support the Windows CE runtime environment.

VBScript is a programming language that is often viewed as a dialect of VBA (Visual Basic for Applications), although it is really its own language. The VBScript language attempts to balance flexibility, capability and ease of use. VBA is a subset of Visual Basic that was developed to automate Microsoft Office applications, whereas VBScript was originally developed to support Server-side and Client-side web applications. Although VBScript and VBA provide many of the same features, there are some differences between them, primarily due to the applications they were each developed to support.

Using VBScript With InduSoft HMI/SCADA Applications

InduSoft Web Studio (IWS) supports both the IWS Scripting Language (proprietary Scripting Language) using one or more Math worksheets, as well as VBScript (IWS Version 6.1 or later). Developers can use either scripting language or a combination of both. VBScript code is placed in one of several modules, based on the functionality to be performed and the scope of the code and its variables. This subject is covered more completely in the *VBScript Configuration and Operation in IWS* section.

Examples of how VBScript can be used:

- Execute a logic sequence or routine when opening or closing a screen, or while the screen is open
- Execute a logic sequence in the background
- Run a simple VBScipt code segment based on an IWS object's command dynamic
- Interaction with IWS Tags and control of IWS built-in functions
- Manipulation of ActiveX Controls and ActiveX Control event handler
- Simple file I/O (e.g. text files)
- Database interfaces (e.g. via ADO.NET), especially where use of SQL is required
- Interface to Windows Management Instrumentation (WMI) and Web Services (via WSDL)
- Interface to Microsoft Office applications (e.g. Excel, Access, Word) and Microsoft Office components via OLE Automation
- Run on a Web Thin Client

Where you should use IWS instead of VBScript

- User Interface. IWS does not support Windows Scripting, which typically provides the User Interface for VBScript via Forms.
- Device I/O (e.g. PLC communications). VBScript does not directly support serial or network communications. However, ActiveX Controls can be used.

IWS implements Visual Basic Script Edition 5.6 or higher, and functions as the "host" for VBScript. IWS provides an integrated development environment where the HMI/SCADA application developer can take advantage of the functionality and ease of use of VBScript, yet have access to all IWS tags and all built-in functions directly from VBScript. The diagram below illustrates the IWS architecture. Since VBScript is an interpreted language, the VBScript Engine parses the language at runtime and executes commands subject to limitations placed by the VBScript Host. InduSoft allows VBScript code to be located several areas in an IWS application:

- **Global Procedures**. This is an area for subroutines and functions that can be called by any other VBScript routine, or by a built-in IWS function (requires IWS Version 6.1 Service Pack 1 or later).
- **Graphic Script**. Code in this area gets executed whenever any graphics (screens) are active.
- **Screen Script**. This is where code is executed when an individual screen is active.
- **Command Dynamic**. When an object has a Command Dynamic, one option is to run VBScript code.
- **ActiveX Events**. A VBScript code segment can be run based on an ActiveX event
- **Background Task**. VBScript code can be running as a background task. One or more VBScript groups are supported, allowing conditional processing of the various VBScript background tasks.

In a Web Thin Client configuration, VBScripts associated with a screen can run either on the workstation runtime display or on a Web Thin Client station running Microsoft Internet Explorer. The VBScript routines that can execute on a Web Thin Client include those located in a Screen Script, a Command Dynamic, and an ActiveX Event. Since VBScript runs on all Microsoft operating system platforms, there are no limitations to VBScript running on any Microsoft compatible platform.

VBScript Limitations in IWS

Microsoft initially developed VBScript to work with websites (web pages). In the web server environment, VBScript was designed to work with the Windows Scripting host and ASP, which provide file access and form generation. On the web client side, VBScript was designed to work with Microsoft Internet Explorer using HTML and DHTML, which provide display generation. So as a result of the initial design goals, VBScript does not have much in the way of built-in language support for Forms, File I/O, Communications or direct Printing control. Additionally, IWS has its own built-in web server and does not use ASP.

By using IWS built-in functions, ActiveX controls and Microsoft Office Applications (or components), there are several methods for workarounds to these limitations as well as to extend VBScript's capability.

The following are some of VBScript's limitations and workarounds.

Item	VBScript	Workarounds
Forms	Does not support	Use IWS objects for user interface, pass parameters to IWS. Can also use ActiveX Controls.
File I/O	Limited support directly	Use Scripting Objects and/or IWS built-in functions. Can also use ActiveX Controls.
Communications	Does not directly support	Use IWS built-in functions or 3rd party ActiveX controls
Printing	Does not directly support	Use Microsoft Office Applications or IWS built-in functions
Charting/Graphing	Does not directly support	Use IWS trending, Microsoft Office Applications, Microsoft Office Components, or 3rd party ActiveX controls
DDE	Does not support	Supported in IWS built-in commands (not under Windows CE).

Differences between VBScript and VBA

Since other HMI/SCADA products support VBA, it might be worth highlighting some of the key differences between VBScript and VBA. For HMI/SCADA applications, these differences are relatively minor. However, VBScript support for the Windows CE operating system is a major differentiator between the two products. For additional details or a complete listing of the differences, please reference the MSDN website at http://msdn.microsoft.com.

Key differences between VBScript vs. VBA

Item	VBA	VBScript
Primary Purpose	Automation of MS Office Applications	Automation of Web Services
Support for Windows CE	No	Yes
Data Types	Stronger Type Declaration. Many data types supported. (e.g. String, Integer, Date, Boolean)	Typeless, uses Variant Type. The final data subtype will be determined at runtime based on use. Supports same data subtypes as VBA and VB (e.g. String, Integer, Date, Boolean, etc)
Dimension Statement	Dim Var as Type	Dim Var (Cannot specify Type, but it is determined at runtime based on use)
Class Block declaration	Must use separate Class Module	Class Block Declaration supported
Object	Clipboard Collection	Not supported
Object Manipulation	TypeOf	Not supported
Eval function	Not supported	Expression evaluation supported
Execute function	Not supported	Allows interpreted code to be executed on the fly.
RegExp	No	Allows creation of regular expressions
Error Handling	Several different types	Supported but more limited
Arrays	Lower bound can be <>0	Lower bound is 0
File I/O	Supported	Not directly supported but VBScript can use FileSystemObject and can access IWS built-in I/O functions
DDE	Supported	Not supported
Financial functions	Supported	Not supported
Strings	Fixed length strings	Variable length only
Debugging	Debug, Print, End, Stop	Use MsgBox or IWS built-in functions
Line labels	Supported	Not supported

VBScript Language Overview

This section contains a short summary of the VBScript Language. A more complete reference of the VBScript language can be found in VBScript Reference Manual.

VBScript Functionality

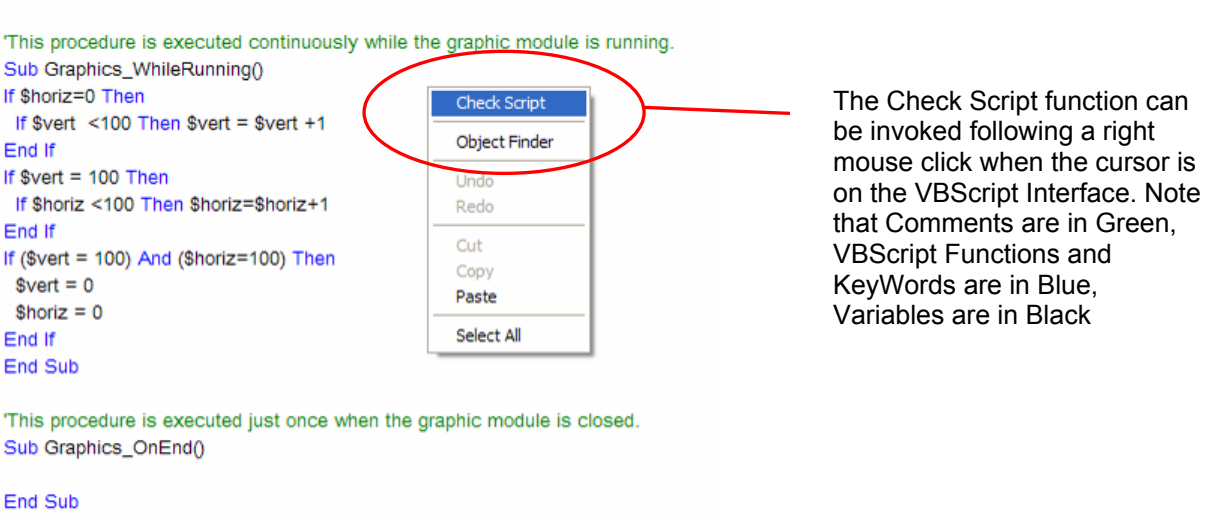
VBScript has inherited much of VB & VBA's functionality including support for math operations, string manipulation, arrays, flow control, data conversion, procedures, COM objects, and date/time functions. Since VBScript was initially designed for Web applications, direct support for file I/O and user interface functions was not included. However, in a Windows XP/2000/NT/Server 2003/Vista environment, VBScript can use the *FileSystemObject* COM object (scrrun.dll) to manipulate local files and folders.

VBScript does not support explicitly declared data types. This was eliminated to speed up the runtime performance of the VBScript Scripting Engine. All variables are type *Variant* and their subtype (e.g. Integer, Real, etc.) is determined at runtime.

VBScript Elements

There are several VBScript elements, but the most important ones are variables, constants and types. A variable is an item holding data that can change during the execution of the VBScript program. A constant is an item that holds data but cannot change during the execution of the VBScript program. The data that variables and constants hold can be classified into types.

Note that with IWS, you can check the VBScript syntax for errors by choosing the Check VBScript command (right mouse click when in a VBScript interface). VBScript is always checked when saving the Script interface.



The Check Script function can be invoked following a right mouse click when the cursor is on the VBScript Interface. Note that Comments are in Green, VBScript Functions and KeyWords are in Blue, Variables are in Black

The VBScript elements include:

- Variables (Type, Declaration, Scope)
- Constants (Explicit, Implicit)
- Keywords
- Errors (Runtime, Syntax)
- Operators
- Functions and Procedures
- Statements
- Objects and Collections

Variable Data Types and Subtypes

All variables in VBScript are a data type called **Variant**. This means that you do not (and cannot) explicitly declare the variable type. In fact, with VBScript you do not need the Dim statement to allocate storage for a variable. At runtime, the Parser in the VBScript Scripting Engine determines the **Variant** data subtype to be used. These correspond to the more traditional classifications of data types (see chart below).

Variant data subtypes

Subtype	Description
Boolean	Either *True* or *False*
Byte	Contains integer in the range 0 to 255
Currency	Floating-point number in the range -922,337,203,685,477.5808 to 922,337,203,685,477.5807
Date(Time)	Contains a number that represents a date between January 1, 100 to December 31, 9999
Double	Contains a double-precision, floating-point number in the range -1.79769313486232E308 to -4.94065645841247E-324 for negative values; 4.94065645841247E-324 to 1.79769313486232E308 for positive values
Empty	Uninitialized Variant
Error	Contains an error number used with runtime errors
Integer	Contains integer in the range -32,768 to 32,767
Long	Contains integer in the range -2,147,483,648 to 2,147,483,647
Null	A variant containing no valid data
Object	Contains an object reference
Single	Contains a single-precision, floating-point number in the range -3.402823E38 to -1.401298E-45 for negative values; 1.401298E-45 to 3.402823E38 for positive values
String	Contains a variable-length string that can be up to approximately 2 billion characters in length.

The Parser's choice of data subtype will depend on how the variable is used in a statement or function. Note that a variable's subtype can change within a code segment.

Data Subtype Identification

If it is important to determine the **Variant** data subtype used at runtime, you may use any of the three categories of functions to determine the data subtype:

- The **VarType**(*variable*) function which returns a code based on the **Variant** data subtype used
- Various **IsXxxx**(*variable*) functions which return boolean values indicating whether the variable is of a specific data subtype.
- A **TypeName**(*variable*) function which returns a string based indicating the data subtype

Example:
```
If varType(a) = vbInteger Then
    Msgbox "a is an Integer"
EndIf
```

Data Subtype Conversion

VBScript provides several functions that convert a variable from one data subtype to another. Since VBScript uses the **Variant** data type, these functions are not generally required. However, when passing data between IWS (or CEView) and VBScipt, or calling built-in IWS functions from VBScript where variables need to be put into the proper argument format, these VBScript data subtype conversion functions can be very useful.

```
Example:    a = 4.2
            b = cInt (a)                ' b is an Integer with a value of 4
```

Variable Naming Rules & Conventions

VBScript has four primary rules for naming. These are:

1. Variable names must begin with an alpha character (a..z, A...Z) or an underscore character
2. After the first character, the variable name can contain letters, digits and underscores
3. Variable names must be less than 255 characters in length
4. The variable name must be unique in the scope in which they are declared

VBScript variable names are not case sensitive. Microsoft recommends following their naming convention for variables, which puts attaches different prefixes to the variable name based on the data subtype.

Variable Scope

Variables have “scope” which defines a variable’s visibility or accessibility from one procedure (or VBScript Interface) to another, which is principally determined by where you declare the variable. Generally, when you declare a variable within a procedure, only code within that procedure can access or change the value of that variable. This is called local scope and is for 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. However, as previously noted, InduSoft enforces certain restrictions on the scope of Variables and Procedures.

VBScript Constants

VBScript supports both *explicit* and *implicit* constants. Constants should never be used as variable names.

Explicit constants are defined by the programmer. Explicit constants have a defined value which, unlike a variable, is not allowed to change during the life of the script.

Implicit constants are pre-defined by VBScript. VBScript implicit constants usually begin with a **vb** prefix. VBScript implicit constants are available to the VBScript programmer without having to define them. Other objects, such as those used by ADO.NET, also have implicit constants predefined, usually with different prefixes. However, the implicit constants for these objects may not be know to VBScript and if not, will have to be defined as an explicit constant.

VBScript defines the following categories of implicit Constants:

Intrinsic Constant Category	Intrinsic Constant Category
Color Constants	File Attribute Constants
Comparison Constants	File Input/Output Constants
Date and Time Constants	MsgBox Constants
Date Format Constants	MsgBox Function Constants
Days of Week Constants	SpecialFolder Constants
New Years Week Constants	String Constants
Error Constants	Tristate Constants
VBScript Runtime Errors	VarType Constants
VBScript Syntax Errors	Locale ID (LCID)

Declaring VBScript Variables and Constants

VBScript does not require the explicit declaration of scalar variables, i.e. those variables with only one value assigned at any given time. Arrays, Objects (except **Err**) and Constants must be declared. While it may initially be convenient not to declare variables, any typing (spelling) errors of the variable or constant names may produce unexpected results at runtime.

VBScript Keywords

VBScript has many keywords. Keywords are merely the names or symbols used with built-in VBScript functions. Keywords are reserved, i.e. they may not be used by the programmer as names of variables or constants. VBScript keywords can be grouped into categories which include:

- Constants & Literals
- Operators
- Functions
- Statements
- Objects

Operators

VBScript defines various operators that perform operations based on the **Variant** subdata type(s). Arithmetic operators are used to perform operations on two or more numbers.

Arithmetic

Symbol	Definition
+	Add
-	Subtract
*	Multiply
/	Divide
\	Integer Divide
^	Exponentiation
MOD	Modulus Division

Comparison

Symbol	Definition
<	Less than
<=	Less than or equal
>	Greater than
>=	Greater than or equal
=	Equal or assignment
<>	Not equal

Logical

Symbol	Definition
AND	And
OR, |	Or
XOR	Exclusive OR
Eqv	Equivalence
Imp	Implication
Not	NOT

String

Symbol	Definition
&, +	Concatenation

Object

Symbol	Definition
Is	Is (compare)

IWS

Symbol	Definition
$	Access to IWS Tags and Built-in functions

Operator Precedence

When several operations occur in an expression, each part is evaluated and resolved in a predetermined order called operator precedence. Parentheses can be used to override the order of precedence and force some parts of an expression to be evaluated before other parts. Operations within parentheses are always performed before those outside. Within parentheses, however, normal 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:

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 its precedence does fall in 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 only checks to determine if two object references refer to the same object.

Operator 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	&

Functions

VBScript contains a number of built-in functions (not to be confused with the Function Procedure). These functions may or may not have arguments. These functions are called in a statement and may return a result that can be assigned to a variable. VBScript's functions are grouped as follows:

Array Functions

Array Functions	Description
Array	Returns a variant containing an array
Filter	Returns a zero-based array that contains a subset of a string array based on a filter criteria
IsArray	Returns a Boolean value that indicates whether a specified variable is an array
Join	Returns a string that consists of a number of substrings in an array
LBound	Returns the smallest subscript for the indicated dimension of an array
Split	Returns a zero-based, one-dimensional array that contains a specified number of substrings
UBound	Returns the largest subscript for the indicated dimension of an array

Data Conversion Functions

Function	Description
Abs	Returns the absolute value of a specified number
Asc	Converts the first letter in a string to its ASCII decimal representation
CBool	Converts an expression to a variant of subtype Boolean
CByte	Converts an expression to a variant of subtype Byte
CCur	Converts an expression to a variant of subtype Currency
CDate	Converts a valid date and time expression to the variant of subtype Date
CDbl	Converts an expression to a variant of subtype Double
Chr	Converts the specified ANSI code to a character
CInt	Converts an expression to a variant of subtype Integer
CLng	Converts an expression to a variant of subtype Long
CSng	Converts an expression to a variant of subtype Single
CStr	Converts an expression to a variant of subtype String
Fix	Returns the integer part of a specified number
Hex	Returns the hexadecimal value of a specified number
Int	Returns the integer part of a specified number
Oct	Returns the octal value of a specified number
Round	Returns a rounded number
Sgn	Returns the integer portion of a number

Date and Time Functions

Function	Description
CDate	Converts a valid date and time expression to the variant of subtype Date
Date	Returns the current system date
DateAdd	Returns a date to which a specified time interval has been added
DateDiff	Returns the number of intervals between two dates
DatePart	Returns the specified part of a given date
DateSerial	Returns the date for a specified year, month, and day
DateValue	Returns a date
Day	Returns a number that represents the day of the month (between 1 and 31, inclusive)
FormatDateTime	Returns an expression formatted as a date or time
Hour	Returns a number that represents the hour of the day (between 0 and 23, inclusive)
IsDate	Returns a Boolean value that indicates if the evaluated expression can be converted to a date
Minute	Returns a number that represents the minute of the hour (between 0 and 59, inclusive)
Month	Returns a number that represents the month of the year (between 1 and 12, inclusive)
MonthName	Returns the name of a specified month
Now	Returns the current system date and time
Second	Returns a number that represents the second of the minute (between 0 and 59, inclusive)
Time	Returns the current system time
Timer	Returns the number of seconds since 12:00 AM
TimeSerial	Returns the time for a specific hour, minute, and second
TimeValue	Returns a time
Weekday	Returns a number that represents the day of the week (between 1 and 7, inclusive)
WeekdayName	Returns the weekday name of a specified day of the week
Year	Returns a number that represents the year

Expression Functions

Expressions	Description
Eval	Evaluates an expression and returns the result
RegExp	Provides simple regular expression support.

Format Functions

Function	Description
FormatCurrency	Returns an expression formatted as a currency value
FormatDateTime	Returns an expression formatted as a date or time
FormatNumber	Returns an expression formatted as a number
FormatPercent	Returns an expression formatted as a percentage

I/O Functions

Input/Output	Description
InputBox	Displays a prompt in a dialog box, waits for the user to input text or click a button, and returns the contents of the text box.
MsgBox	Displays a message in a dialog box, waits for the user to click a button, and returns a value indicating which button the user clicked.
LoadPicture	Returns a picture object

Math Functions

Function	Description
Abs	Returns the absolute value of a specified number
Atn	Returns the arctangent of a specified number
Cos	Returns the cosine of a specified number (angle)
Exp	Returns *e* raised to a power
Hex	Returns the hexadecimal value of a specified number
Int	Returns the integer part of a specified number
Fix	Returns the integer part of a specified number
Log	Returns the natural logarithm of a specified number
Oct	Returns the octal value of a specified number
Randomize	Initializes the random-number generator
Rnd	Returns a random number less than 1 but greater or equal to 0
Sgn	Returns an integer that indicates the sign of a specified number
Sin	Returns the sine of a specified number (angle)
Sqr	Returns the square root of a specified number
Tan	Returns the tangent of a specified number (angle)

Miscellaneous Functions

Miscellaneous	Description
GetLocale	Returns the current locale ID
RGB	Returns a whole number representing an RGB color value
SetLocale	Sets the current locale ID

Script Engine Functions

Script Engine ID	Description
ScriptEngine	Returns a string representing the scripting language in use
ScriptEngineBuildVersion	Returns the build version number of the scripting engine in use
ScriptEngineMajorVersion	Returns the major version number of the scripting engine in use
ScriptEngineMinorVersion	Returns the minor version number of the scripting engine in use

String Functions

Function	Description
InStr	Returns the position of the first occurrence of one string within another. The search begins at the first character of the string
InStrRev	Returns the position of the first occurrence of one string within another. The search begins at the last character of the string
LCase	Converts a specified string to lowercase
Left	Returns a specified number of characters from the left side of a string
Len	Returns the number of characters in a string
LTrim	Removes spaces on the left side of a string
Mid	Returns a specified number of characters from a string
Replace	Replaces a specified part of a string with another string a specified number of times
Right	Returns a specified number of characters from the right side of a string
RTrim	Removes spaces on the right side of a string
Space	Returns a string that consists of a specified number of spaces
StrComp	Compares two strings and returns a value that represents the result of the comparison
String	Returns a string that contains a repeating character of a specified length
StrReverse	Reverses a string
Trim	Removes spaces on both the left and the right side of a string
UCase	Converts a specified string to uppercase

Variant Identification Functions

Variant Function	Description
IsArray	Returns a Boolean value indicating whether a variable is an array
IsDate	Returns a Boolean value indicating whether an expression can be converted to a date
IsEmpty	Returns a Boolean value indicating whether a variable has been initialized.
IsNull	Returns a Boolean value that indicates whether an expression contains no valid data (Null).
IsNumeric	Returns a Boolean value indicating whether an expression can be evaluated as a number
IsObject	Returns a Boolean value indicating whether an expression refers to a valid Automation object.
TypeName	Returns a string that provides Variant subtype information about a variable
VarType	Returns a value indicating the subtype of a variable

Statements

VBScript statements are used to perform fundamental operations such as decision making, repetition (looping) and assignments. Statements combined with Operators are the building blocks for more complex code.

Multiple statements can appear on the same line as long as they are separated by a colon (**:**). For purposes of code readability, it is recommended to use one statement per line.

Assignment Statements

Many of VBScripts assignment statements have already been covered. For consistency purposes, they are listed here. Refer to the VBScript Users Manual for a more detailed description of their use.

Assignment Statements

Statement	Description
Const	Declares constants for use in place of literal values
Dim	Declares variables and allocates storage space
Erase	Reinitializes the elements of fixed-size arrays, deallocates dynamic-array storage space.
Option Explicit	Forces explicit declaration of all variables in the script
Private	Declares private variables and allocates storage space
Public	Declares public variables and allocates storage space
ReDim	Declare dynamic array variables, allocates or reallocates storage space at procedural level

Comment Statements

Comment statements are used to provide documentation comments with the code.

Comment Statements

Comments	Description
Rem	Includes explanatory remarks in a program
‘	Includes explanatory remarks in a program (single quote)

Control Flow Statements

By default, VBScript sequentially moves (flows) through the script from statement to statement. As is typical with virtually all high-level programming languages, control flow statements can alter this flow by branching to other code sections based upon logic conditions, inputs, errors, etc.

One of the most commonly used control flow statement is the **If..Then..Else** statement. This control flow statement takes the following format:

{simple format} **If** condition **Then** statement(s) [**Else** elsestatement(s)]

{block format}

If condition **Then**
[statement(s)]
[**ElseIf** condition-n **Then**
[elseifstatement(s)]] **. . .**
[**Else**
[elsestatement(s)]]
End If

The condition can be a boolean constant or boolean variable, or a numeric or string expression that evaluates to **True** or **False**.

Refer to the Appendix for a detail description of these functions.

Control Flow Statements

Function	Description
Do…Loop	Repeats a block of statements while a condition is True or until a condition becomes True
Execute	Executes one or more specified statements
Execute Global	Executes one or more specified statements in the global namespace of a script
Exit Do	Exit a Do Loop Function. Transfers control to the statement following the Loop statement.
Exit For	Exit a For Loop Function (For…Next or For Each…Next loop). Transfers control to the statement following the Next statement.
For...Next	Repeats a group of statements a specified number of times
For Each…Next	Repeats a group of statements for each element in an array or collection
If…Then…Else	Conditionally executes a group of statements, depending on the value of an expression
Select Case	Executes one of several groups of statements, depending on the value of an expression
While…Wend	Executes a series of statements as long as a given condition is **True**
With…End With	Executes a series of statements on a single object

Procedure Statements

There are two types of procedure statements; the **Sub** procedure and the **Function** procedure. Both of these procedure statements are intended to encapsulate a set of statements that provide functionality that can be repeatedly called, but the difference between the two is how arguments are passed and results returned.

The **Sub** procedure is a series of VBScript statements (enclosed by **Sub** and **End Sub** statements) that perform actions but don't return a value as part of the **Sub** name. A **Sub** procedure can take arguments (constants, variables, or expressions that are passed by a calling procedure). A resultant value or set of values can be returned through the arguments. If a **Sub** procedure has no arguments, its **Sub** statement must include an empty set of parentheses ().

The **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 in the **Function** name. 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**.

Procedure Statements

Function	Description
Call	Transfers control to a Sub or Function procedure
End Function	Immediately exits a Function procedure
End Sub	Immediately exits a Sub procedure
Exit Function	Exit a Function, generally as a result of a condition
Exit Sub	Exit a Subroutine, generally as a result of a condition
Function	Declares the name, arguments, and code that form the body of a Function procedure
GetRef	Associates an event handler with a specific function
Sub	Declares the name, arguments, and code that form the body of a Sub procedure (Subroutine).

VBScript Object Commands

VBScript includes several Functions and Statements that can be used to access objects, including their methods and properties. There are a large variety of objects available to VBSript, including user-defined objects, intrinsic objects and extrinsic objects.

VBScript Object Functions

Function	Description
CreateObject	Creates and returns a reference to an Automation object
GetObject	Returns a reference to an Automation object from a file
IsObject	Returns a Boolean value indicating whether an expression references a valid Automation object.

Object Statements

Statement	Description
Class	Declares the name of a class, as well as a definition of the variables, properties, and methods that comprise the class
Exit Property	Forces an exit from inside a Property Set function.
For Each…Next	Repeats a group of statements for each element in an array or a collection.
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	Sets a reference to an object
Set	Assigns an object reference to a variable or property, or associates a procedure reference with an event. Usually used to instantiate an object.

Error Handling Statements

Statement	Description
On Error	Enables or disables error-handling

Object & Collection Summary

Objects & Collections	Description
Debug	The Debug object is an intrinsic global object that can send an output to a script debugger, such as the Microsoft Script Debugger.
Dictionary	An associative array that can store any type of data. Data is accessed by a key.
Drive	An object that refers to a specific Drive
Drives	A collection of Drive objects.
Err	Contains information about the last run-time error. Accepts the Raise and Clear methods for generating and clearing run-time errors.
File	An object that refers to a specific File
Files	A collection of File objects.
FileSystemObject	An object model used to access the Windows file system
Folder	An object that refers to a specific Folder
Folders	A collection of Folder objects.
Match	Provides access to the read-only properties of a regular expression match.
Matches	Collection of regular expression Match objects.
RegExp	Provides simple regular expression support.
Submatches	A collection of regular expression submatch strings.
TextStream	An object that refers to a text File

VBScript Objects and Collections

VBScript has certain Objects and Collections that are inherent with VBScript. These include:

- Debug
- Err Object
- Match Object & Matches Collections
- Scripting Dictionary Object
- Scripting FileSystemObject
  - Drive Object
  - File Object
  - FileSystemObject Collections
  - Folder Object
- Regular Expression Object & Submatches Collection
- TextStream Object

VBScript Implicit Objects and Collections

Objects & Collections	Description
Class Object	Declares the name of a class, as well as a definition of the variables, properties, and methods that comprise the class
Debug	The Debug object is an intrinsic global object that can send an output to a script debugger, such as the Microsoft Script Debugger.
Err	Contains information about the last run-time error. Accepts the Raise and Clear methods for generating and clearing run-time errors.
Match Object	
Dictionary	An associative array that can store any type of data. Data is accessed by a key.
Matches Collection	
RegExp Object	
SubMatches Collection	

Object & Collection Summary

Objects & Collections	Description
Drive	An object that refers to a specific Drive
Drives	A collection of Drive objects.
File	An object that refers to a specific File
Files	A collection of File objects.
FileSystemObject	An object model used to access the Windows file system
Folder	An object that refers to a specific Folder
Folders	A collection of Folder objects.
Match	Provides access to the read-only properties of a regular expression match.
Matches	Collection of regular expression Match objects.
RegExp	Provides simple regular expression support.
Submatches	A collection of regular expression submatch strings.
TextStream	An object that refers to a text File

Err Object

The VBScript **Err** object contains information about run-time errors.

Err Object Properties

Properties	Description
Description	The descriptive string associated with an error.
HelpContext	A context ID for a topic in a Windows help file.
HelpFile	A fully qualified path to a Windows help file.
Number	A numeric value identifying an error.
Source	The name of the object or application that originally generated the error.

Err Object Methods

Properties	Description
Clear	Clears all property settings.
Raise	Generates a run-time error.

The properties of the **Err** object are set by the generator of an error-Visual Basic, an Automation object, or the VBScript programmer.

The default property of the **Err** object is Number. **Err.Number** contains an integer and can be used by an Automation object to return an SCODE.

When a run-time error occurs, the properties of the **Err** object are filled with information that uniquely identifies the error and information that can be used to handle it. To generate a run-time error in your code, use the VBScript Err Object Raise Method. The Err object's properties are reset to zero or zero-length strings ("") after an **On Error Resume Next** statement. The VBScript Err Object Clear Method can be used to explicitly reset **Err**.

The **Err** object is an intrinsic object with global scope-there is no need to create an instance of it in your code.

Scripting FileSystemObject

The VBScript FileSystemObject object provides access to a computer's file system

FileSystemObject Methods

Method	Description
BuildPath	Appends a name to an existing path.
CopyFile	Copies one or more files from one location to another.
CopyFolder	Recursively copies a folder from one location to another.
CreateFolder	Creates a folder.
CreateTextFile	Creates a specified file name and returns a TextStream object.
DeleteFile	Deletes a folder and its contents.
DeleteFolder	Deletes a folder and its contents.
DriveExists	Indicates the existence of a drive.
FileExists	Indicates the existence of a file.
FolderExists	Indicates the existence of a folder.
GetAbsolutePathName	Returns a complete and unambiguous path from a provided path specification.
GetBaseName	Returns the base name of a path.
GetDrive	Returns a Drive object corresponding to the drive in a path
GetDriveName	Returns a string containing the name of the drive for a path.
GetExtensionName	Returns a string containing the extension for the last component in a path.
GetFile	Returns a File object corresponding to the file in a path.
GetFileName	Returns the last component of a path that is not part of the drive specification.
GetFolder	Returns a Folder object corresponding to the folder in a specified path.
GetParentFolderName	Returns a string containing the name of the parent folder of the last component in a path.
GetSpecialFolder	Returns the special folder requested.
GetTempName	Returns a randomly generated temporary file or folder name.
MoveFile	Moves one or more files from one location to another.
MoveFolder	Moves one or more folders from one location to another.
OpenTextFile	Opens a file and returns a **TextStream** object

FileSystemObject Properties

Properties	Description
Drives	A **Drives** collection of all **Drive** objects available on the local machine.

Collections returned by **FileSystemObject** method calls reflect the state of the file system when the collection was created. Changes to the file system after creation are not reflected in the collection. If the file system might be changed during the lifetime of the collection object, the method returning the collection should be called again to ensure that the contents are current.

```
Set fs = CreateObject("Scripting.FileSystemObject")
Set a = fs.CreateTextFile("c:\testfile.txt", True)
a.WriteLine("This is a test.")
a.Close
```

In the code shown above, the CreateObject function returns the FileSystemObject (fs). The CreateTextFile method then creates the file as a TextStream object (a) and the VBScript TextStream Object WriteLine Method writes a line of text to the created text file. The VBScript TextStream Object Close Method flushes the buffer and closes the file.

Drive Object

The **Drive** object provides access to the properties of a particular disk drive or network shared drive.

Drive Object Properties

Properties	Description
AvailableSpace	The amount of space available to a user on the specified drive or network share.
DriveLetter	The drive letter of a physical local drive or network share
DriveType	A value indicating the type of a drive.
FileSystem	The amount of free space available to a user on the drive or network share.
FreeSpace	The amount of free space available to a user on the drive or network share.
IsReady	True if the drive is ready, False if not.
Path	The file system path for a drive.
RootFolder	A Folder object representing the root folder of a drive.
SerialNumber	The decimal serial number used to uniquely identify the disk volume.
ShareName	The network share name of a drive
TotalSize	The total space, in bytes, of a drive or network share
VolumeName	The volume name of a drive.

The following code illustrates the use of the Drive object to access drive properties:

```
Sub ShowFreeSpace(drvPath)
Dim fs, d, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set d = fs.GetDrive(fs.GetDriveName(drvPath))
s = "Drive " & UCase(drvPath) & " - "
s = s & d.VolumeName & vbCrLf
s = s & "Free Space: " & FormatNumber(d.FreeSpace/1024, 0)
s = s & " Kbytes"
Response.Write s
End Sub
```

File Object

The **File** object provides access to all the properties of a file.

File Object Methods

Properties	Description
Copy	Copies a file from one location to another.
Delete	Deletes a file.
Move	Moves a file from one location to another.
OpenAsTextStream	Opens a file and returns a **TextStream** object.

File Object Properties

Properties	Description
Attributes	The attributes of a file.
DateCreated	The date and time that the file was created.
DateLastAccessed	The date and time that the file was last accessed.
DateLastModified	The date and time that the file was last modified.
Drive	The drive letter of the drive on which the file resides.
Name	The name of the file.
ParentFolder	The Folder object for the parent of the file.
Path	The file system path to the file.
ShortName	The short name used by programs that require 8.3 names.
ShortPath	The short path use by programs that require 8.3 names.
Size	The size, in bytes, of a file.
Type	Information about the type of a file.

The following code illustrates how to obtain a File object and how to view one of its properties.

```
Sub ShowFileInfo(filespec)
Dim fs, f, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFile(filespec)
s = f.DateCreated
Response.Write s
End Sub
```

Folder Object

The VBScript **Folder** object provides access to all the properties of a folder.

Folder Object Methods

Properties	Description
Copy	Copies a folder from one location to another.
Delete	Deletes a folder.
Move	Moves a folder from one location to another.
CreatTextFile	Creates a file and returns a **TextStream** object.

Folder Object Properties

Properties	Description
Attributes	The attributes of a folder.
DateCreated	The date and time a folder was created.
DateLastAccessed	The date and time that the folder was last accessed.
DateLastModified	The date and time that the folder was last modified.
Drive	The drive letter of the drive on which the folder resides.
Files	A **Files** collection of all **File** objects in the folder.
IsRootFolder	**True** if this is the root folder of a drive.
Name	The name of the folder.
ParentFolder	The **Folder** object for the parent of the folder.
Path	The file system path to the folder.
ShortName	The short name used by programs that require 8.3 names.
ShortPath	The short path used by programs that require 8.3 names.
Size	The size, in bytes, of all files and subfolders contained in a folder
SubFolders	A **Folders** collection containing all the folders in a **Folder** object

The following code illustrates how to obtain a **Folder** object and how to return one of its properties:

```
Sub ShowFolderInfo(folderspec)
Dim fs, f, s,
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFolder(folderspec)
s = f.DateCreated
Response.Write s
End Sub
```

Examples

```
Const OverWrite = TRUE
Const DeleteRdOnly = True
SourceFile = “C:\data\MyData.MDB”
SourceFiles = “C:\data\*.MDB”
DestPath = “C:\Backup”
DeleteFile = “C:\backup\Mydata.MDB”
DeleteFiles = “C:\backup\*.MDB)
Set objFS = CreateObject(“Scripting.FileSystemObject”)

‘  Copy a single file to a new folder, overwrite any existing file in destination folder
objFS.CopyFile (SourceFile, DestPath, OverWrite)

‘ Copy a set of files to a new folder, overwrite any existing files in destination folder
objFS.CopyFile (SourceFiles, DestPath. OverWrite)
```

```
' Delete a file
objFS.DeleteFile(DeleteFile)

' Delete a set of files in a folder
objFS.DeleteFile(DeleteFiles, DeleteRdOnly)

' Move a file to a new folder
objFS.MoveFile(SourceFile, DestPath)

' Move a set of files to a new folder
objFS.MoveFile(SourceFiles, DestPath)

' Rename a file
objFS.MoveFile(SourceFile, "C:\data\MyData041406.MDB")

' Verify if a file exists
If objFS.FileExists (SourceFile) Then
    Set objFolder =objFS.GetFile(SourceFile)
    MsgBox "File Exists " & objFolder          ' Will display "File Exists " and Path + File
Else
    MsgBox "File does not exist"
End If
```

VBScript Drives Collection

Read-only collection of all available drives. Removable-media drives need not have media inserted for them to appear in the Drives collection.

Drives Collection Object Properties

Properties	Description
Count	Returns the number of items in a collection. Read-only
Item	Returns an item on the specified key. Read/Write

The following code illustrates how to get the Drives collection and iterate the collection using the For Each...Next statement:

```
Sub ShowDriveList
Dim fs, d, dc, s, n
Set fs = CreateObject("Scripting.FileSystemObject")
Set dc = fs.Drives
For Each d in dc
  s = s & d.DriveLetter & " - "
If d.DriveType = Remote Then
  n = d.ShareName
Else
  n = d.VolumeName
End If
s = s & n & vbCrLf
Next
Response.Write s
End Sub
```

VBScript Files Collection

Collection of all **File** objects within a folder.

Files Collection Object Properties

Properties	Description
Count	Returns the number of items in a collection. Read-only
Item	Returns an item on the specified key. Read/Write

The following code illustrates how to get a Files collection and iterate the collection using the For Each...Next statement:

```
Sub ShowFolderList(folderspec)
Dim fs, f, f1, fc, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFolder(folderspec)
Set fc = f.Files
For Each f1 in fc
s = s & f1.name
s = s & vbCrLf
Next
Response.Write s
End Sub
```

VBScript Folders Collection

Collection of all Folder objects contained within a Folder object.

Folders Collection Methods

Properties	Description
Add	Adds a new **Folder** to a **Folders** collection

Folders Collection Properties

Properties	Description
Count	Returns the number of items in a collection. Read-only
Item	Returns an item on the specified key. Read/Write

The following code illustrates how to get a Folders collection and how to iterate the collection using the For Each...Next statement:

```
Sub ShowFolderList(folderspec)
Dim fs, f, f1, fc, s
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFolder(folderspec)
Set fc = f.SubFolders
For Each f1 in fc
s = s & f1.name
s = s & vbCrLf
Next
Response.Write s
End Sub
```

TextStream Object

The VBScript **TextStream** object facilitates sequential access to a file

TextStream Object Methods

Properties	Description
Close	Closes an open stream.
Read	Reads a specified number of characters from a stream.
ReadAll	Reads an entire stream.
ReadLine	Reads an entire line from a stream.
Skip	Skips a specified number of characters when reading a stream.
SkipLine	Skips the next line when reading a stream.
Write	Writes a specified string to a stream.
WriteBlankLines	Writes a specified number of newline characters to a stream.
WriteLine	Writes a specified string and newline character to a stream.

TextStream Object Properties

Properties	Description
AtEndOfLine	**True** if the file pointer is before the end-of-line marker.
AtEndOfStream	**True** if the file pointer is at the end of the stream
Column	The column number of the current character in the stream.
Line	The current line number of the stream.

VBScript TextStream Object

Description: The VBScript TextStream object
Usage: oTextStream.{property | method}
Return: Depends on Property or Method used
Remarks
Example: In the following code, a is the TextStream object returned by the CreateTextFile method on the FileSystemObject:

```
Set fs = CreateObject("Scripting.FileSystemObject")
Set a = fs.CreateTextFile("c:\testfile.txt", True)
a.WriteLine("This is a test.")
a.close
```

COM Objects and Collections

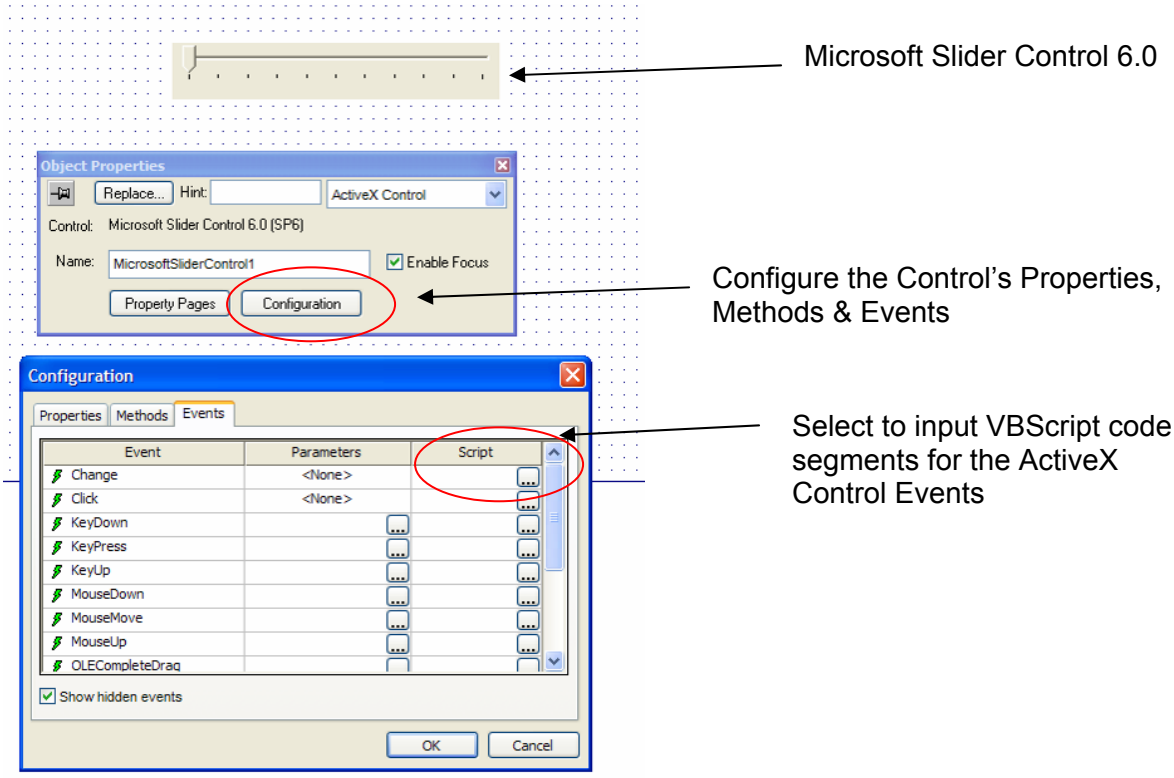
In addition to user-defined Class Objects and VBScript Objects and Collections, there are many different COM Objects (and Object Collections) and other system objects based on COM technology that are accessible from VBScript. These Objects include:

- ActiveX Controls inserted on an IWS Screen (via Insert OCX tool)
- ActiveX Controls instantiated via VBScript
- ADODB and ADOX Objects and Collections
- Microsoft Office OLE Automation (Word, Excel, Access, Outlook & Components)
- WMI
- WSH
- WSDL
- XMLDOM

ActiveX Controls Inserted On An IWS Screen

InduSoft Web Studio (IWS) serves as an ActiveX control container, which is a parent program that supplies the environment for an ActiveX control to run. Through the IWS development interface (insert OCX tool), one or more ActiveX controls can be added to a screen. The OCX (ActiveX Control) must first be registered, if it was not already done so as part of the installation of the ActiveX control. IWS provides a Register Controls tool (under Tools on the toolbar) to allow registration of ActiveX controls, and to verify if a control has already been registered.

After the OCX is inserted on the screen, IWS will assign the control a name. This name can be changed in the Object Properties dialog box, accessed by double clicking on the control in the IWS development environment, but the name of the control must be unique from any other control used by the current IWS application. In the Object Properties dialog box, the Configuration button will provide access to the Properties, Methods and Events accessible for this ActiveX control. In the Configuration dialog box, there is a tab for Events, which allow for the execution of a VBScript code segment if an Event is triggered for the ActiveX control. In the Properties and Methods tabs, parameters, triggers, IWS tags, etc. can be tied to the various Properties and Methods.



Interaction with the ActiveX control from VBScript is accomplished through VBScript code placed in a Screen Script that is associated with the screen where the ActiveX control is placed. By entering a right mouse click on a blank portion of the screen, and selecting Screen Script, the Screen Script is accessed. For ActiveX Objects placed on the screen, you do not need to instantiate the Object in VBScript, IWS has already taken care of this. You simply need to reference the ActiveX control by its name, found in the Object Properties dialog box. **Note: when referring to the name from VBScript, with IWS 6.1 SP1, the ActiveX control name was case sensitive**. **With SP2, it is no longer case sensitive.** From the VBScript screen interface, you can access the ActiveX control's Properties and Methods. Events are not accessible from the VBScript Screen Script interface.

Notes:

- You must use the VBScript Screen Script interface for the screen which contains the ActiveX control in order to access the ActiveX control's Properties and Methods. You cannot access the ActiveX control's Properties and Methods from another Screen Script, or from any other VBScript interface in IWS.
- From VBScript, you can only access the ActiveX control's Properties and Methods. VBScript code segments for Events that are triggered by the ActiveX control can be entered, but these VBScript code segments must be entered from the Configuration dialog box (i.e. Object Properties → Configuration → Events).
- When the ActiveX control is referenced from the VBScript Screen Script interface, the ActiveX control's name is case-sensitive with SP1. This was corrected in SP2.
- You do not need to instantiate the ActiveX control. IWS has already taken care of this. Simply refer to the ActiveX control name followed by a "." and then the Property or Method.
- In the VBScript Screen Script interface, place the cursor in a code segment area (Subroutine) and press Ctrl –Space to invoke IntelliSense to see the VBScript statements and functions, as well as the ActiveX controls available for this Script Interface.
- Once you enter the ActiveX control object name, when you type a period ("."), Intellisense will display a list of available Properties and Methods for the ActiveX control referenced.

ActiveX Controls Instantiated from VBScript

ActiveX controls can be instantiated from VBScript by using the CreateObject and referencing the Program ID (ProgID) of the ActiveX object, although the ActiveX object will not show up on the IWS screen if the script segment is associated with a Screen.

ADODB and ADOX Objects and Collections

ADODB is the database class for ADO.NET, or ActiveX Data Objects for Microsoft's .NET Framework. ADO.NET is Microsoft's database interface technology that provides an API to database client applications (i.e. IWS and VBScript), supporting a common interface to access and manipulate data contained in a wide variety of database servers from different vendors. From the database client side, there is a level of abstraction provided by the API that enables interaction (e.g. database access and manipulation) to various vendor's databases with virtually no code changes, except for the connection string to the database Provider (an object that interacts with the physical database). There are various ADODB Objects and Collections available to the developer.

ADOX are Microsoft's ActiveX Data Object Extensions for Data Definition Language (database schema creation, modification and deletion) and Security. It is a companion set of Objects to the core ADO.NET objects.

VBScript Configuration and Operation in IWS

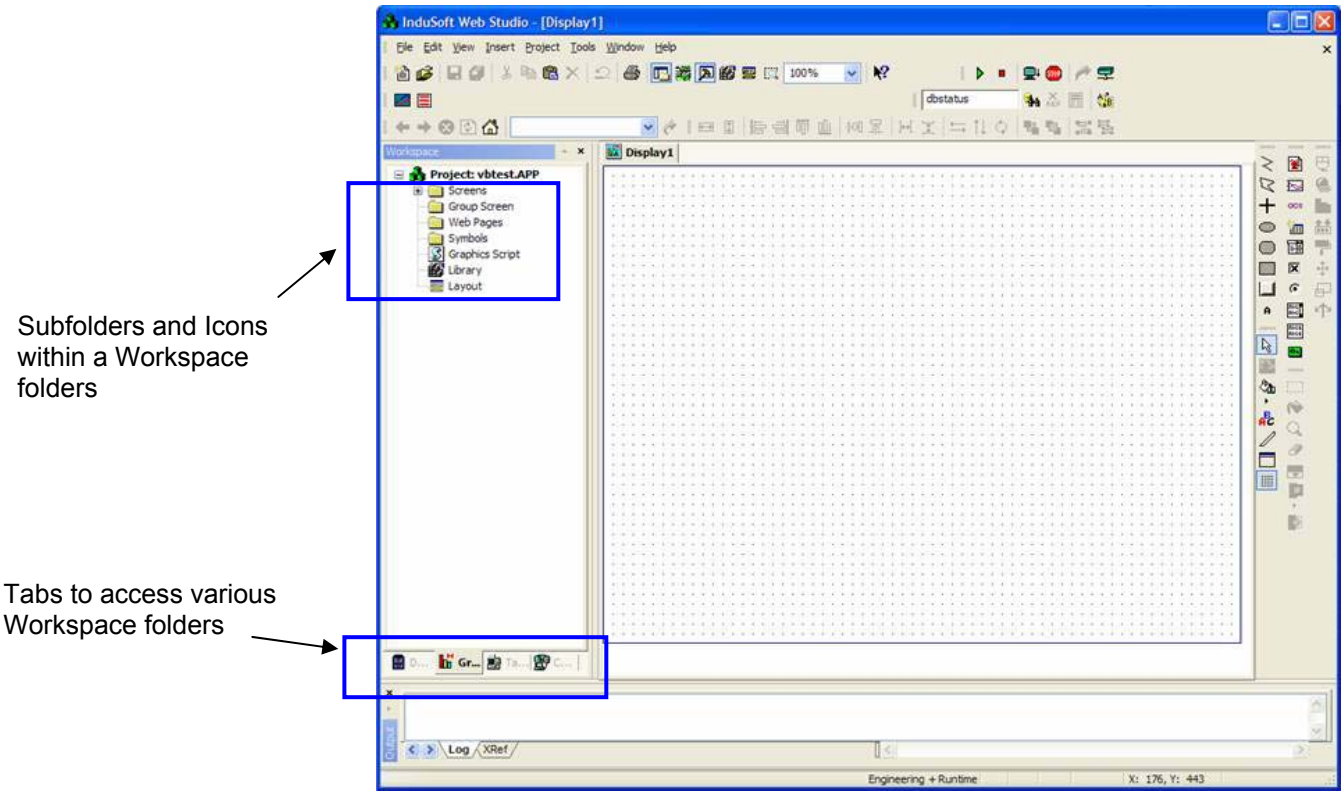
IWS acts as the host application for the Microsoft VBScript Engine. This means that to write VBScript, you need to be in the IWS development (engineering) environment. It is important to note that there is no one central location where a VBScript interface is located inside an IWS application. The location of the VBScript interface (where the VBScript code gets placed) depends on the function the VBScript code is to perform and the scope of access to its Procedures and Variables. InduSoft has implement VBScript in this manner to simplify its use, and to be consistent with the IWS architecture as well as current licensing methods.

VBScript is interpreted code. While it executes fairly efficiently, it is nevertheless interpreted and will never execute as efficiently as compiled code. This should not present any concern for HMI/SCADA applications since IWS is performing the real-time management of the tag database and key functions such as alarming, logging, etc. The interpreted nature of VBScript allows changes to be made quickly to an application. IWS supports dynamic, on-line configuration and this capability is maintained with the addition of VBScript support

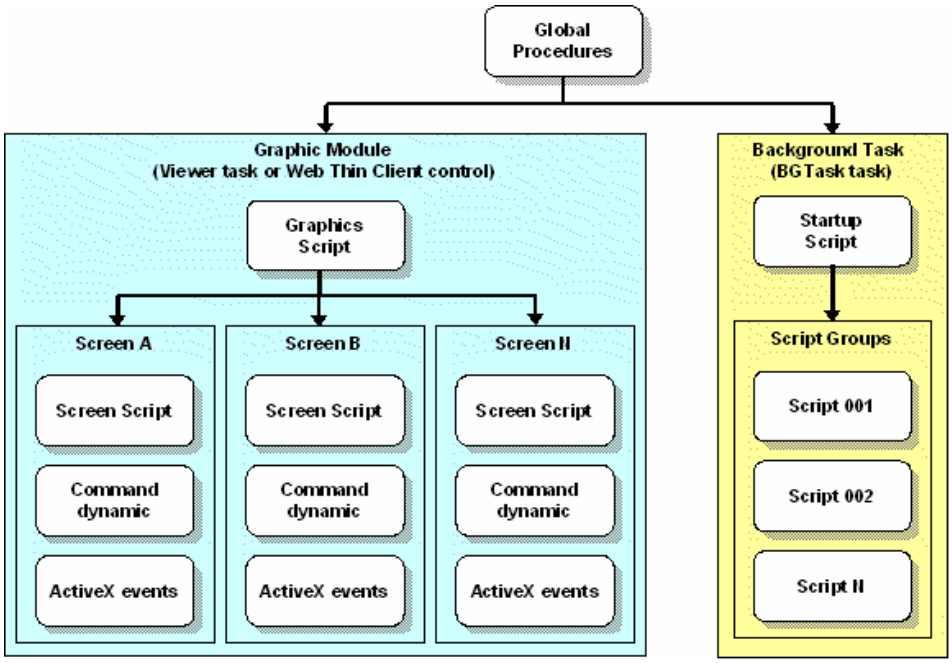
Developers familiar with IWS know that in the bottom left corner of the development window are tabs that provide access to the Database, Graphics, Tasks, and Communications Workspace folders containing the different application components. The developer will need to navigate among these different folders and application components when using VBScript.

VBScript interfaces can be found in 6 different areas:

- Database Workspace folder – Global Procedures
- Graphics Workspace folder - Graphics Script
- Graphic Screens – Screen Scripts
- IWS Objects on a Screen – Command Dynamic
- ActiveX Objects on a screen – ActiveX Events
- Tasks Workspace folder – Background Startup Script and Background Script Groups

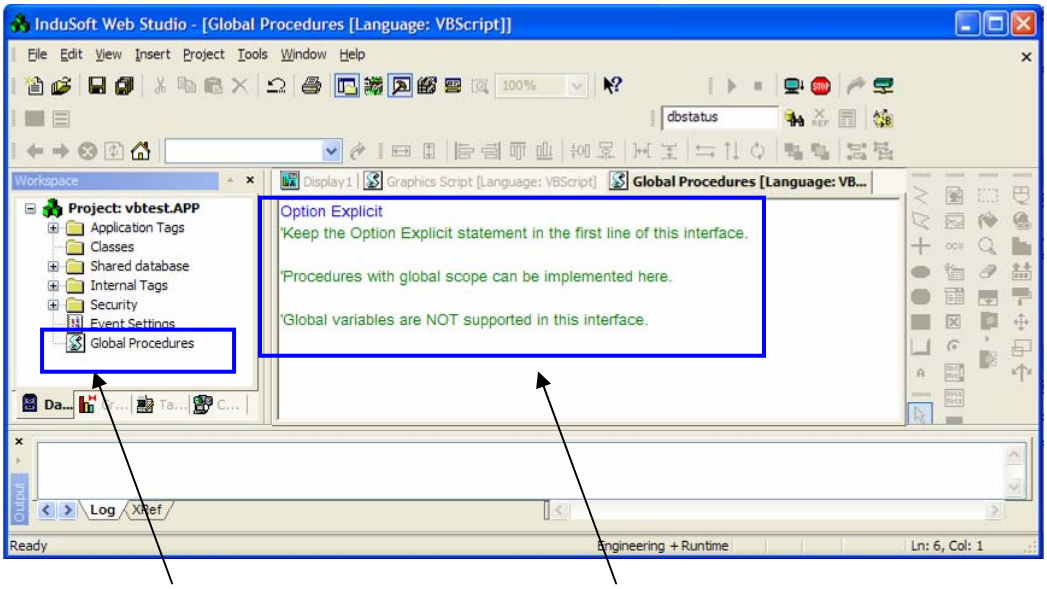


The figure below shows the structure of the VBScript interfaces within a typical IWS project (application). Note that there are certain types of VBScript interfaces that have one instance (e.g. Global Procedures, Background Startup Tasks and Graphic Script) while others can have multiple instances (e.g. Background Script Groups, Screen Scripts, Command Dynamic and ActiveX Events).



Global Procedures

Global Procedures are located in the Database Worksheet folder. Global Procedures are shared by both the Graphics Module Scripts (Graphics Script and Screen Scripts) and the Background Task Scripts (Background Startup Script and Background Script Groups). Note that it this is only the Procedures that are shared, not the Variables. Other VBScript interfaces within the Graphic Module or Background Task do not share variables or procedures between them; they are independent of each other.



Global Procedures Subfolder in Database Worksheet folder

VBScript Interface

Notes:

- Before executing the application, be sure to save (or close) the Screen after any VBScript is entered. Otherwise the changes might not be updated. This is true for all VBScript interfaces.

Graphics Script

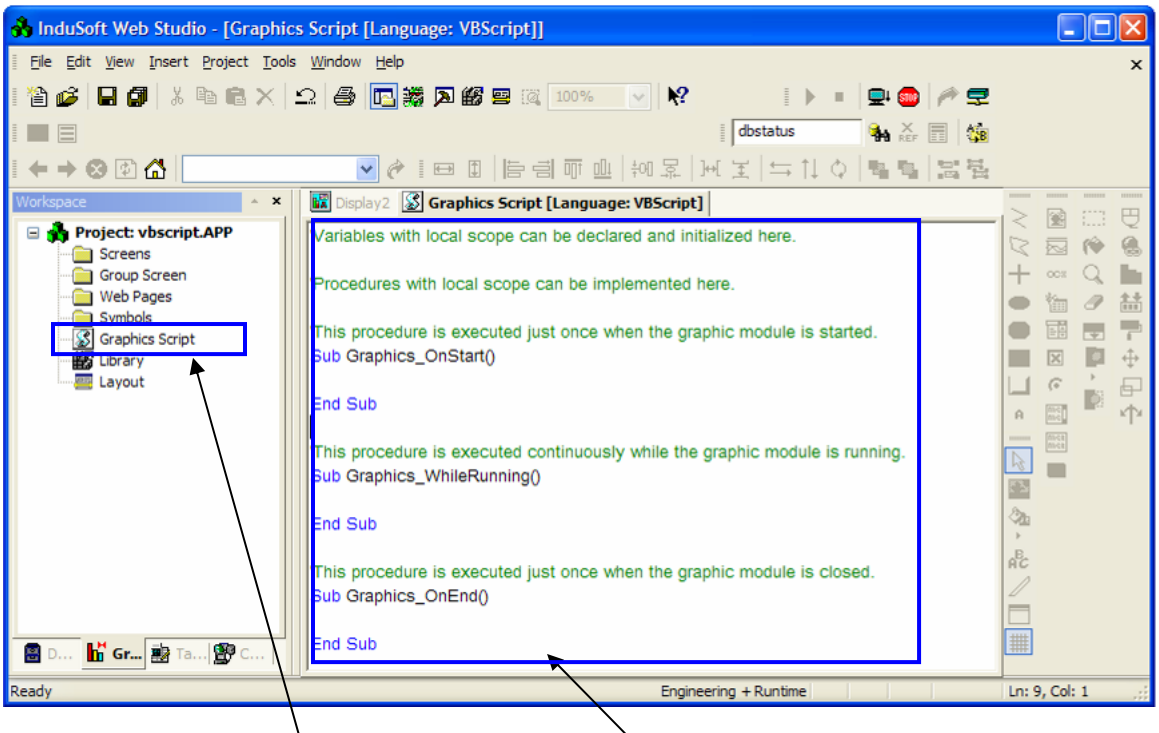
The Graphics Script is located in the Graphics Worksheet folder. Procedures and Variables declared in the Graphics Script interface are available locally but are not accessible by any Screen Script interface, or from any other VBScript interface within IWS. Procedures and Variables declared in a Screen Script interface are not accessible by the Graphics Script. If common Procedure(s) are required, they should be put into the Global Procedures interface. Note that the Graphics Script is scanned (processed) by IWS before the Screen Scripts.

The Graphics Script has three different pre-configured subroutines to execute VBScript code. These subroutines execute the VBScript contained in them based on the event state of the Graphics Module. These are:

Graphics_OnStart
Code contained within this subroutine is automatically executed just once when the Graphics Module is started. This is a good area to initialize variables or execute start-up code.

Graphics_WhileRunning
Code contained within this subroutine is automatically executed continuously while the Graphics Module is running. The rate at which this subroutine is called depends on the performance of the hardware platform and other tasks running at the time.

Graphics_OnEnd
Code contained within this subroutine is automatically executed just once when the Graphics Module is closed.



Graphic Script Icon in Graphics Worksheet folder

VBScript Interface

Note:
- Do not change the name of the pre-configured subroutines in the VBScript interface. Otherwise they many not properly execute.

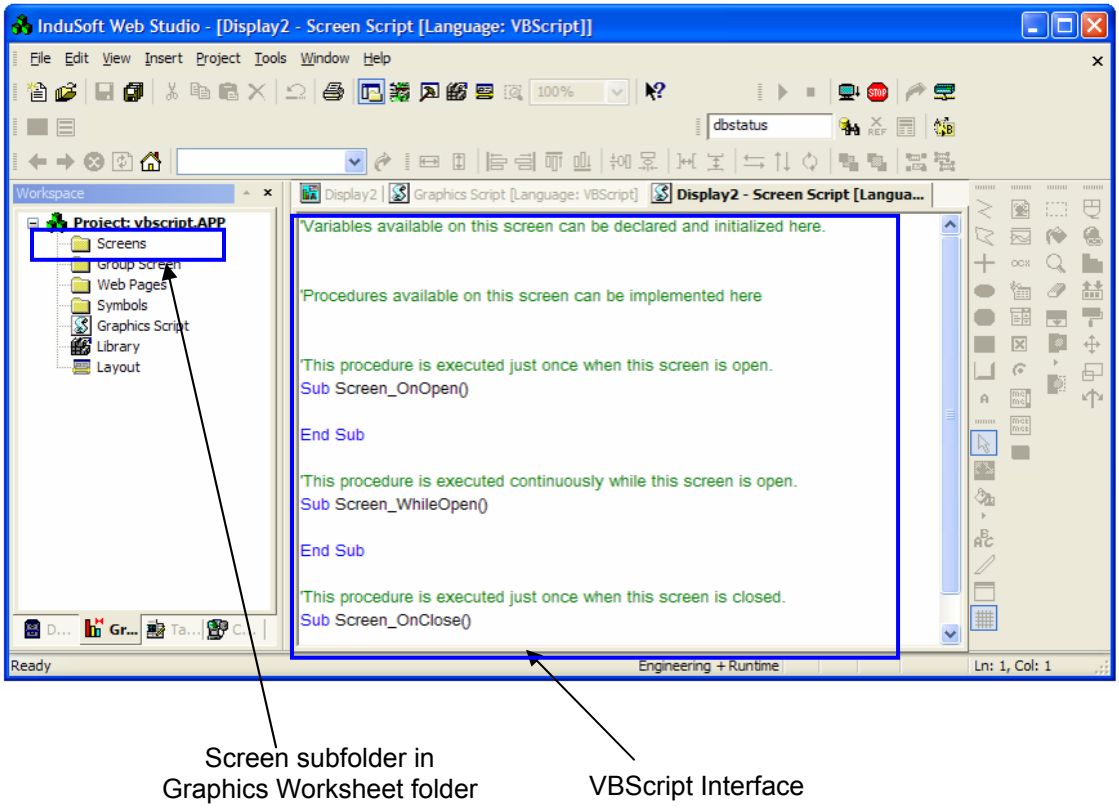
The Graphics Script operates for both the Server (the host processor where the IWS application is running) and Web Thin Clients (web browser interface using Microsoft Internet Explorer). For the Server, the Graphics module is the Viewer task (the display on the host processor), while the ISSymbol control is the Graphics module for Web Thin Clients.

The operation of the Graphics Script on the Server is described above, and starts when the application is started on the Server, assuming there are one or more screens. But since Web Thin Clients can log on at any time after the Server is started, the functioning of the Graphics Script is different for Web Thin Clients and is independent of the operation of the Graphics Script on the host Server. Web Thin Client operation is as follows:

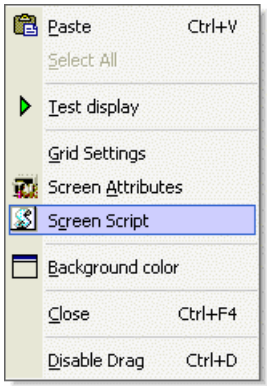
- When a Web Thin Client logs on to the Server, following completion of the log on process, the Graphics_OnStart subroutine will be executed for the Web Thin Client. This will occur each time any new Web Thin Client logs on to the Server.
- Following completion of the execution of the Graphics_OnStart subroutine, the Graphics_WhileRunning subroutine will be executed for as long as the Web Thin Client (browser) hosts the ISSymbol control (i.e. while an active network link exists and the ISSymbol is active in the browser).
- When the Web Thin Client is shut down or when the ISSymbol control is no longer hosted by the browser, the Graphics_OnEnd subroutine is executed

Screen Scripts

Screen Scripts are associated with individual graphical screens. These screens can be for display on the host Server (where the IWS application is running), for a Web Thin Client, or both. Procedures and Variables declared in a Screen Script VBScript interface are not accessible by any other VBScript interface within IWS. However, the Screen Script interface can access procedures declared in the Global Procedures script interface.



There are two methods to access a Screen Script. The first is to select the desired Screen and have it displayed on the active IWS workspace. Then, perform a right mouse click while the cursor is located on the display screen. A pop-up menu will let you select the **Screen Script** (as shown at the right). When the **Screen Script** option is selected, the IWS workspace will display the Screen Script VBScript interface.

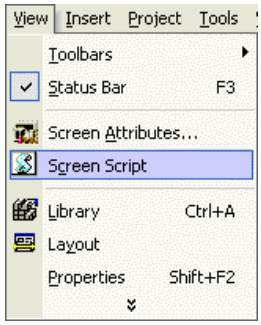


Pop-Up Menu

Notice the Screen Script VBScript interface looks very similar to the Graphics Script interface. The differences between the Screen Script and the Graphics Script are:

- There is only one Graphics Script. The Graphics Script is activated when the Graphics Module starts
- You can have multiple Screen Scripts. There is one Screen Script available per Screen, but you can have multiple screens.

The second method to access a Screen Script is to select the desired Screen and have it displayed on the active IWS workspace. Then from the top toolbar, select **View**. A pull-down menu (as shown at the right) will have the **Screen Script** option available. By selecting this option, you will activate the Screen Script VBScripting interface.



Pull-down Menu

The Screen Script interface has three predefined subroutines. These are:

Screen_OnOpen
Code contained within this subroutine is automatically executed just once when the Screen is opened.

Screen_WhileOpen
Code contained within this subroutine is automatically executed continuously while the Screen is open. The rate at which this subroutine is called depends on the performance of the hardware platform and other tasks running at the time.

Screen_OnClose
Code contained within this subroutine is automatically executed just once when the Screen is closed.

The execution of the Screen Script subroutines on the Server executes independently from the execution on Web Thin Clients.

Notes:

- Before executing the application, be sure to save (or close) the Screen after any VBScript is entered. Otherwise it might not be updated. This is true for all VBScript interfaces.
- The Graphic Script is scanned (processed) by IWS before the Screen Scripts are processed.

Command Dynamic

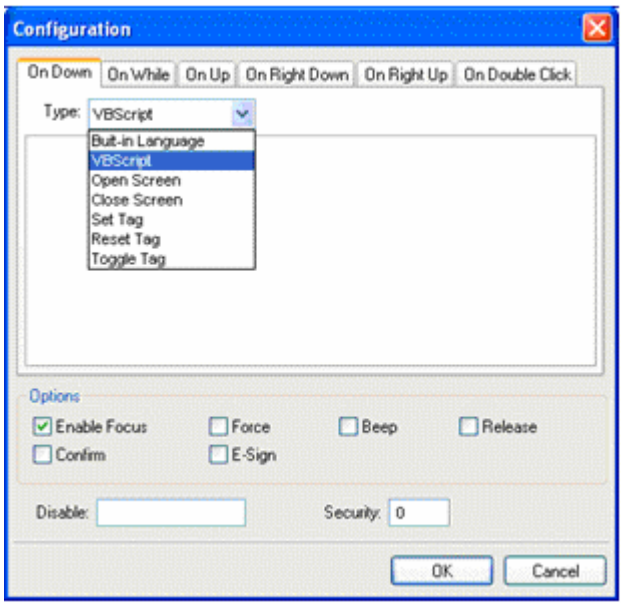
A Command Dynamic is associated with a specific object on a Screen, and allows one or more actions to take place when an event occurs with the specific object. A typical use is a button (perhaps a rectangle) that is placed on the screen. When an operator selects on the button (via mouse click or pressing a touchscreen over the object), this action is expected to initiate some action. That action may be to set/reset a PLC bit, jump to a different screen, whatever. The Command Dynamic allows the developer to chose what action to take.

With Version 6.1, IWS adds new capability to the Command Dynamic interface. In addition to the IWS built-in language command, the Command Dynamic can now execute VBScript code. The steps to access the VBScript interface within a Command Dynamic are:

1. Select the object on the Screen currently opened in the IWS workspace. If the object has a Command Dynamic associated with it, then right click on the object. Otherwise, click on the Command Dynamic icon (right) from the Mode toolbar and then right click on the object.

2. Now, the Object Properties dialog box for the Command Dynamic will open. Click on the Config… button in the lower right corner of the dialog box.

3. Select the event condition (e.g. On Down) where your want code to be execute and then select VBScript as the Type.



4. Enter your VBScript code (variable declarations and executable statements).

Within the Command Dynamic, you enter VBScript variables and executable statements subject to the following conditions:

- Any variable declared in this interface will only have a local scope.
- You cannot implement procedures (i.e. Subroutines or Functions) within this interface.

Notwithstanding these restrictions, VBScript code within a Command Dynamic still has access to all Global Procedures.

VBScript code within the Command Dynamic interface is executed whenever one or more of the selected event conditions (listed in the Command Dynamic configuration screen) occur for the selected object. The execution of the Command Object script on the Server executes independently from the execution on Web Thin Clients.

Notes:

- Before executing the application, be sure to save (or close) the Screen after any VBScript is entered. Otherwise it might not be updated. This is true for all VBScript interfaces.

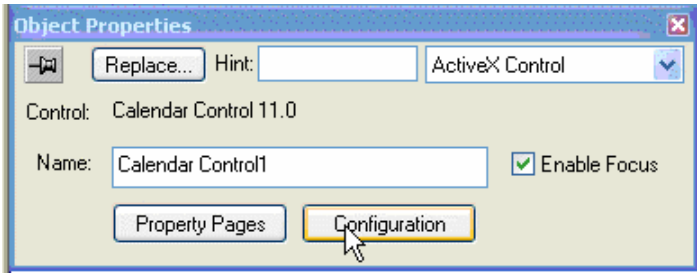
ActiveX Events

IWS is an ActiveX container, supporting ActiveX controls, generally inserted on a given graphical screen. With IWS Version 6.1, there is a VBScript interface to ActiveX Events so that an ActiveX object event can trigger a VBScript code segment.

The steps to accessing the VBScript ActiveX Event interface are as follows:

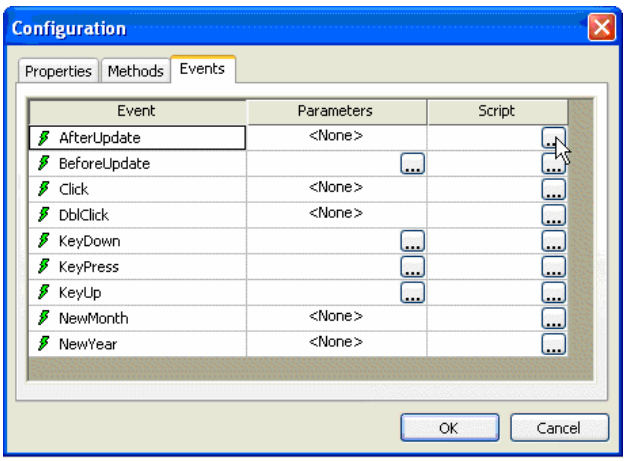
1. Select the ActiveX object on the Screen currently opened in the IWS workspace. Right click on the object to open its Object Properties dialog box. If you need to insert an ActiveX object, select the ActiveX Control icon from the Mode toolbar and then right click on the object

   In the lower right corner of the ActiveX Object Properties dialog box will be a Configuration button. Click this to open up the Configuration options dialog box.

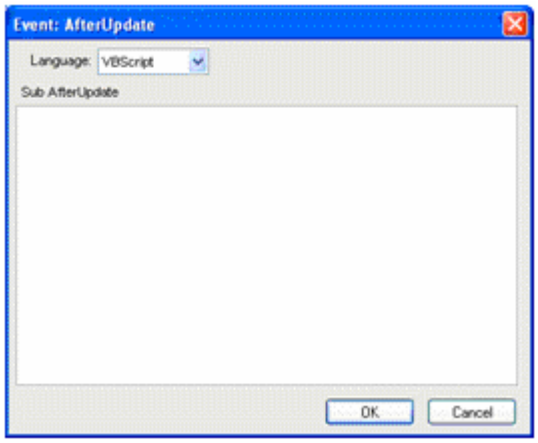


2. Click on the Events tab (as shown at the right).

3. Click on the … button in the Script Column for the event you want to write VBScript for.



This is the scripting interface for ActiveX Events. Be sure VBScript language is selected. You can now insert code that will execute when the selected ActiveX Event is triggered.



Within the ActiveX Event interface, you enter VBScript variables and executable statements subject to the following conditions:

- Any variable declared in this interface will only have a local scope.
- You cannot implement procedures (i.e. Subroutines or Functions) within this interface.

Notwithstanding these restrictions, VBScript code within the ActiveX Event interface still has access to all Global Procedures, as well as any procedures in the Screen Script for the same Screen where the ActiveX object is configured.

VBScript code within the ActiveX Event interface is executed whenever one or more of the selected Event conditions (listed in the Configuration dialog box) occur for the selected ActiveX object. The execution of the script on the Server executes independently from the execution on Web Thin Clients.

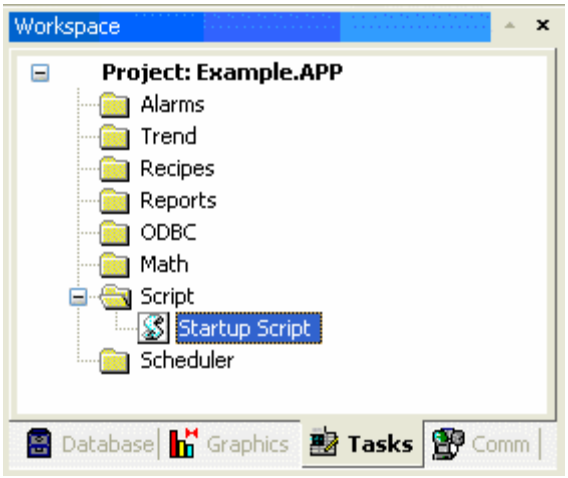
Notes:

- Before executing the application, be sure to save (or close) the Screen after any VBScript is entered. Otherwise it might not be updated. This is true for all VBScript interfaces.

Background Task Startup Script

In the Tasks Worksheet folder is the Script subfolder which will contain a default Startup Script icon and any Background Task Script Groups declared. To edit the Background Task Startup Script:

1. Click on the Tasks Worksheet folder
2. Click on the Script subfolder.



Any VBScript code placed in this interface will execute when the Background Task module is started, which occurs when the IWS application is started. This code will only execute once, and is meant for initialization purposes.

Variables and Procedures declared in the Background Task Startup Script are available to the Background Task Script Group, but are not available to any VBScript interfaces in the Graphic Module. Remember that the Background Task Group Startup Script can access the procedures declared in Global Procedures.

Since the Background Task Startup Script has no interaction with a Graphics script, the only Server display I/O functions that can be implemented are **MsgBox** and **InputBox** functions.

Since the Background Task Startup Script runs on the IWS Server, there is no effect with Web Thin Clients.

Background Task Script Groups

The Background Task Script Groups consist of one or more VBScript interface groups that run in the Background Task. By default, there are no Background Task Script Groups unless added by the developer. These Script Groups will execute in a background as long as their Execution Field is in a **TRUE** state.

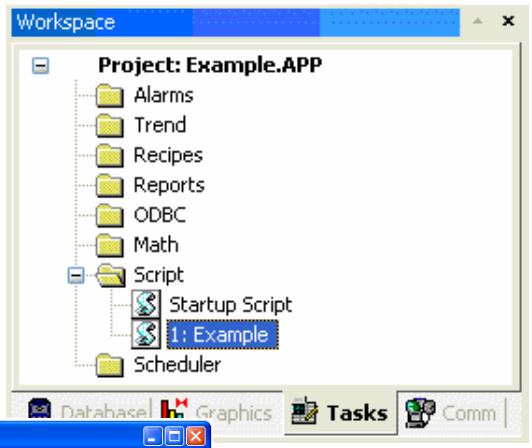
Background Task Script Groups have the following limitations:

- Variables declared in a Background Task Script Group have a local scope for it's specific Script Group only. Variables cannot be shared with other Script Groups, nor any other VBScript Interface.
- Background Task Script Groups cannot declare their own Procedures (Subroutines and Functions).
- The Execution Field of the Script Group will only support IWS tags or built-in functions. No support for VBScript variables or Procedures is provided in the Execution Field.

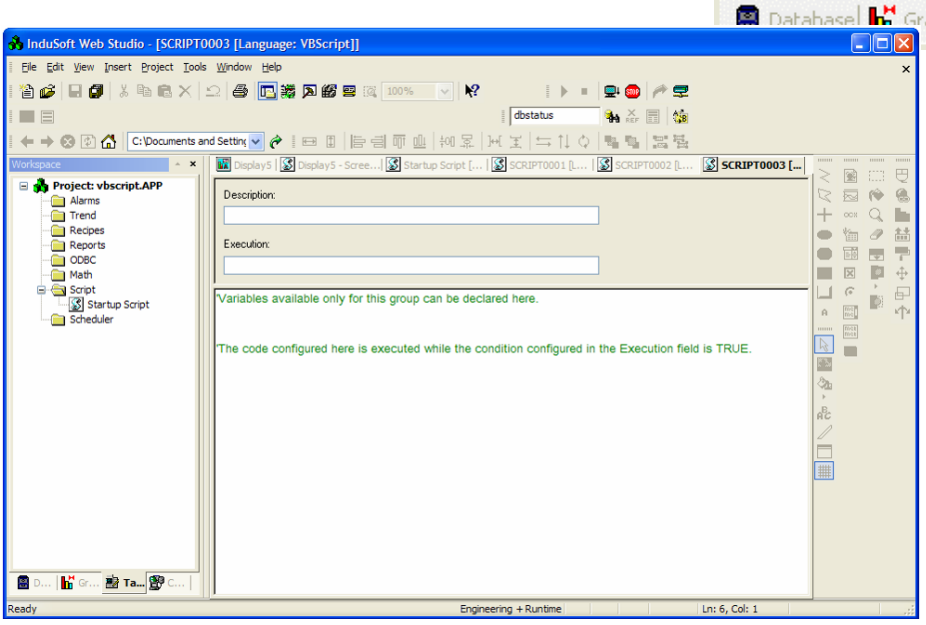
However, the Background Task Script Groups can do the following:

- Access Procedures and Variables within the Background Task Startup Script.
- Access Procedures declared in Global Procedures.

To create a new Script Group, right-click on the Script subfolder in the Tasks tab of the Workspace. Select the Insert option from the pop-up menu. Note that the Startup Script is already defined. To open (edit) an existing Script Group, simply click its icon in the Script subfolder of the Tasks workspace tab.



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 set to or is evaluated to be **TRUE** (a value different from 0).



When any Script Group is saved during runtime (e.g. from an on-line configuration download), the Startup Script interface will be executed again, and the current value of the local variables contained in any Script Group will be reset, if any exist.

Since the Background Task Script Groups run on the IWS Server, there is no effect with Web Thin Clients.

Notes:

- The Execution Field of the Script Group only supports syntax as specified by the IWS built-in language.
- Before executing the application, be sure to save (or close) the Screen after any VBScript is entered. Otherwise it might not be updated. This is true for all VBScript interfaces.
- If any Script Group is saved during runtime (i.e. on-line configuration), the Startup Script Group will be executed again and the current value of local variables will be reset

Scope of VBScript Procedures and Variables

The following table summarizes the relationship between the IWS VBScript interface location and its Scope of Variables and Procedures. The table also defines where the Scripts are located

Procedures (local)

Item	Scope of Procedures and Variables	Execution	Functionality	Location/Access
Global Procedures	All Procedures are global, Variables are accessible only within Global	Procedures are accessible to any Script on the host Server	Declaration of Procedures (Functions and Subroutines) that are available globally	Database Workspace Folder
Graphics Scripts	Procedures and Variables accessible within Graphics Script interface only. Can Call Global Procedures.	Executes on host Server and/or Web Thin Client where any screen is displayed.	Condition-based execution - Graphics Start - Graphics Open - Graphics Close Graphics Scripts execute before Screen Scripts	Graphics Workspace Folder
Screen Scripts	Procedures and Variables accessible within Screen where the Script is written. Screen Script procedures accessible to ActiveX Events for ActiveX objects contained in the Screen. Can Call Global Procedures.	Executes on host Server and/or Web Thin Client where the specific screen is displayed	Condition-based execution - Screen Start - Screen Open - Screen Close	Within the Screen.
Command Dynamic	Variables and Script accessible only in Object where the Script is configured. Can Call Global Procedures.	Executes on host Server and/or Web Thin Client where the screen with the specific Object is displayed	Execution of Script when Object condition is met	Within Object (Command) Properties. The Screen that uses the Object must be open.
ActiveX Events	Variables accessible only in Object where the Script is configured. Screen Script Procedures are accessible. Can Call Global Procedures.	Executes on host Server and/or Web Thin Client where the screen with the specific Object is displayed	Execution of Script when selected ActiveX Event occurs	Within the ActiveX object. The Screen that uses the Object must be open.

Item	Scope of Procedures and Variables	Execution	Functionality	Location/Access
Background Startup Script	Procedures and Variables accessible within the Script Group. Can Call Global Procedures.	Executes on Server as a background task	Declaration of Procedures and Variables that are available for Background Scripts	Tasks Workspace Folder
Background Script Groups	Accessible within Script Group only. Can Call Global Procedures.	Executes on Server as a background task	Condition-based execution in background mode. Can have multiple Script pages.	Tasks Workspace Folder

Accessing IWS Tags and IWS Built-in functions

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:

```
CurTime = $Time                 ' Returns the value of the tag Time from the tags database
a = $MyTag                      ' Sets a to the value of the IWS tag MyTag
$Open("main.scr")               ' Executes the Open() function to open the "main" screen
```

IWS tags and built-in functions are accessible from any VBScript code segment, regardless where located. If the IWS function returns a value (e.g. error or status information), this can be assigned to a VBScript variable. IWS tags can be used as arguments in VBScript statements and functions.

If an undefined name follows the “$”, when the programmer does a **Check Script** function or attempts to **Save** the script, IWS will ask the programmer if they want to define the IWS tag, and if so, prompt for the tag type.

IWS supports the following application tag types:

- **Boolean** (a Boolean (True/False) or digital value (0 or 1))
- **Integer** (a 32-bit long-word signed integer type)
- **Real** (a real number stored as a double precision word)
- **String** (a string of characters of up to 255 characters that holds letters, numbers, or special characters)
- **Class** (a user-defined, compound tag)
- **Array** (an array of values from 0 to 16,384)

Passing variables between VBScript and IWS is straightforward but there are some conversion considerations that should be noted:

IWS Boolean

With VBScript, variable can be of the data subtype Boolean. VBScript defines keywords **True** and **False** for logical states True and False, respectively. In VBScript, False has a numeric value of 0, while True has a numeric value of -1. This is because Booleans are not actually stored as bits, but as 32-bit signed integers. If all bits are zero, then it is a 0 or logical False. If all bits are set to 1, then it is a signed value of -1 or a logical True.

IWS objects that display IWS-defined boolean tags (e.g. Text I/O) will have the boolean values displayed as 0 or 1 (0=False, 1=True), not as False or True. Consider the following VBScript code segment:

```
$MyBool = True                  ' Will be displayed as a “1” in an IWS object (*see below)
$MyBool = False                 ' Will be displayed as a “0” in an IWS object
```

The value for True assumed by Boolean IWS tags depends on the value of the parameter **BooleanTrueAboveZero** that is located in the **[Options]** section of the <Application>.APP file. To access this parameter, you need to open the <Application>.APP file with a simple text editor such as Microsoft Notepad. For example:

```
[Options]
BooleanTrueAboveZero = 0        IWS Boolean tag set to value 1 (True) when value <> 0

[Options]
BooleanTrueAboveZero = 1        IWS Boolean tag set to value 1 (True) only when value > 0
```

One item to watch for is the boolean NOT operator. With an IWS tag, even though the tag is of type Boolean, it is really stored internally as a 32-bit signed variable. If you NOT a 0, the lower bit is set to one but in reality all the bits are set to 1’s, meaning that with a variable that is a signed integer, the NOT of 0 is really -1. For example,

```
a = CBool(Not(0))
$c = a                          ' $c (IWS tag c) will display as  -1
```

One programming trick that can be used when attempting to toggle IWS Boolean Tags between 0 and 1 is either:

```
$tag = Abs ($tag=0)          ‘ Either one of these statements will toggle the tag
$tag = $If ($tag=0,1,0)      ‘ between 0 and 1
```

IWS Integer

All IWS integer tags are stored as 32-bit values. VBScript has 3 different variant subtypes that are of interest. Bytes are 8-bit values that are positive whole numbers ranging from 0 to 255. Integers are 16-bit signed values that range from -32,768 to 32,767. Long Integers are 32-bit values that range from -2,147,483,648 to 2,147,483,647.

When storing to an IWS integer tag, the conversion to a 32-bit signed integer type will be automatically made. For example:

```
a = CInt (-30)               ‘ a is a 16-bit signed integer with a value of -30
$MyInt = a                   ‘ MyInt is a 32-bit signed integer with a value of -30
b = CByte (-30)              ‘ Generates an error since Bytes are 0 to 255, not negative
b = CByte (30)               ‘ b is a 8-bit unsigned integer with a value of 30
$MyInt = b                   ‘ MyInt is a 32-bt signed integer with a value of 30
```

When converting from an IWS integer tag to an IWS tag, this is really not a problem since VBScript variables are type variant. For example:

```
$MyInt = 400                 ‘ Store a vale larger than 255 (the Byte limit)
a = CByte (10)               ‘ store as a byte subdata type
a = $MyInt                   ‘ a will equal 400.
```

IWS Strings

In IWS, strings are up to 255 in length, while VBScript strings can be virtually unlimited in length (limited by available memory only). During the conversion from a VBScript string variable to an IWS string, any characters beyond the first 255 will be truncated. For example:

```
a = “ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789”
a = a & a & a & a & a & a & a & a & a & a     ‘ String is 360 characters long
$MyStr = a                   ‘ Store string in IWS string
                             ‘ Result is 7 strings of a + “ABC” for total of 255 characters
```

In most cases, this string length difference is not of material significance. However, certain ActiveX Controls can be used for block transfer of data to real-world devices and strings are ideal for forming variable length data blocks. The string can then be parsed to extract the data of interest.

IWS Classes

IWS Classes are simply user-defined compound tags that can be made up of one or more IWS tag type. The IWS Classes and Tags are defined in the Database Worksheet. For example, if we define a IWS Class (under the Classes Folder in the Database worksheet) called MyClass with the following elements

```
MyClass
    Item1       Integer
    Item2       Integer
    Message     String
```

Next, a Class tag is created (in the Application Tags Folder)

```
Cls1    MyClass
```

Finally, in VBScript, we can refer to the elements in the Class tag as follows:

```
$Cls1.Item1 = 10
$Cls1.Item2 = 20
$Cls1.Message = “Hello World”
```

IWS Arrays

Using the Class example from above, if (in the Application Tags folder) we had declared the variable Cls1 to have a size of 10, this would be an array with 11 elements. [Remember that the count starts at 0, not 1].

In VBScript, we would refer to the elements in the Class array tag as follows:

```
$Cls1[1].Item1 = 10
$Cls1[1].Item2 = 20
$Cls1[1].Message = “Hello World”
```

We can also use a VBScript variable for the index of the Class array tag. For example:

```
Dim i
i = 1
$Cls1[i].Item1 = 10
$Cls1[i].Item2 = 20
$Cls1[i].Message = “Hello World”
```

Notes:

- IWS tags can be added through the VBScript interface. Simply type a “$” followed by a valid IWS name, and when the Script is Saved, Closed or Check Script function invoked, the programmer will be prompted to create new IWS tag(s).
- VBScript variables and IWS variables can be passed to each other.
- Watch for string length differences with IWS (max. 255 characters) versus VBScript (no limit).

Accessing ActiveX Objects from VBScript

Any of the VBScript interfaces relating to a Screen (i.e. Screen Script, Command Dynamic, and/or ActiveX Events) can directly access the Properties and Methods of an ActiveX control (OCX) that is inserted on a screen.

Using ActiveX Controls is fairly straight forward. First, the ActiveX controls must be registered (i.e. the Operating System Windows Registry must have an entry and Class ID (CLSID) established for the ActiveX Control). Usually when an ActiveX Control is installed in the PC, the installation program will register the ActiveX Control in the final stage of the installation process. If not, registration can be done with one of two methods:

1. Use the Microsoft **RegSvr32** command
    - Invoke the Microsoft Windows **Run** command
    - In the dialog box, type CMD, then OK
    - Type REGSVR32 "C:\<path to OCX control>\<ActiveX Control Name>.OCX", then Enter (Be sure path name is in quotes)
    - If the ActiveX Control registers properly, you will get a message indicated this
    - Close the dialog box
2. Use the **Register Controls** utility provided by IWS (under **Tools** on the main toolbar)
    - Click on **Tools,** then **Register Controls**
    - On the dialog box that pops up, click on **Register**
    - Use the file navigator to locate the ActiveX Control that you want to register
    - Click on Open.
    - Click on Close in the Register Controls dialog box.

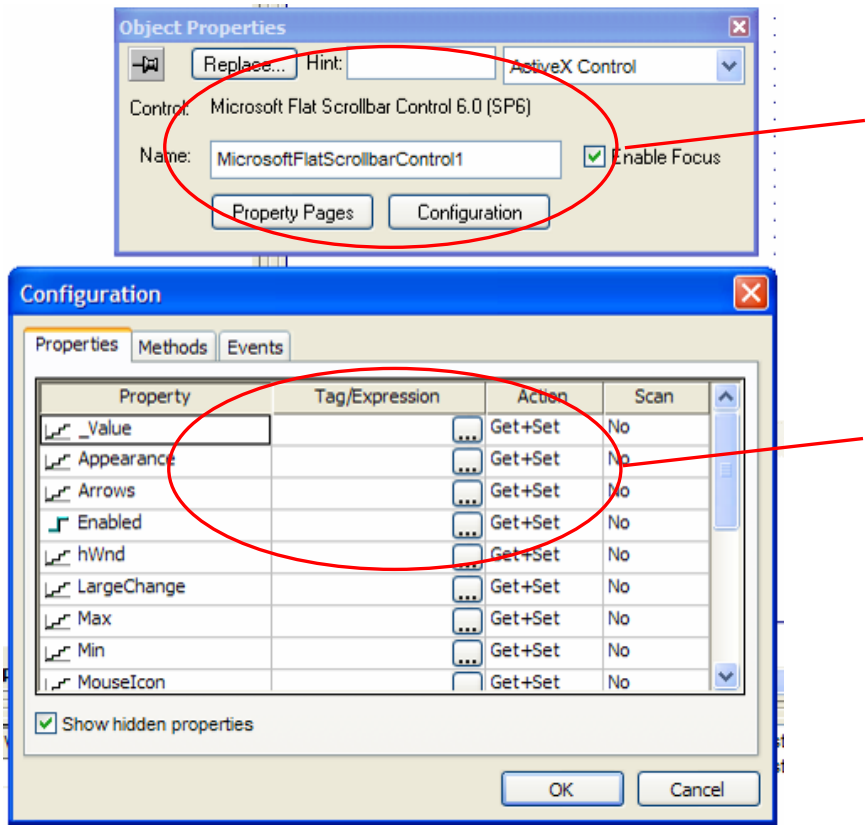
You can also use the IWS **Register Controls** utility to verify that the ActiveX Control has been registered. Beware that the registered name and the file name may not be the same, and in many cases they are not. The best way to verify the control is properly registered is to examine the path of the registered Control.

When the ActiveX Control has been registered, it can be inserted onto a display screen using either the OCX tool in the IWS toolbar or by using the Insert -> ActiveX Object from the top toolbar. A dialog box will appear with a scrolled list of ActiveX objects that are available. Insert the ActiveX object(s) that are appropriate for the application by clicking on OK. IWS will automatically assign a unique name to the ActiveX control. You can use this name or change it, the only requirement being that it must be unique from other ActiveX controls.

Now that an ActiveX Control has been placed on the Screen, any VBScript interface associated with that screen can access the ActiveX Control. These VBScript interfaces are limited to the Screen Script, Command Dynamic for objects located on the same Screen, and ActiveX Event Handler for other ActiveX objects located on the same Screen.

For example, Microsoft has an ActiveX scrollbar control called "MicrosoftFlatScrollBarControl 6.0 (SP6)". Assuming this was inserted for the first time onto a Screen in an IWS application, IWS would likely name this Control "MicrosoftFlatScrollbarControl1". For brevity, let us rename this to "MFSC1". I could easily click on the ActiveX Control on the screen to access its Property Pages, Properties, Methods and Events.

Note that Property Pages and Events are not accessible through the VBScript Interface, although a VBScript Interface is available with the ActiveX's Event Handler. Only an ActiveX Control's Properties and Methods are available from VBScript as implemented in IWS. By clicking on the object to get the Object Properties dialog box



ActiveX Control Name established by IWS. You can rename this Control.

Use this interface to tie Properties and Methods to IWS tags.

To access the ActiveX Control's Properties and Methods from VBScript, you simply type the name of the Control, followed by a Period "**.**" and then the Property or Method. You will need to reference documentation from the developer of the ActiveX Control to determine which properties are setting (Set) or retrieving (Get), and the functioning of the Methods available. For example, with the Microsoft scroll bar control, we access Properties using the following code:

```
MFSC1.Min = 0                ' Set the min value of the scroll bar to 0
MFSC1.Max = 100              ' Set the max value of the scroll bar to 100
$LocTag = MFSC1.Value        ' Get the current location of the scroll bar, pass to IWS tag
```

IWS tools such as **Position** and **Command** can be used with ActiveX controls. To enable these tools, insert the ActiveX control on the Screen and then make sure the ActiveX control is selected (highlighted). Then, select the **Position** or **Command** tool. For example, with the **Position** tool, you can control the visibility of the ActiveX Control, or change its location on the screen.

Notes:

- All ActiveX Controls must have a unique name
- When referencing an ActiveX object name that has been inserted on a screen, note that the reference is case-sensitive from VBScript.
- Only ActiveX Properties and Methods can be accessed via VBScript. Event handling must be set-up by configuring the object (i.e. right click on the object)
- ActiveX Controls can only be accessed by VBScript interfaces associated with the Screen which contains the ActiveX Control (i.e. Screen Script, Command Dynamic, ActiveX Event Handler)

IntelliSense

The VBScript Editor provides a useful tool called IntelliSense, a feature first popularized in Microsoft Visual Studio. Intellisense can be thought of providing “auto-completion” based on the language elements, variables and class members, as well as a convenient listing of available functions. As the developer

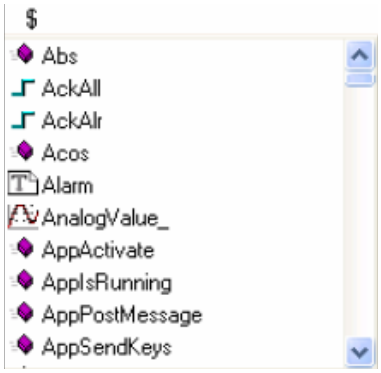
IntelliSense the dialog box can display the following:

- VBScript Functions
- ActiveX Controls, Properties and Methods (the ActiveX Control must be inserted on the Screen where the Screen Script, Command Dynamic or ActiveX Event is used)
- IWS tags and tag fields.
- IWS built-in functions

As the programmer begins to type and characters are recognized, IntelliSense may turn on. If not, the programmer can activate IntelliSense by pressing the Ctrl key plus the Spacebar (“Ctrl” + “ “). By typing a “$” at the beginning of a line, this allows access to IWS tags and built-in functions to be referenced.

When IntelliSense is activate, a pop-up box will appear. The contents of the pop-up box depend on what the programmer has already typed. Sample IntelliSense pop-up dialogs are shown below:

IntelliSense Dialog for VBScript Functions



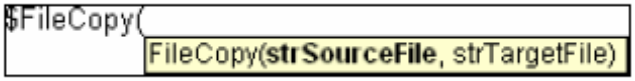
IntelliSense Dialog for IWS Functions

IntelliSense Dialog for IWS Tag Fields

IntelliSense uses different Icons to indicate the type of item that is being referenced. Some Icons are used to indicate different items, so it is important to notice what object is being referenced (i.e. is it an IWS tag, ActiveX Control, VBScript function, etc.)

IntelliSense Icon	Use
	IWS Boolean Tag
	IWS Integer Tag
	IWS Real Tag
	IWS String Tag
	IWS Class Tag
	VBScript Function, built-in IWS function, or ActiveX Control Method
	ActiveX Control Property, VBScript Constants

For many of the functions (both VBScript functions and IWS built-in functions), IntelliSense provide a Parameter Quick Info pop-up dialog. This pop-up dialog may appear once the VBScript or IWS function is entered. An example is:



Notes:

- Use the Ctrl key plus Spacebar key (“Ctrl” + “ ”) to activate IntelliSense. Doing this on a blank line will show all available VBScript functions and any ActiveX controls available.
- Use the Ctrl key plus Spacebar key (“Ctl” + “ ”) to auto-complete any VBScript function, IWS tag, IWS tag field, IWS Class or Class Member, IWS built-in function, or ActiveX Control name, Property or Method once enough of the characters have been entered so that the reference is no longer ambiguous.
- Typing a “$” at the beginning of a line will invoke IntelliSense, referencing existing IWS tags and built-in functions
- Typing the name of an IWS tag, followed by the minus key “-“ plus a greater than arrow key “>” will open the list of available fields for the IWS tag

VBScript with Web Thin Clients

In a Web Thin Client environment, the browser serves as the host for both HTML web pages published by the IWS Server, as well as the host for VBScript code segments that are associated with a particular Screen or object on the Screen. Generally, Microsoft Internet Explorer serves as the browser in a Web Thin Client environment. A InduSoft ActiveX Control (ISSymbol) is used to coordinate communications between the IWS Server and a Web Thin Client.

In a Windows XP/2000/NT-based Web Thin Client environment, Microsoft Internet Explorer (e.g. Version 6 or later) supports VBScripts and ActiveX by default. In a Windows CE-based Web Thin Client environment, Microsoft Internet Explorer (typically provided with PocketPC products) supports both VBScript and ActiveX, but VBScript support must be enabled in the Windows CE image (part of the Platform Build process, typically done by the hardware supplier). Windows CE systems with Microsoft Pocket Explorer (different that Microsoft Internet Explorer) will not work with VBScript as Pocket Explorer does not support VBScript due to memory limitations. Also remember that any ActiveX controls used on a Windows CE Web Thin Client must be developed to support Windows CE.

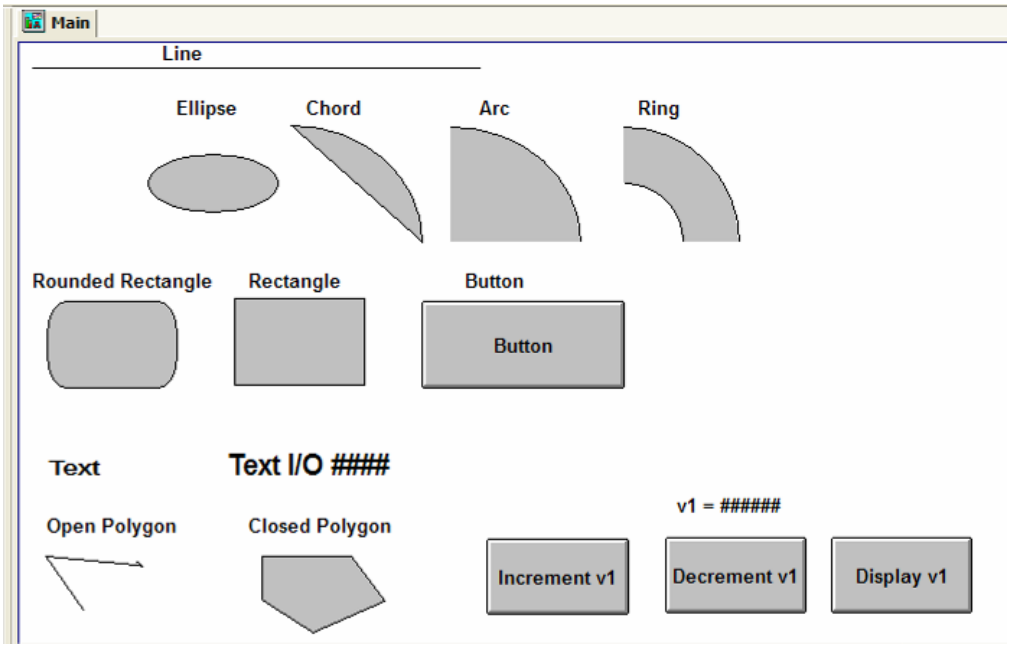
VBScript Interface	Functioning related to a Web Thin Client
Global Procedures	VBScript Global Procedures are accessible to VBScript code segments that execute on a Web Thin Client
Graphics Module	Operates on IWS Server PC only. Procedures and Variables not accessible to a Web Thin Client.
Screen Scripts	This VBScript interface (for a Web Page) executes independently from the VBScript Interface for a Screen running on the IWS Server. • The Graphics_OnStart() subroutine starts when the Web Thin Client Station is successfully logged in and ISSymbol is hosted on the Web Browser • The Graphics_WhleRunning() subroutine executes on the Web Thin Client while the Web Thin Client remains logged in and the ISSymbol Control remain hosted on the Web Browser • The Graphics_OnEnd() subroutine is executed once the Web Thin Client logs off or the ISSymbol Control is no longer hosted by the Web Browser
Command Dynamic	This VBScript interface (for a Web Page) executes independently from the VBScript Interface for a Screen running on the IWS Server.
ActiveX Event Handler	This VBScript interface (for a Web Page) executes independently from the VBScript Interface for a Screen running on the IWS Server.
Background Task Startup	Operates on IWS Server PC only. Procedures and Variables not accessible to a Web Thin Client.
Background Task Scripts	Operates on IWS Server PC only. Procedures and Variables not accessible to a Web Thin Client.

Notes:

- Under Windows XP/2000/NT, to check or modify Internet Explorer's settings for support of VBScript and ActiveX Controls, open Internet Explorer, then click on Tools → Internet Options → Security → Custom Level.
- All VBScript interfaces unique to the Web Thin Client continue to have access to IWS tags and IWS built-in functions.
- When using a Windows CE device for the Web Thin Client, be sure ActiveX support and VBScript support is enabled. This is a function of the Windows CE OS image built using Microsoft Platform Builder.
- When using a Windows CE device for the Web Thin Client, verify that MsgBox and InputBox functions are enabled in the Windows CE OS image if you intent to use them.

Exercise: Using VBScript with Screens

In this exercise, we will create 4 Objects; 3 button objects and a Text I/O object. The buttons will increment, decrement and display the value of Tag v1.



- ☐ Open the **Main.scr** Screen
- ☐ Using the Button tool from the Static Object toolbar, add 3 Button Objects.
- ☐ Add the Command Dynamic to each Button Object.
- ☐ Set the following properties to the buttons:

  Button1:
  - Caption: Increment v1
  - Command Dynamic: $v1 = $v1 + 1 (Click on **Config…**, choose VBScript and put in the **On Down** tab)

  Button2:
  - Caption: Decrement v1
  - Command Dynamic: $v1 = $v1 + 1 (Click on **Config…**, choose VBScript and put in the **On Down** tab)

  Button3:
  - Caption: Increment v1
  - Command Dynamic: Msgbox "v1 = " & $v1 (Click on **Config…**, choose VBScript and put in the **On Down** tab)

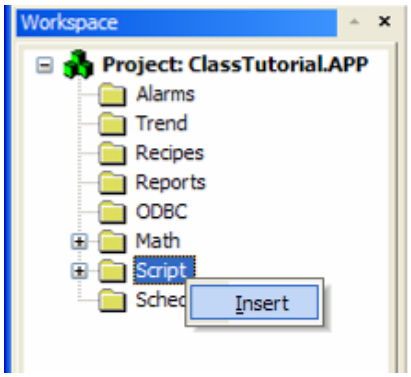
- ☐ Add a Text Object, caption is **v1 = ######**
- ☐ Add the **Text I/O** Dynamic Property to the Text Object. Put **v1** in the Tag Expression field.
- ☐ Save and Close the Screen

Exercise: Using VBScript & Script Worksheets

This Exercise demonstrates a VBScript alternative to a previous Exercise where we used the IWS Scripting Language to create a Field Process Simulator.

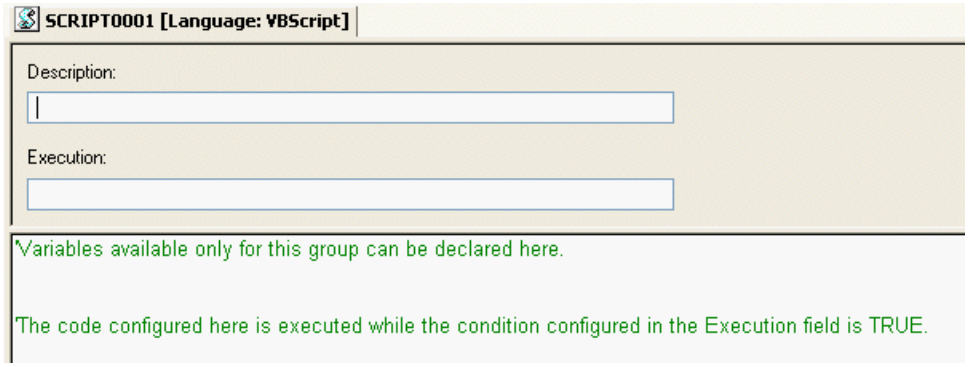
Use the following procedure to configure a Script worksheet:

- ☐ In the **Workspace**, select the **Tasks** tab and then right-click on the **Script** folder.
- ☐ When the pop-up menu displays, select the **Insert** option:



Opening a Math Worksheet

A blank Script worksheet displays:



Blank Math Worksheet

The header portion of the worksheet contains the following fields:

Description field: Provides space for an *optional* description of the worksheet.

Execution field: Controls the script execution. You can type a full expression here, a simple condition, a tag name, or a value. Whatever it is, the script will execute while it is TRUE.

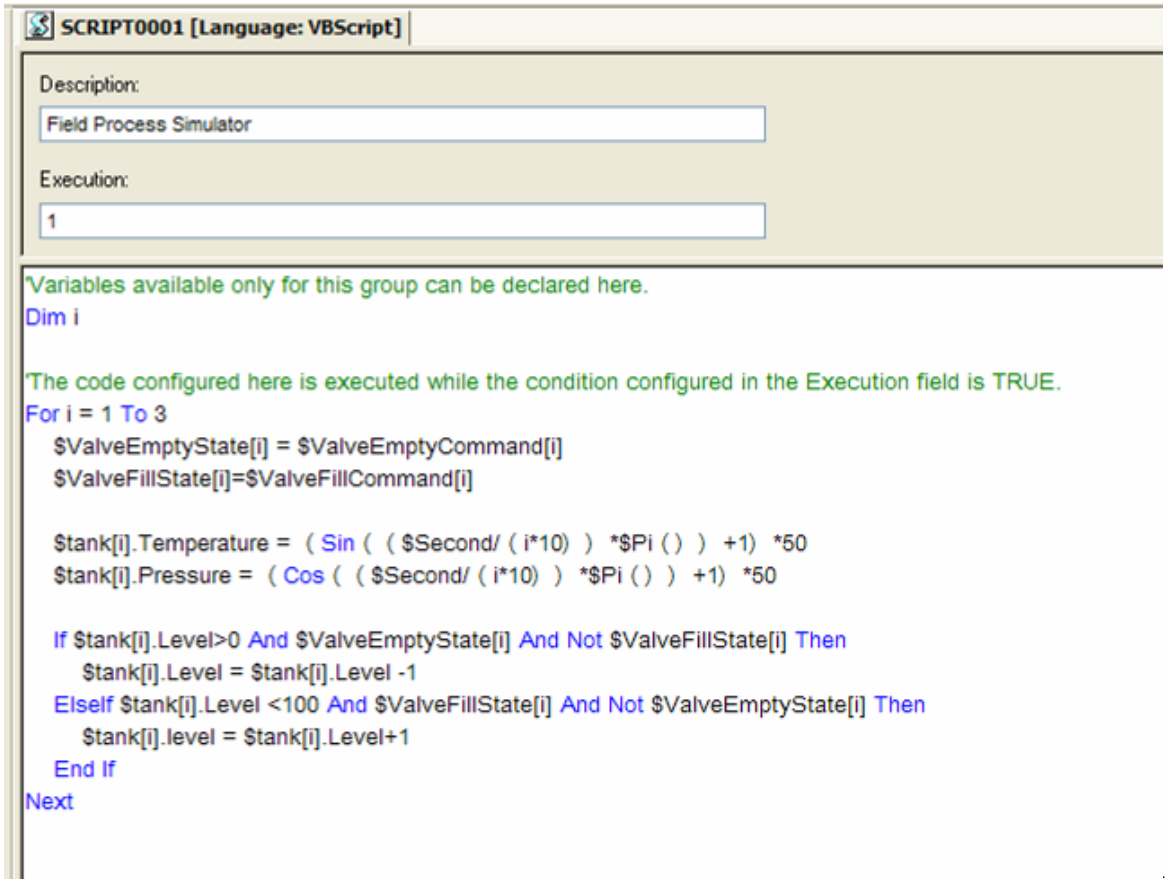
The body section consists of a plain text field into which you can type VBScript.

- ☐ In the **Description** field, type **Field Process Simulator**.
- ☐ In the **Execution** field, type **1**. This enables continuous execution of this Math worksheet; the value 1 is the TRUE condition.

Next, to generate simulated process data that can be used by the various screens, we must set the following variables:

- Valve status (according to the command given)
- Temperature, pressure, and level for three tanks
- Simulated valve status (just transfer the value from the command tags to the status tags).
- Simulated temperature and pressure properties for each tank (configure these properties using *sine* and *cosine* trigonometric functions)
- Simulated level properties for each tank (remember that both the Fill and Empty valves allow for the same flow).

Given this information, complete the body of the Script worksheet as follows:



Completed Script Worksheet

Notes:

- In the preceding example, the Script worksheet is executed continuously. In a real world application, we strongly recommend that the execution of each Script worksheet be carefully controlled to improve system performance.
- The Execution Field in the Header can usually be set to any non-zero value to execute the script. Depending on the **BooleanTrueAboveZero** setting (set in the Application File), this value must either be >0 or non-zero.
- **BE SURE TO ONLY USE EITHER THE MATH WORKSHEET OR THE SCRIPT WORKSHEET, NOT BOTH**

Notes

Chapter 12. Trends

The *Trend* task keeps track of process variables' behavior. IWS allows you to store samples in a history file, and to show both history and on-line samples on a screen trend graph.

To display a trend graph on the screen, you must create a Trend Control object. To store the historical variable behavior, you must create a Trend worksheet.

Trend Control Object

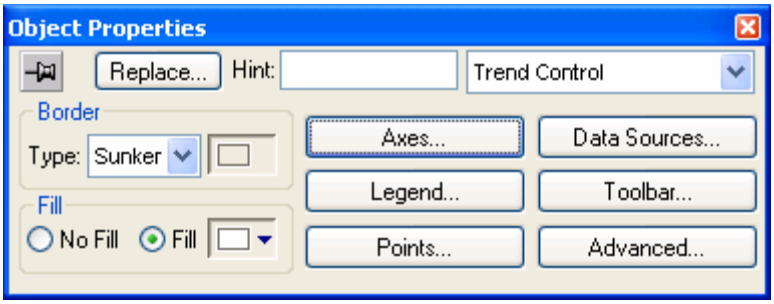
The Trend Control object displays data points (values) from different data sources in a graphic format. The main features provided by the Trend Control object are:

- Display of multiple pens simultaneously
- Support for different Data Sources, such as Tag, Batch, Database and Text File
- Capability to generate X/Y graphs from the configured data sources
- 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 that provides interfaces for the user to interact with the Trend Control object during the runtime
- Built-in legend that displays the main information associated to each pen linked to the object
- Zooming and auto-scaling tools
- Horizontal and vertical orientation

Trend Control Development Interface

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:



Trend Control Object Properties Dialog

Border 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 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:

Axes Opens a dialog box that allows you to configure the settings for the X and Y axis.

Data Sources Opens a dialog box to define 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.

Legend Lets you choose whether to show an embedded legend during the runtime, and if so, which fields to display in it.

Toolbar Opens a dialog box that lets you choose whether to show an embedded toolbar during the runtime, and if so, which buttons that trigger actions to display in it.

Points 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.

Advanced Opens a dialog box that gives you more configuration options.

Trend Control Runtime Interface

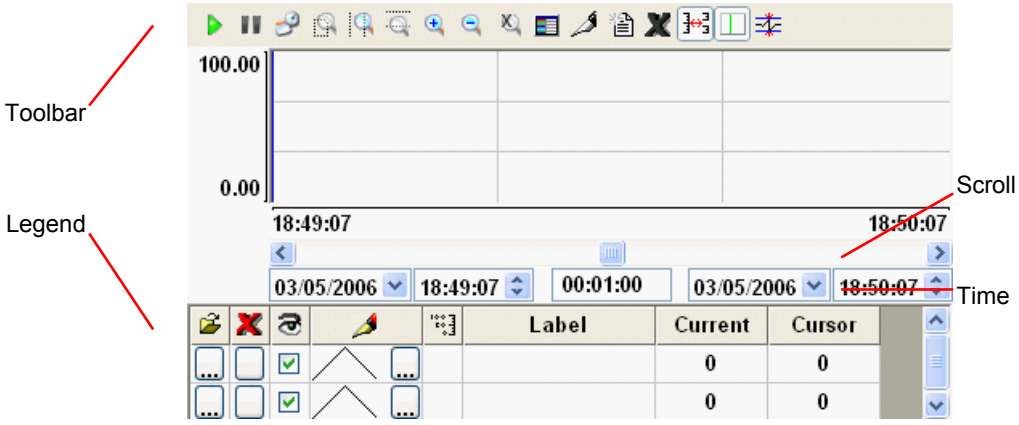
When enabled, some embedded interfaces can help you to interact with the Trend Control during the runtime:

Toolbar Lets you choose whether to show an embedded toolbar during the runtime, and if so, which buttons that trigger actions to display in it.

Legend Lets you choose whether to show an embedded legend during the runtime, and if so, which fields to display in it.

Scroll Bar Allows you to slide through the X-axis values, according to the period configured for this scale.

Time Bar Allows you to 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).



Embedded Interfaces in the Trend Control Object

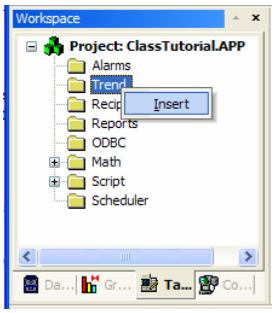
Exercise: Creating a Trend

The online trend you create in this exercise will graph the temperatures of the tanks in real-time, updated every second. While the application runs, trend data are saved to an internal history file, so the trend graph can be scrolled to display any period in the trend history.

CREATING A TREND WORKSHEET

First, you must create a Trend worksheet specifying the Tags that will be saved to the history file and made available to any Trend Control objects in the application:

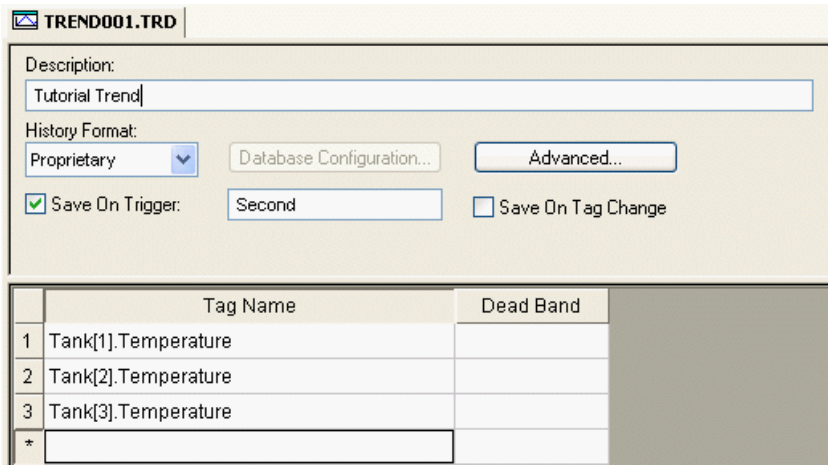
☐ Create a new Trend worksheet by right-clicking on the **Trend** folder (in the **Tasks** tab of the **Workspace**) and selecting **Insert**:



Inserting a Trend Worksheet

Notes:
- You can also create new worksheets using the **New** dialog, which is accessed by selecting File → New or by pressing the **Ctrl+N** keyboard shortcut.
- The Trend Worksheet is used to define where & how the data is to be logged. The Trend Control Object is used to define how the data is to be displayed.

☐ Configure the Trend worksheet as shown:



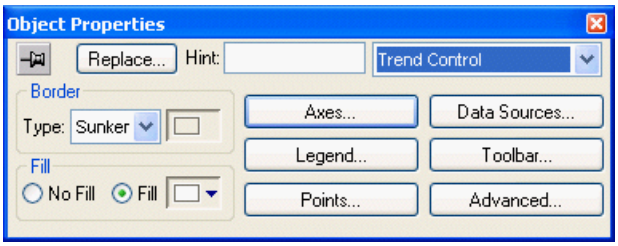
Configuring the TREND001.TRD Worksheet

☐ Save and Close the Worksheet

CREATING THE TREND SCREEN

Next, we will open the **TrendData.scr** Screen and add a Trend Control object. This will be used to display the trend data.

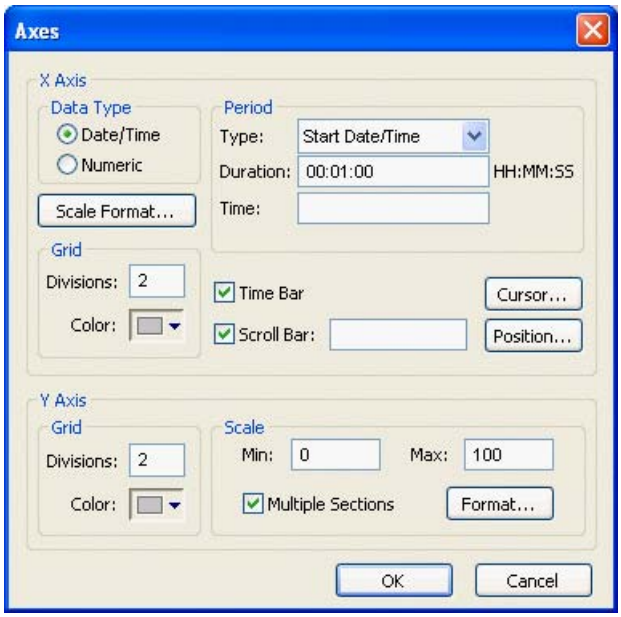
- ☐ Open the **TrendData.scr** Screen
- ☐ Click on the **Trend Control** tool icon in the Active Objects toolbar and add a **Trend Control** Object on the screen. Open its Object Properties dialog:



Trend Control Object Properties Dialog

Because of the large number of properties available on a Trend Control Object, the properties are divided among secondary dialogs.

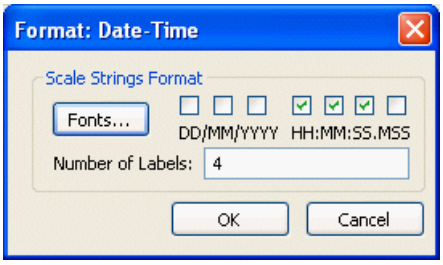
- ☐ Click the **Axes** button to open the Axes dialog:



Trend Control Object Properties — Axes Dialog

The X-axis of the trend graph will be standard Date/Time, but its format must be adjusted.

☐ In the **X Axis** area, click the **Scale Format** button to open the **Format: Date-Time** dialog:

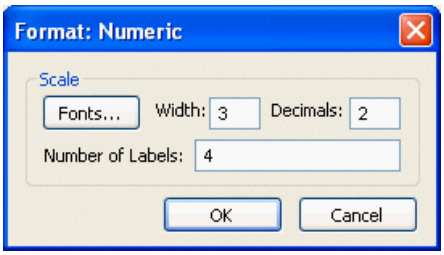


X Axis — Format: Date-Time

☐ Change **Number of Labels** to 4, and then click **OK** to close the **Format: Date-Time** dialog and return to the **Axes** dialog.

The Y-axis format must also be adjusted.

☐ In the **Y Axis** area, click the **Format** button to open the **Format: Numeric** dialog:



Y Axis — Format: Numeric

☐ Change **Number of Labels** to 4, click **OK** to close the **Format: Numeric** dialog and return to the **Axes** dialog.

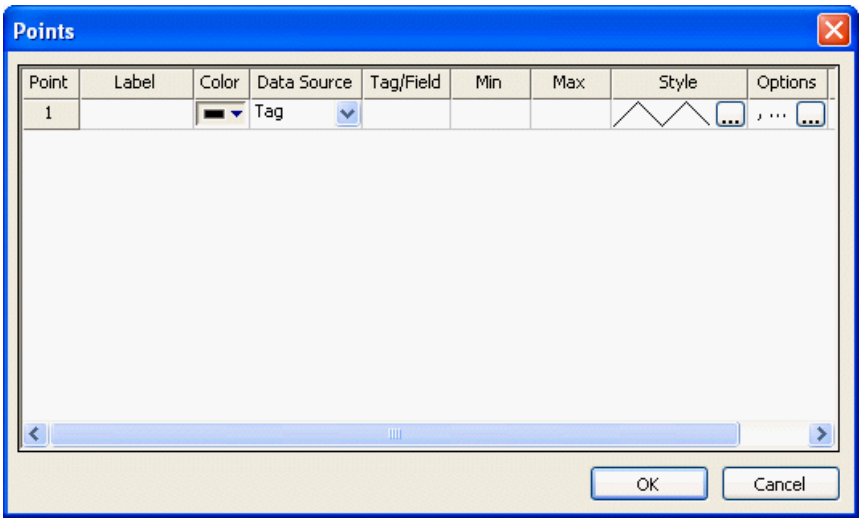
By default, multiple trend data points are each displayed in their own sections of the trend graph. However, in this exercise the data points should be displayed together in the same section, so that they can be compared as they progress.

☐ In the **Y Axis** area, click the **Multiple Sections** checkbox to deactivate it (it should be unchecked).

☐ Click **OK** to close the **Axes** dialog and return to the Object Properties dialog.

Next, the data points themselves must be configured.

☐ In the Object Properties dialog, click the **Points** button to open the Points dialog:



Trend Control Object Properties — Points Dialog

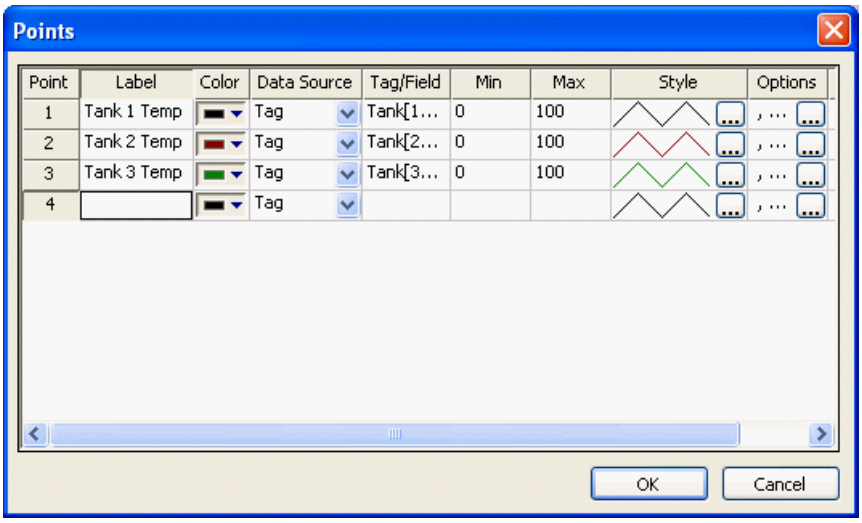
☐ Configure three data points as follows…

Point 1:
Label: Tank 1 Temp
Color: Black
Data Source: Tag
Tag/Field: tank[1].Temperature
Min: 0
Max: 100

Point 2:
Label: Tank 2 Temp
Color: Dark Red
Data Source: Tag
Tag/Field: tank[2].Temperature
Min: 0
Max: 100

Point 3:
Label: Tank 3 Temp
Color: Dark Green
Data Source: Tag
Tag/Field: tank[3].Temperature
Min: 0
Max: 100

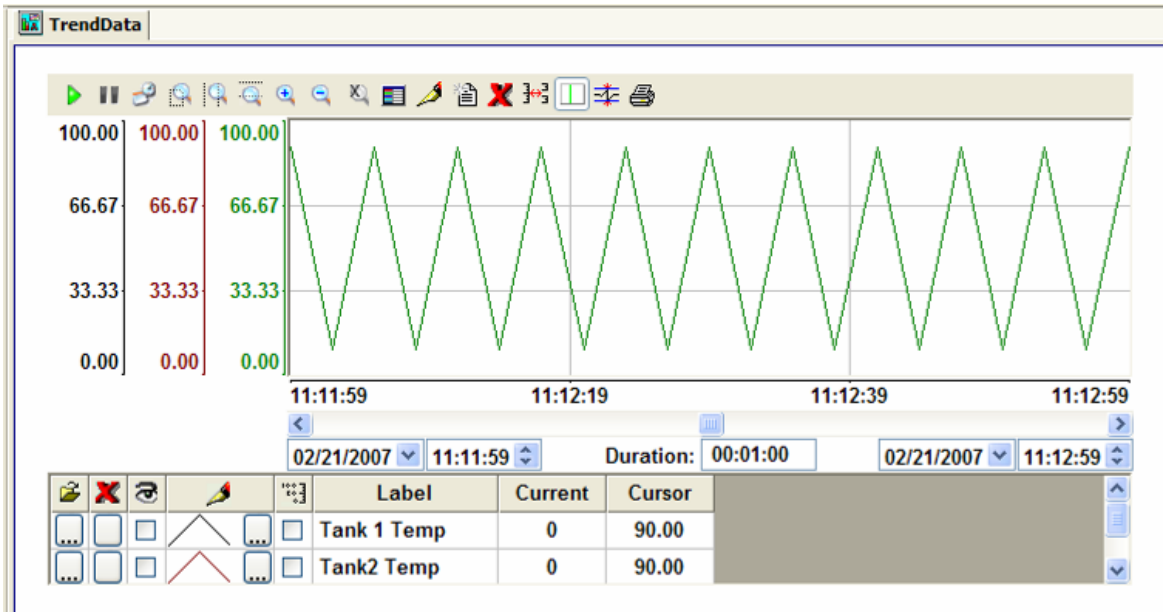
When you are done, the Points dialog should look something like this:



Points Dialog with Three Points Configured

☐ Click **OK** to close the Points dialog, and then close the Object Properties dialog.

The finished **TrendData** screen should look something like this:



Trend Online Screen

☐ Save and Close the Screen

If you run the application now, the trend graph will display (in real-time) the simulated data generated by the field process simulator that you created in the previous chapter.

Decreasing the Save Frequency Using the Scheduler

As it is now configured, the Trend Worksheet saves to the history file and the Trend Control Object is updated every second. While this makes for a smooth graph, it also decreases system performance and generates extremely large historical data files.

You can decrease the save frequency by reconfiguring the **Save On Trigger** field (in the Trend worksheet) with a different tag or expression. The field is currently configured with the **Second** tag, which is one of the standard Internal Tags in IWS. Every time the value of the tag changes — which is every second — the save is triggered.

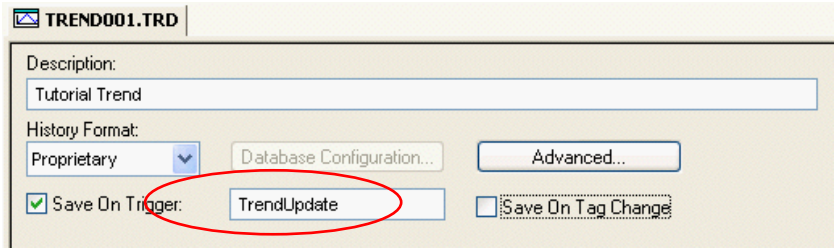
You could easily reconfigure the **Save On Trigger** field with another Internal Tag (e.g. **Minute**, **Hour**, **Month**, and so on), but this may trigger saves too infrequently to be useful.

To trigger saves at a nonstandard frequency, use the **Scheduler**:

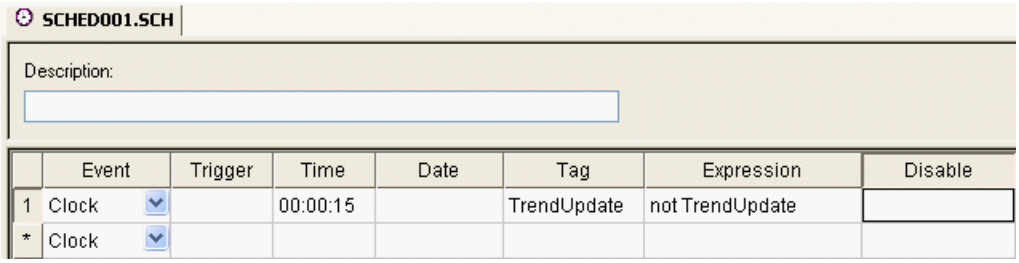
Create a new Tag in the Application database, for example:

Name: TrendUpdate
Size: 0
Type: Boolean
Description (*optional*)
Scope: Local

Open the Trend worksheet and reconfigure the *Save On Trigger* field with this new tag:



Create a new Scheduler worksheet (in the **Tasks** tab of the **Workspace**), and then add the following line:



Configuring the SCHED001.SCH Worksheet

This line will cause **TrendUpdate** to toggle every 15 seconds. **TrendUpdate** was defined as a Boolean, and the expression changes the value of the tag to its opposite state: from FALSE to TRUE, then from TRUE to FALSE, and so on. Every time the value of the tag changes, a save is triggered.

Save and close both worksheets.

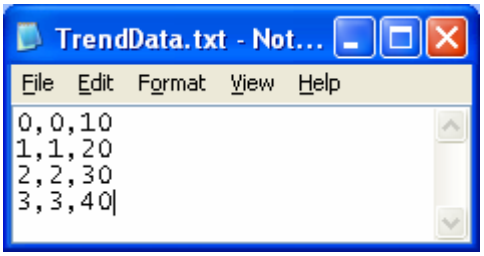
Using an External Text File

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.

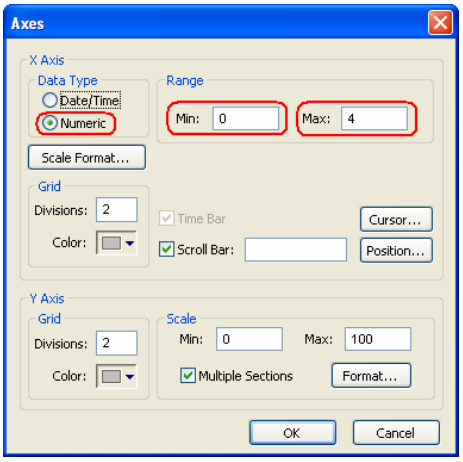
The first step is to convert the data into a text file. If we adopt the comma as our separator the file will be as shown below:



Creating a Text File Using Notepad.exe

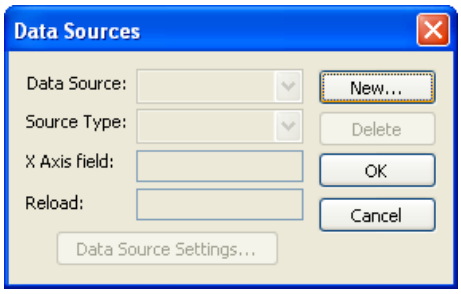
Note:
- You should save the data file in the same folder where the application is. By doing so, you do not have to specify the entire path and your application will still work, even if it is copied to a different computer.

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:



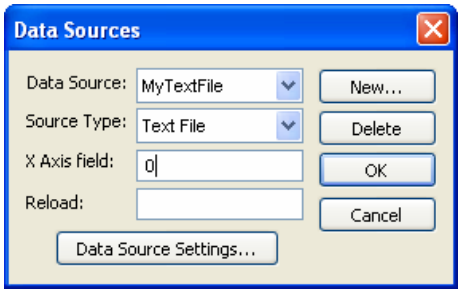
Configuring the X Axis for Numeric Data

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



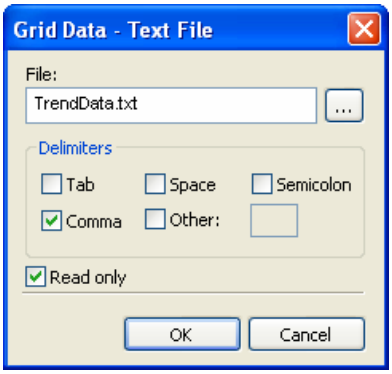
Trend Control Object Properties — Data Sources

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:



Creating a New Data Source (Text File)

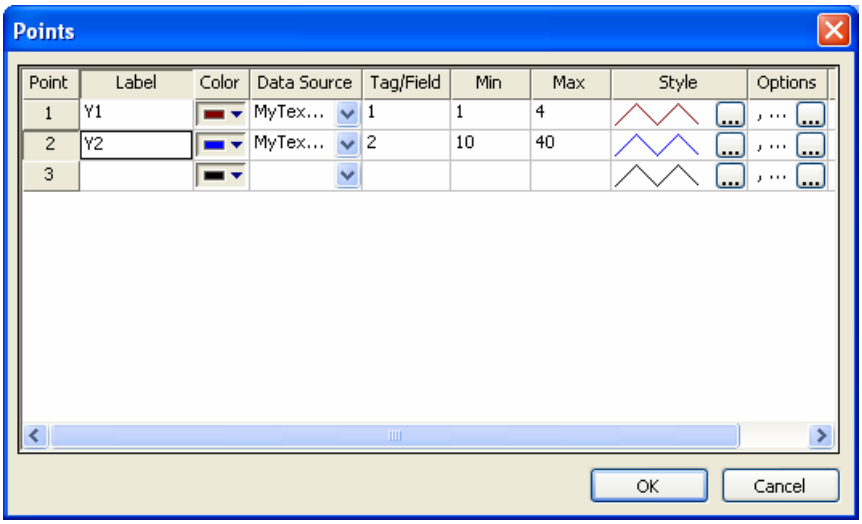
On the **X Axis field** we need to indicate which column in our text file represents the X Axis. In our example we are using column zero, so enter with zero for this field, then click on the button **Data Source Settings**, the following Window will display:



Specifying the Text File Name and Field Delimiter

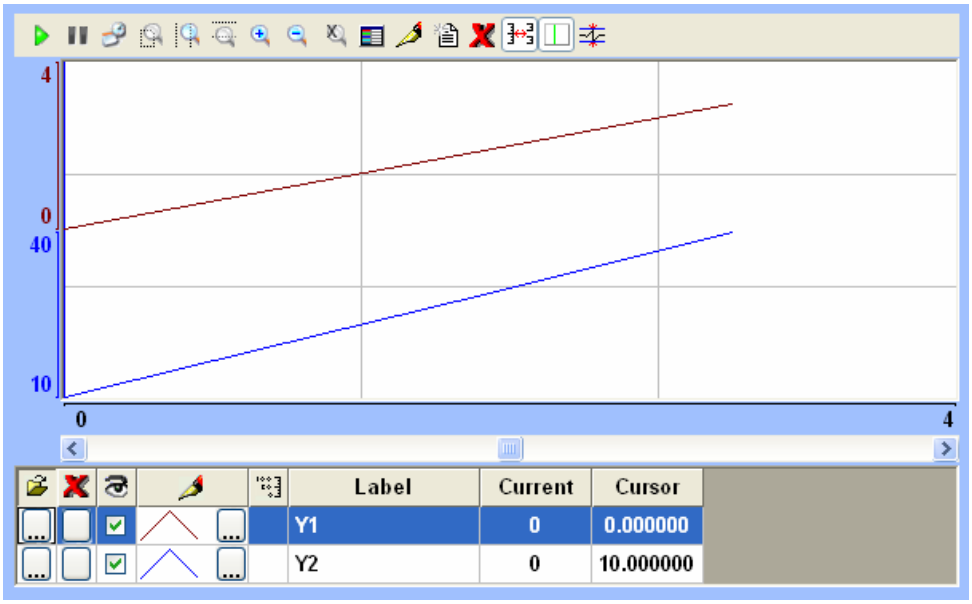
If you have copied the text file to the application folder, you only have to specify the file name, otherwise, enter with the complete path where the file is located (use the browse button ... as needed). Click **Ok** on this window and **Ok** again to finish the data source configuration and close the **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:



Configuring the Data Points (Pens)

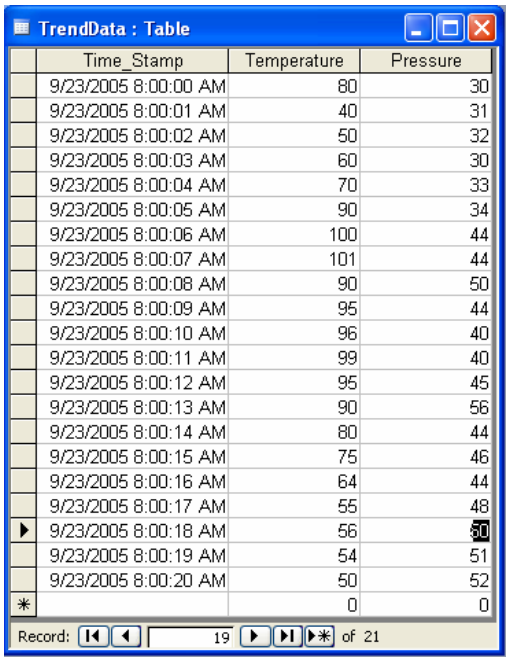
After following these steps, run your application and you should see something similar to the figure below:



Using an External Database

The **Trend Control** Object can generate trend charts from any **Relational Database** that can be accessed through the ADO.NET technology. This Section 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..

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:



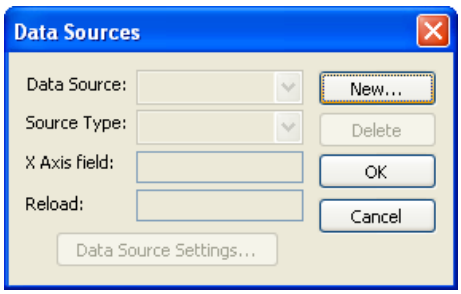
TrendData : Table

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

Record: 19 of 21

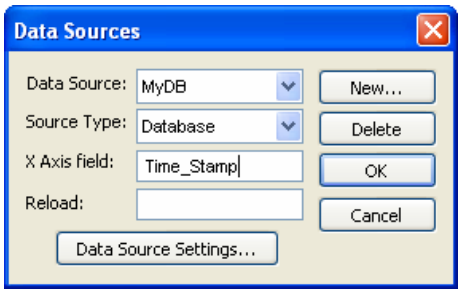
Table in a Relational Database

The first step is to add the Trend Control Object to your screen. Now double click on the object to open then **Object Properties** and click on **Data Sources**. The following window will display:



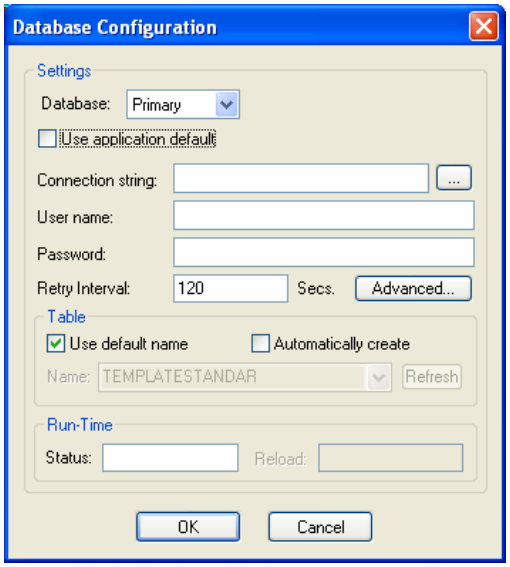
Trend Control Object Properties — Data Sources

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:



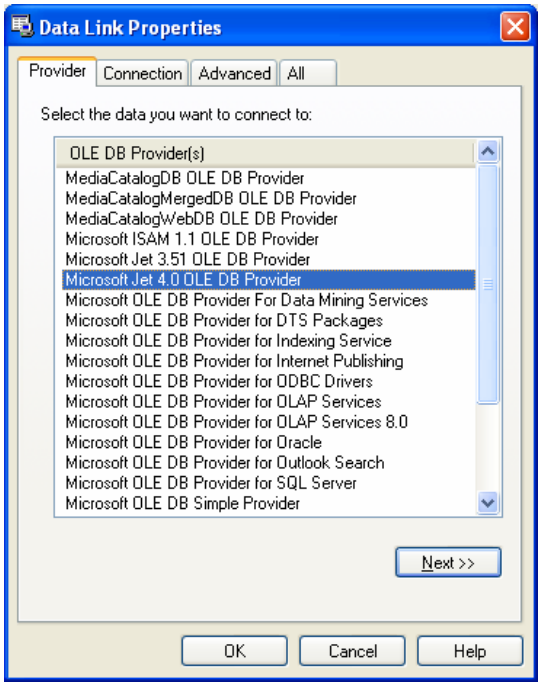
Creating a New Data Source (Database)

Change the Source Type to Database and specify Time_Stamp in the **X Axis field**. Then click on the Data Source Settings button, the following window will display:



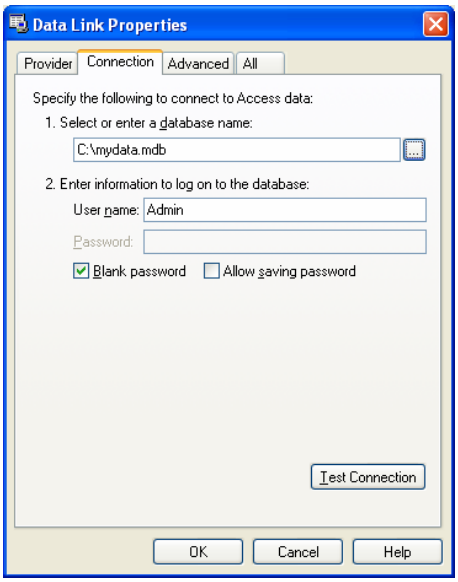
Database Configuration Window

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:



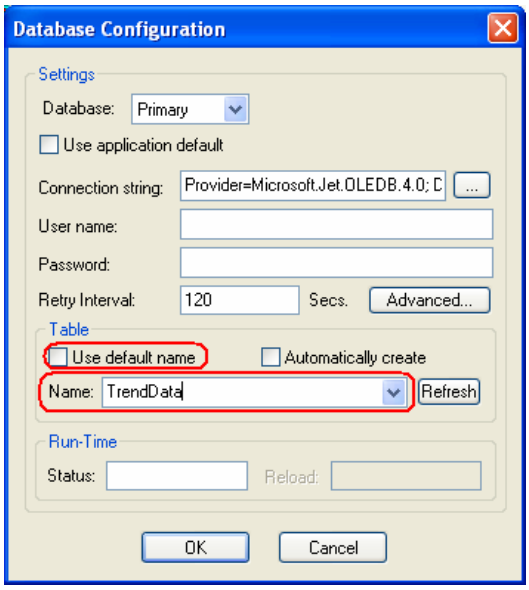
Data Link Properties — Selecting a DB Provider

Select the Microsoft Jet 4.0 OLE DB Provider and click **Next**. In the following window, you should specify the database path:



Data Link Properties — Configuring the DB Connection

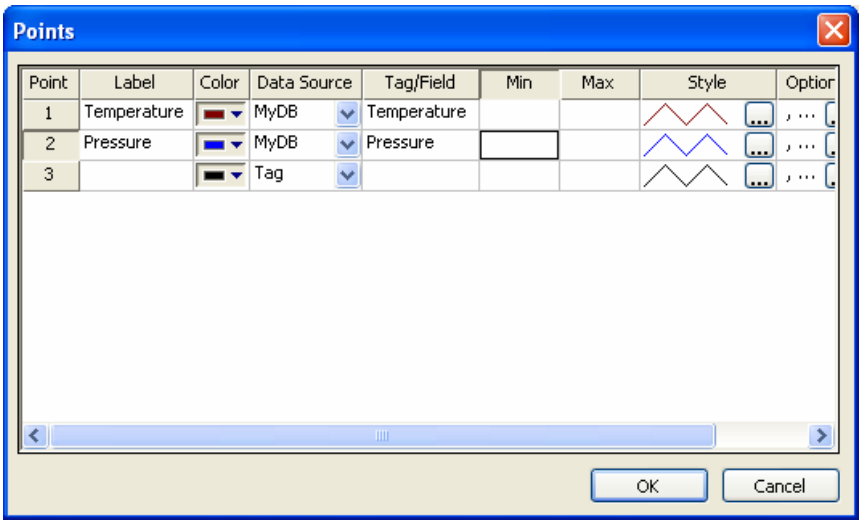
Click **Ok** to finish the Connection String configuration. Now uncheck the option Use default name and select the table from your database as shown below:



Selecting the Database Table

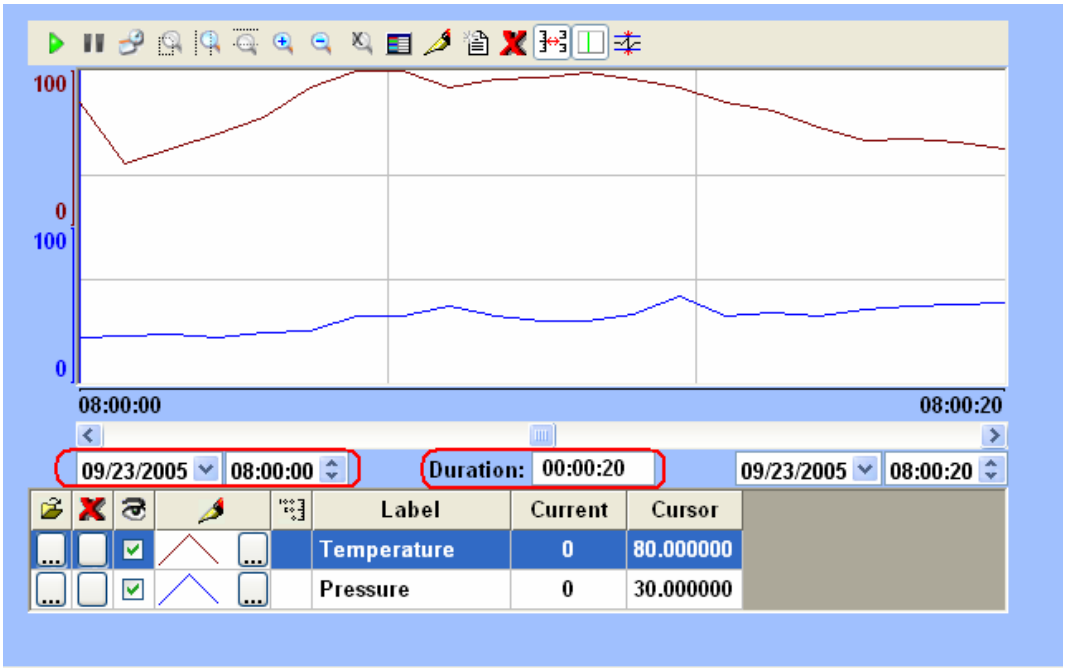
Click **Ok** on this window and **Ok** again to finish the data source configuration and close the **Data Source** configuration window.

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:



Configuring the Data Points (Pens)

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:



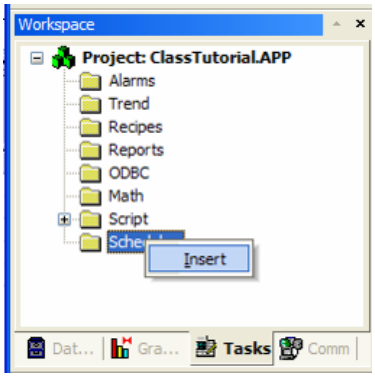
Note:

- If the Tag data was originally logged by IWS into a relational database, and the Trend Worksheet remains connected to the Database, you only need to specify **Tag** as the Data Source. You do not need to configure the Data Source as a Database.

Chapter 13. Configuring a Scheduler Worksheet

The **Scheduler** is an IWS Task that runs in the background. Its purpose is to generate Events with defined mathematical expressions (using the IWS Scripting Language) to be executed based on elapsed time, a date, or any monitored Event. Some example uses of the Scheduler include:

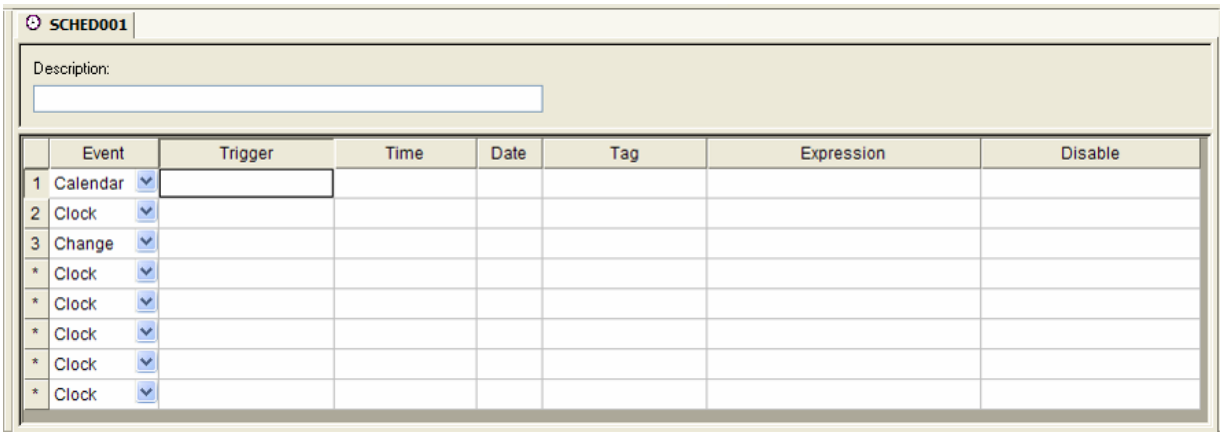
- Automatically saving data at a shift change
- Log out at shift change
- Run a report (hourly, daily)
- Time delay after a Tag changes value
- Setting the trigger period for the Trend Object (and data logging)



The Scheduler is located in the **Tasks** Workspace in the Scheduler folder. To insert a new Scheduler Worksheet, position the mouse on the Scheduler folder and click the right mouse button. Click on the **Insert** option.

Each Scheduler Worksheet is numbered sequentially. Scheduler Worksheets always execute in the background, unlike Math and Script Worksheets. Note that the Scheduler Worksheet has no **Execution** field in the header, whereas Math and Script Worksheets have an Execution field in the header. Having different Scheduler Worksheets is really only for organizational purposes (e.g. you could put all reporting events in one Scheduler Worksheet). The **Description** field is used for documentation purposes.

In the Body of the Scheduler Worksheet there are sequential line numbers, each used to specify a unique Scheduler Event. Each line has several fields, and these will be populated according to the type of Scheduler Event that is being specified. The fields are:



- **Event**
  The Event field is a combo-box that allows you to specify the type of Scheduler Event. There are three (3) Event types:
  - **Calendar**
    The Calendar Event type is generally used when you want to generate an Event that occurs with a time base greater than 24 hours, or you want to schedule an Event on a specific Date. With this Event type, you will use the **Time** and/or **Date** field. If you leave the Date field empty, the Event will occur daily at the time specified in the Time field.

- **Clock**
  The Clock Event type is generally used for Events that occur with a time base less than 24 hours. With this Event type, you will use the **Time** field, not the **Date** field. An example of using the Clock Event type would be to generate a time base (i.e. triggers) for updating the Trend Control Object or data logging.
- **Change**
  The Change Event type is used to trigger an Event when an IWS Tag changes its value. There is no restriction on the value of the Tag, just that it has changed value. Any Tag type will work (Boolean, Integer, Real, String), both Application Tags and Internal Tags. With this Event type, you will use the **Trigger** field. An example of using the Change Event type would be to execute an IWS built-in Function (e.g. a print a Report) when the Internal Tag Hour changes value (signifying a new hour).

- **Trigger**
  The **Trigger** field is used in conjunction with the **Change** Event type. Type the name of an IWS Tag in this field, and when this Tag changes value, the Expression in the **Expression** field will be executed (if the **Disable** field is False). This field is only used by the Change Event.
- **Time**
  The **Time** field is used to specify a time interval in which an Event must occur, relative to the time the IWS Application was started. The Time value is specified in an hours, minutes, seconds and milliseconds (optional) format (e.g. **00:00:30** would be 30 seconds, **00:00:00.1** would be 100 milliseconds). Valid values are 0-23 for hours, 0-59 for minutes, 0-59 for seconds, and .1 - .9 for milliseconds. This field can be used by the Calendar and Clock Event types.
- **Date**
  The **Date** field is used to specify a specific date (i.e. calendar date) when the Calendar Event must occur. The Date value is specified in the MM/DD/YYYY format. Valid values are 01-12 for month (MM), 01-31 for day (DD). Year must be specified using four (4) digits. If this field is left blank, the Event occurs daily. This field is only used by the Calendar Event type.
- **Tag**
  The **Tag** field is used to specify the Tag that will receive the value returned from the **Expression** field. It is not necessary to populate this field. For example, if the Expression field contain an IWS built-in Function which has no return value, or the return value is not used, the Tag field can be empty. This field can be used by all Event types.
- **Expression**
  The **Expression** field is always used, and defines the Event that is to occur based on the Event type and the settings in the Trigger, Time and/or Date fields. If the Expression in the Expression field returns a result, the result value will be stored in Tag specified in the **Tag** field. The Expression can be disabled by the **Disable** field having a logical True value.
- **Disable**
  The **Disable** field can contain an Expression that, if it evaluates to a logical True (>=1), the Expression in the Expression field will not be executed. If the Expression in the Disable field evaluates to zero (Logical False), or if the Disable field is left blank, the Expression in the Expression field will be executed.

Notes:
- The **Time** field can include a millisecond value (minimum of 100 msec). e.g. 00:00:00.1
- The **Tag, Expression and Disable** fields are used for all Scheduler Event types.

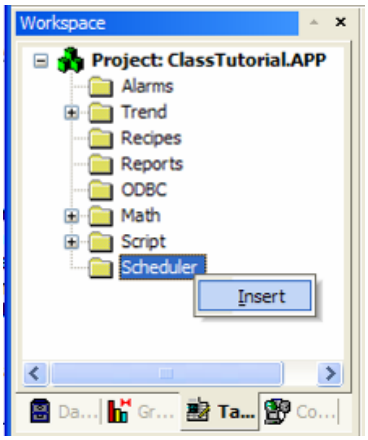
Exercise: Configuring a Scheduler Worksheet

The Scheduler Worksheet can be configured using **Clock**, **Calendar**, and **Change** events.

- You use a **Clock** event to trigger actions based on regular time intervals such as timers and counters. In the **Time** column, you can configure the base time (minimum of 100ms). In the **Tag** column, you must configure a tag to receive the results from the expression configured in the **Expression** column. Finally, you can use the **Disable** field to prevent an expression in the line from being executed. The results of an expression in the **Disable** field will always be TRUE.
- You use a **Calendar** event to trigger actions on a scheduled time. Also, it is possible to specify a fixed date for events in the **Date** column. The **Tag**, **Expression**, and **Disable** columns are used the same in all three-scheduled event functions.
- You use a **Change** event to trigger an action when a tag value changes. In the **Trigger** column, we must configure a tag that will be used to trigger the event when a change in value has occurred. The **Tag**, **Expression**, and **Disable** columns are used of the same in all three-scheduled event functions.

- Use these steps to create a new *Scheduler* worksheet:

☐ Right-click on the **Scheduler** folder:



Opening a Scheduler Worksheet

☐ Configure the *Scheduler* worksheet as follows:

SCHED002.SCH

Description:

	Event	Trigger	Time	Date	Tag	Expression	Disable
1	Clock		00:00:30		TrendUpdate	not TrendUpdate	
2	Change	Hour				Open("Registering")	
3	Calendar		16:00:00			LogOn()	
4	Calendar			01/01/2005		Open("LetsGo")	
5							
6							

Completed Scheduler Worksheet

Chapter 14. Recipes

InduSoft Web Studio allows you to create recipes that can be imported or exported in Text (CSV) or XML (Extensible Mark-up Language) format.

Recipes are often used to load or store machine setup or process parameters. The Recipes are loaded into, or stored from, IWS tags.

Exercise: Creating Recipes

The *Recipes* module allows you to create, load, and delete recipes. Recipes (for the purposes of this class) are a group of tags whose values can be saved and retrieved like a database.

To create Recipes you must create a *Recipe* worksheet. This worksheet tells the system which tag values to store on the disk for retrieval later and where to store the data. When you save a Recipe, IWS creates an ASCII file in standard format or in XML format containing the tag values and the Recipe's file name. When retrieving these tag values, the system locates the values in this ASCII file.

Creating a Recipe Worksheet

For this exercise, we'll use a new Class and Tag to define the "ingredients" in the recipe.

- ☐ Create a new Class (in the **Database** tab of the **Workspace**) named **cCake** and configure it as shown:

Class: cCake

	Name	Type	Description
1	Sugar	Integer	
2	Fruit	Integer	
3	Milk	Integer	
4	Flour	Integer	
5	Yeast	Integer	

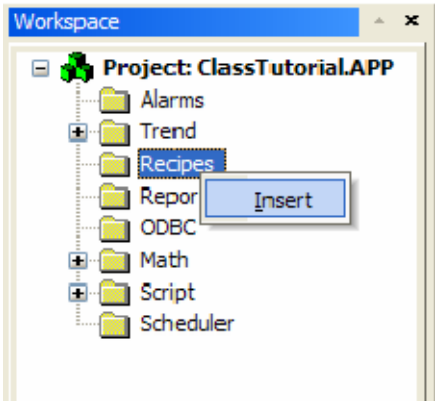
Creating the cCake Class

- ☐ Save (if required) and Close the Class cCake Datasheet View
- ☐ Open the Application Tags Datasheet View
- ☐ Add a Tag named **Cake,** Size 0, Type **cCake**
- ☐ Add a Tag named **RecipeName**, Size 0, Type **String**. This tag will be used to store the name of the Recipe file.

Application Tags

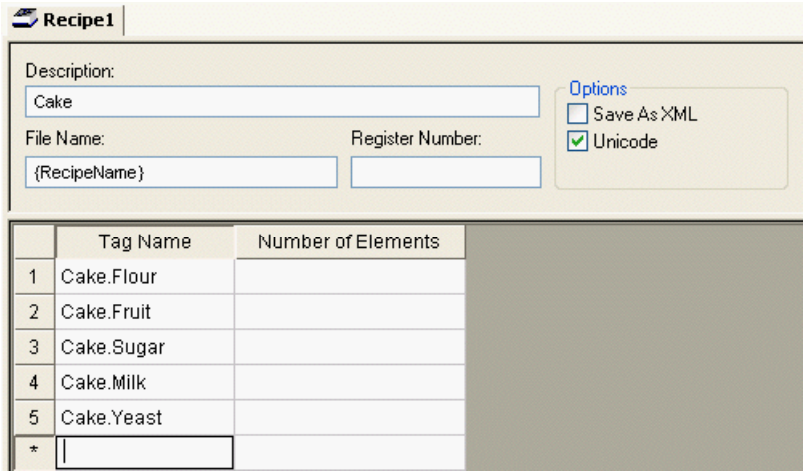
	Name	Size	Type	Description	Scope
1	v1	0	Integer	Variable 1	Server
2	v2	0	Integer	Variable 2	Server
3	v3	0	Integer	Variable 3	Server
4	MyPtr	0	String	String Pointer	Server
5	@My2Ptr	0	Integer	Integer Pointer	Server
6	tank	3	cTank	Array Class Tag	Server
7	TankNum	0	Integer	tank Index	Server
8	ValveFillState	3	Integer	valve fill state	Server
9	ValveEmptyState	3	Integer	valve empty state	Server
10	ValveFillCommand	3	Integer	valve fill command	Server
11	ValveEmptyCom...	3	Integer	valve empty command	Server
12	j	0	Integer	loop counter	Server
13	cake	0	cCake	cake class	Server
14	RecipeName	0	String	Name of recipe	Server
*			Integer		Server

☐ Create a new **Recipe** Worksheet. In the **Workspace**, select the **Tasks** tab and then right-click on the **Recipes** folder. When the pop-up menu displays, select the **Insert** option.



Creating a Recipe Worksheet

☐ Configure this worksheet as shown in the following figure, and save (**File** → **Save** or **Save As**) the worksheet using the default name: **Recipe1.rcp**.

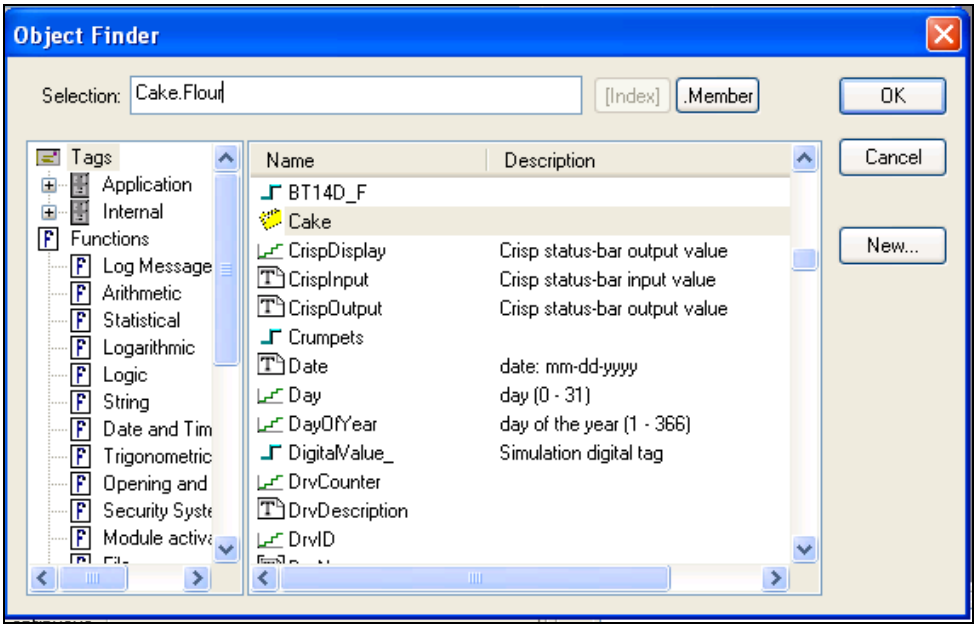


Recipe1.rcp Worksheet

Notes:

- Recipes can be saved (or loaded) in XML format. Simply select the **Save As XML** checkbox in the Recipe Worksheet configuration dialog box.
- Remember that the syntax used to access a value from a class tag is: ***<tag_name>.<member_name>*** (e.g. **Cake.Sugar**, **Cake.Fruit**, etc.).

☐ You can enter the **TagNames** by double-clicking in the **TagName** field, and then selecting the **Cake** tag you created, which will fill in the **Selection** field. Add "**.Flour**" and then select **OK**.



The other fields in the Recipe Worksheet configuration dialog box are:

- **File Name** field
  Enter the name of the file in which to save the recipe tag values.

  If you type a tag name between curly brackets (as in the example), the file will use that tag value to compose the File Name. For example, you could specify a file name such as **c:\AppName\Recipe\{RecipeName}**, where a value contained within the *RecipeName* tag file provides the name of the **c:\AppName\Recipe** directory.

- **Register Number** field
  Contains a tag that defines the register number to be read or written into a DBF file. (**This field is for legacy purposes only and is no longer used).**

- **Number of elements** column
  Specifies how many array tag positions are in use.

  For example, if you want to specify an array tag size 120 in a Recipe, you do not need to type a tag name and index for all 120 positions (**Tag[0]**, **Tag[1]**, **Tag[2]**, and so forth). You just specify a Tag name and type the number of positions you want into the number column (e.g. **Tag 120**).

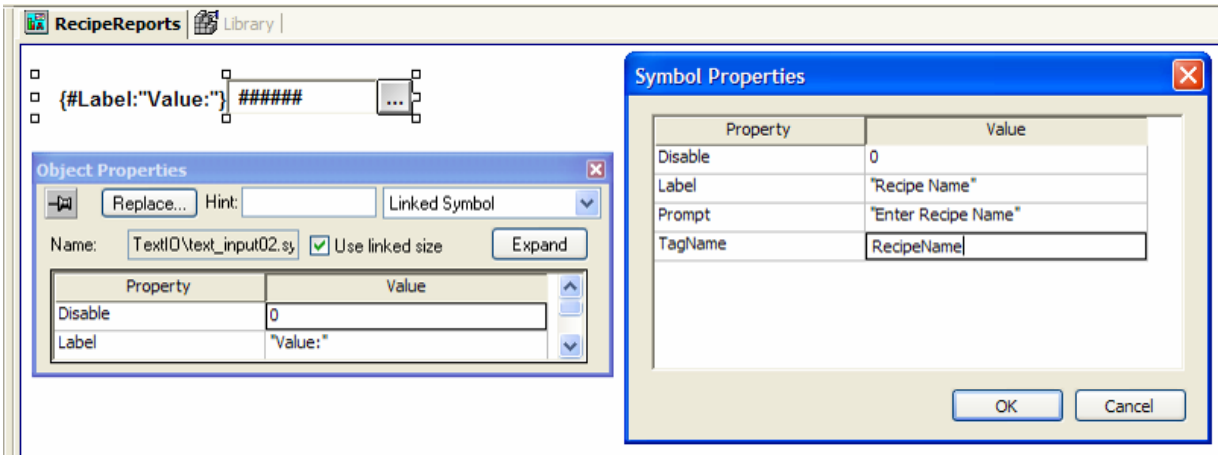
- **Save As XML** check-box
  Saves the file in XML format.

  If you enable the XML option to save this recipe file in XML format, the generated XML files will include all the tag values, along with the tag name from which those values originated. To load this information, you must load it into a tag using the same name.

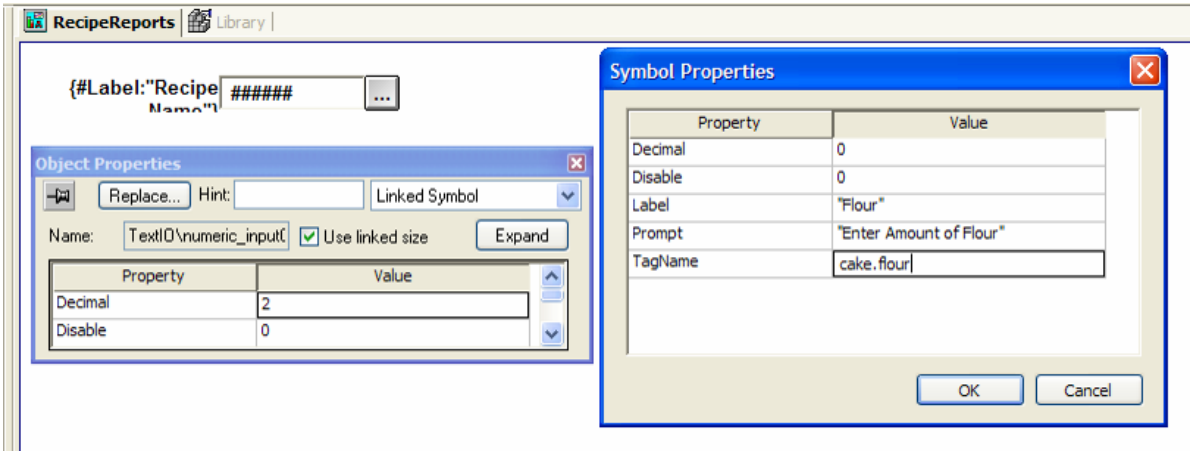
Now, we will proceed to creating a Recipe screen.

Creating a Recipe Screen

- ☐ Open the **RecipeReports.scr** Screen that was previously created.
- ☐ Open the **Symbol Library** and from the **TextIO** folder, insert the Symbol **text_input02**.
- ☐ Double-click on the Linked Symbol in the Application Screen. Click on **Expand.**
- ☐ Enter the information in the **Symbol Properties** dialog box as shown and click **Ok**.



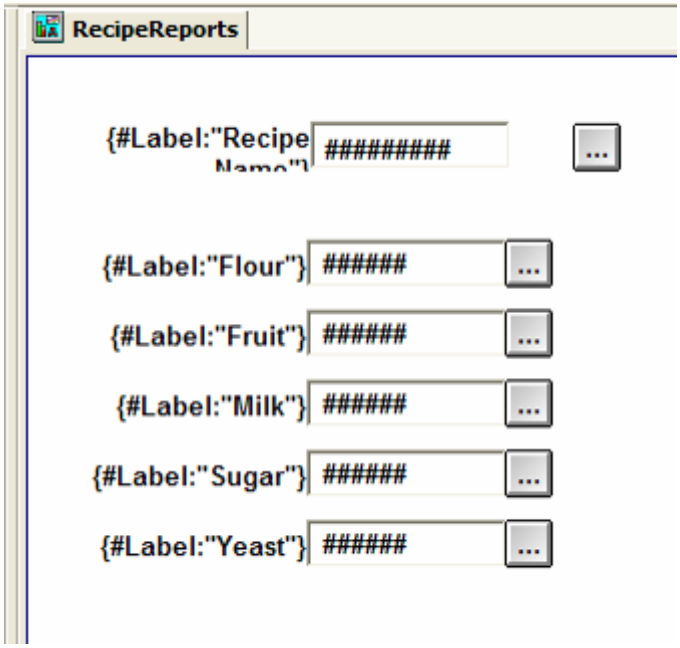
- ☐ Open the **Symbol Library** and from the **TextIO** folder, insert the Symbol **numeric_input02**.
- ☐ Double-click on the Linked Symbol in the Application Screen. Click on **Expand.**
- ☐ Enter the information in the **Symbol Properties** dialog box as shown and click **Ok**



- ☐ Add additional **numeric_input02** symbols to the Screen, configured as shown below:

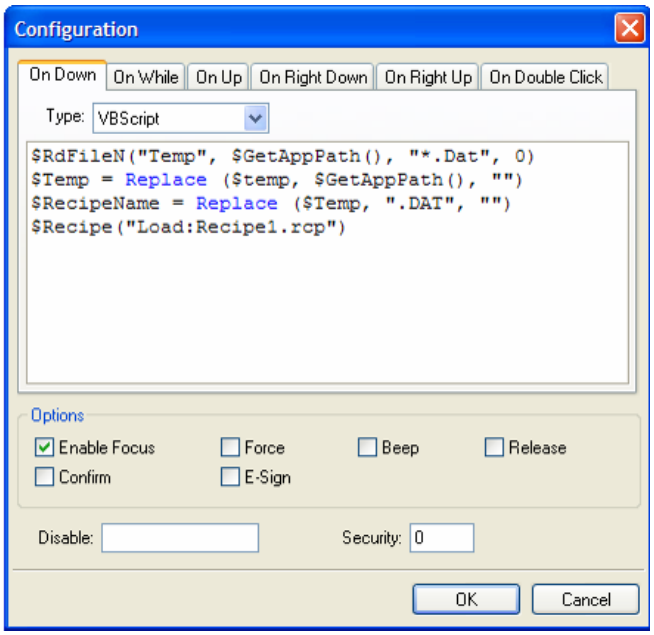
Symbol: Label **"Fruit",** Prompt "**Enter Amount of Fruit**", TagName **cake.fruit**
Symbol: Label **"Milk",** Prompt "**Enter Amount of Milk**", TagName **cake.milk**
Symbol: Label **"Sugar",** Prompt "**Enter Amount of Sugar**", TagName **cake.sugar**
Symbol: Label **"Yeast",** Prompt **"Enter Amount of Yeast"**, TagName **cake.yeast**

Note:
- You may want to edit the **text_input02** Symbol to add more placeholder fields

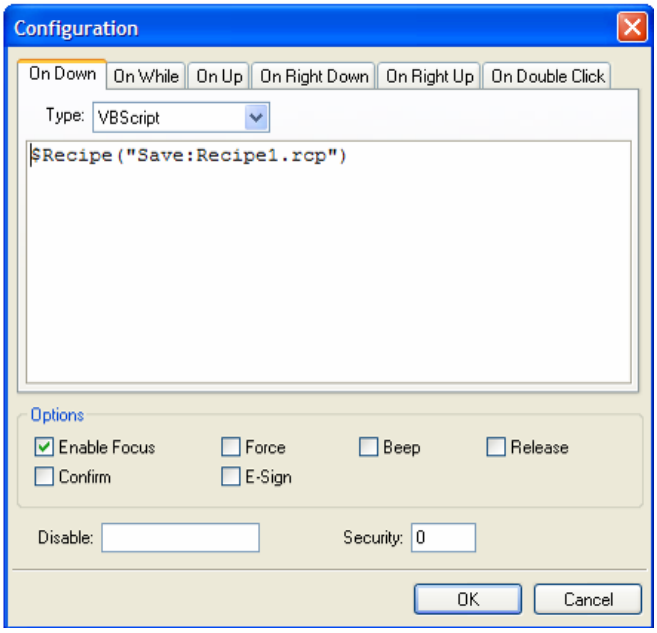


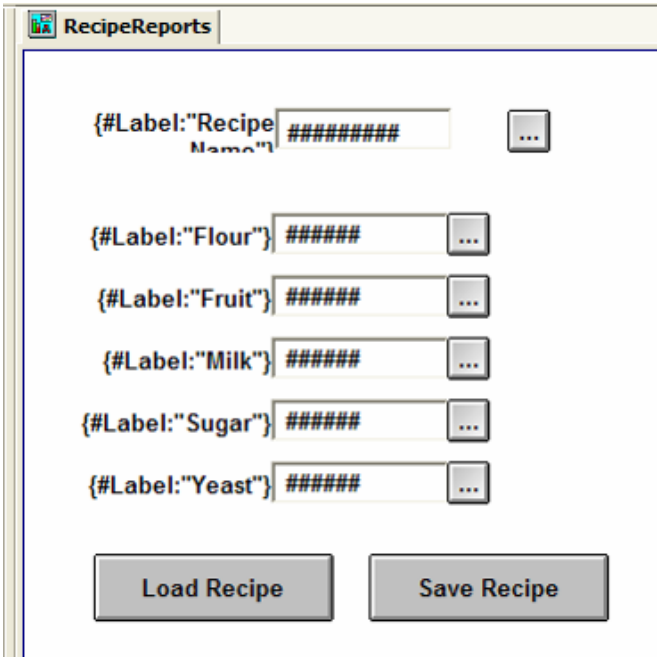
- ☐ Add a Tag named **Temp**, Type **String**, Size **0**
- ☐ Add a button with the Caption **Load Recipe**. Add a Command Dymanic to this button
- ☐ Add a button with the Caption **Save Recipe**. Add a Command Dynamic to this button
- ☐ Configure the Command Dynamic for each of these buttons as shown below:

Load Button Command Dynamic



Save Button Command Dynamic





- ☐ The finished screen should look as shown above
- ☐ To test the Recipe function, do the following:
  - Enter values for the various cake members (ingredients)
  - Enter a Recipe Name
  - Press the **Save Recipe** button
  - Repeat the above for different recipes

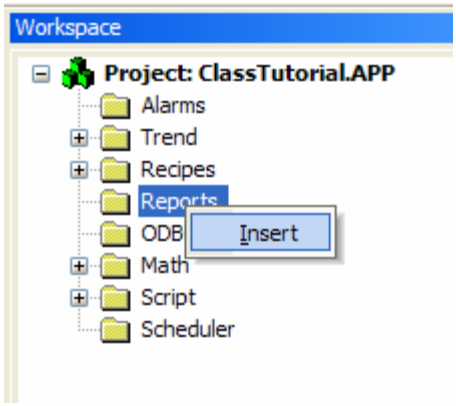
  - To load a recipe, press the load button.

Note:
- The VBScript code in the **Load Recipe** button Command Dynamic executes the IWS built-in function **RdFileN** that looks for files of a specified file extension in a specified directory. Since the Recipe files were saved in Text format, we will look for the **.DAT** file extension. If files were saved in XML format, we would need to filter for the **.XML** file extension.
- If you want to be able to search multiple directories, change the last parameter in the **RdFileN** function from a **0** to a **1.**

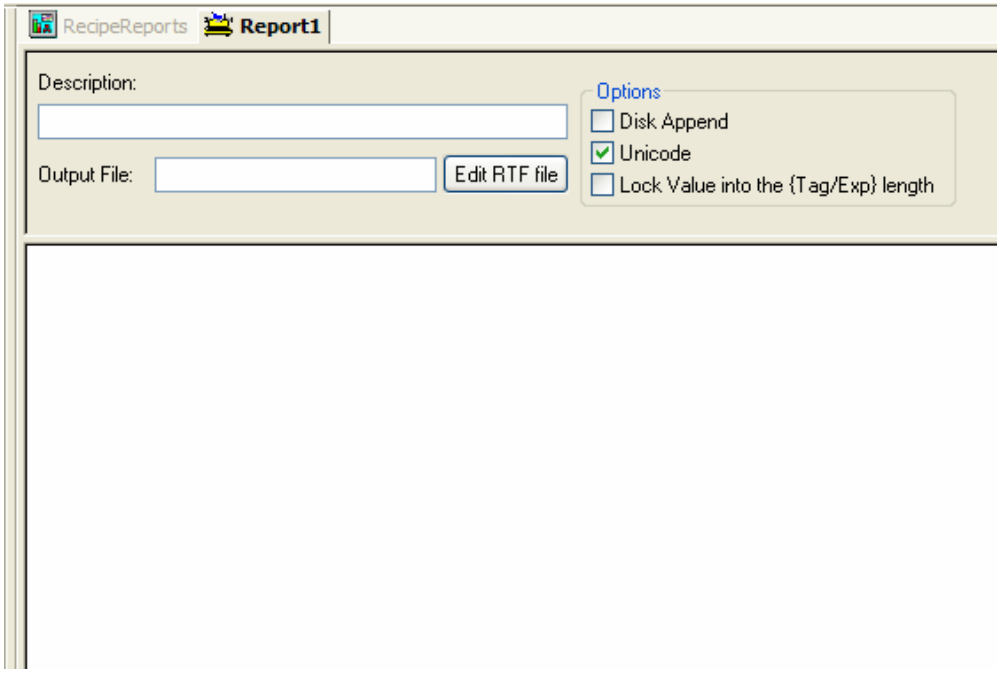
Chapter 15. Creating Reports

Creating reports with the Report tool is very easy — no other programming tools are required. You simply prepare a report mask in ASCII format or use the *Report Writer* tool (which creates RTF files). You can also enter values for tags by typing them between curly brackets (**{tag}**). Report creation in IWS is easy and efficient. To generate a report (i.e. print or send to a disk file), you call the **Report** function, an IWS built-in function from IWS Script or VBScript.

To create a report, you must first create the Report template. Right mouse click on the **Reports** folder and insert a new Report.



A report generator will open. You have your choice of creating Reports in Text format or RTF (Rich Text Format). The advantage of RTF files is that you can change fonts (size, color, type, bold, underline, etc.) and add embedded graphics (e.g. JPEGs).



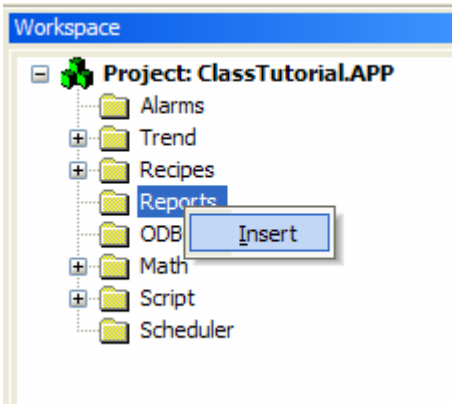
Exercise: Creating a Report

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 following procedure to create a report in ASCII format:

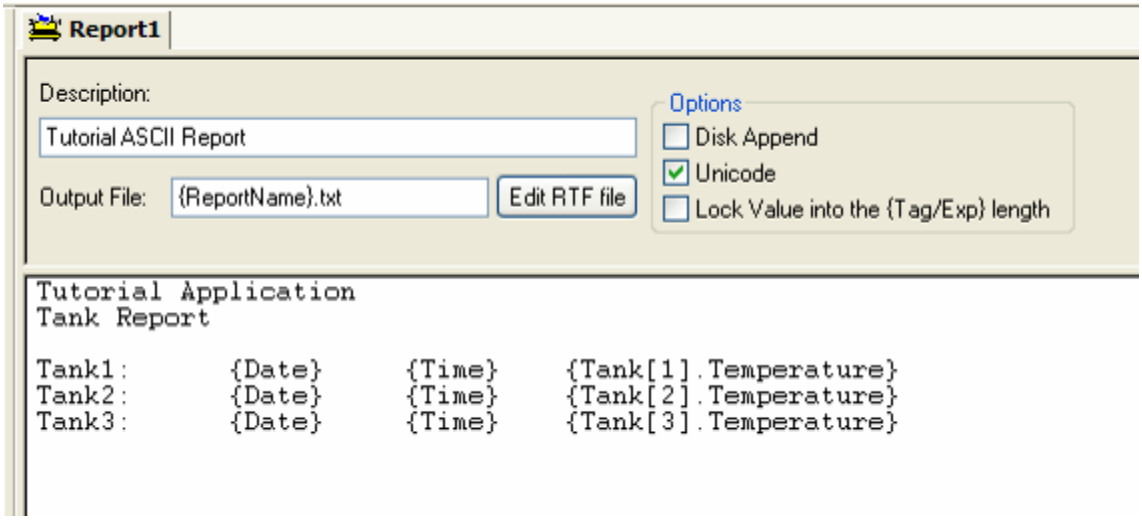
☐ In the **Workspace**, right-click on the **Reports** folder to create a **Report** worksheet. You will use this worksheet to create the body of your report file.



Creating a Reports Worksheet

Configure the **Report** worksheet as follows:

☐ As in the Recipe Exercise, the report (including the report name) will be generated to an output file. To name your report, type **{ReportName}.txt** as shown in the following figure. When you save the Report Worksheet, IWS will ask you to declare **ReportName** as a Tag. Declare it as a **String** Tag.



Report1.rep Worksheet

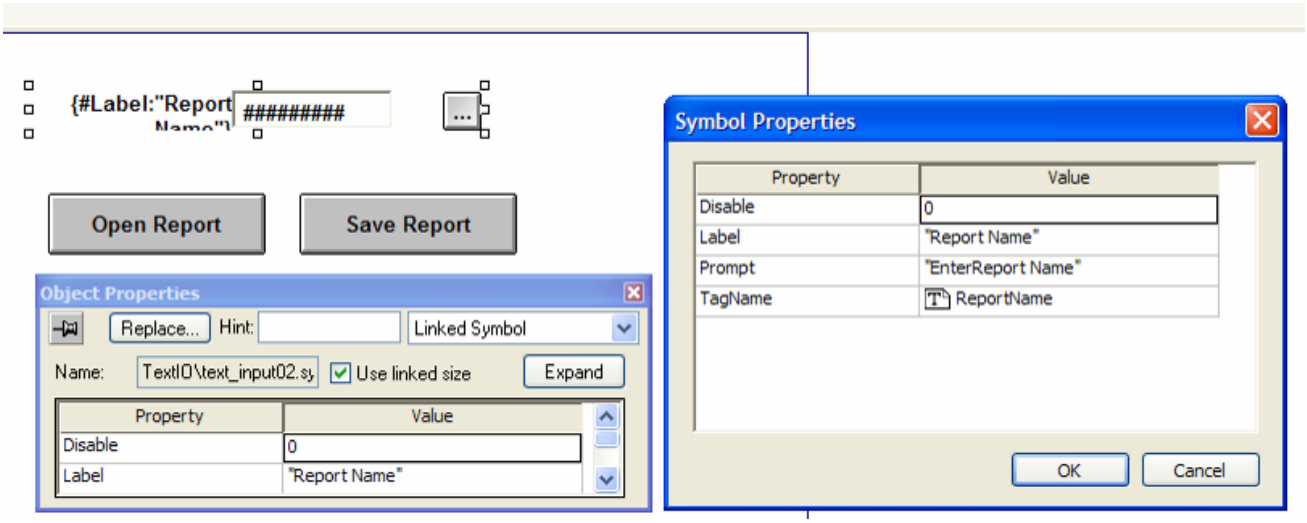
☐ Save the **Report** Worksheet as **Report1.rep** (the default name).

☐ Open the **RecipeReports** Screen and add the following objects:

- **Text_input02** from the **Application Symbols** folder
- A Button Object with the Caption **Open Report**
- A Button Object with the Caption **Save Report**

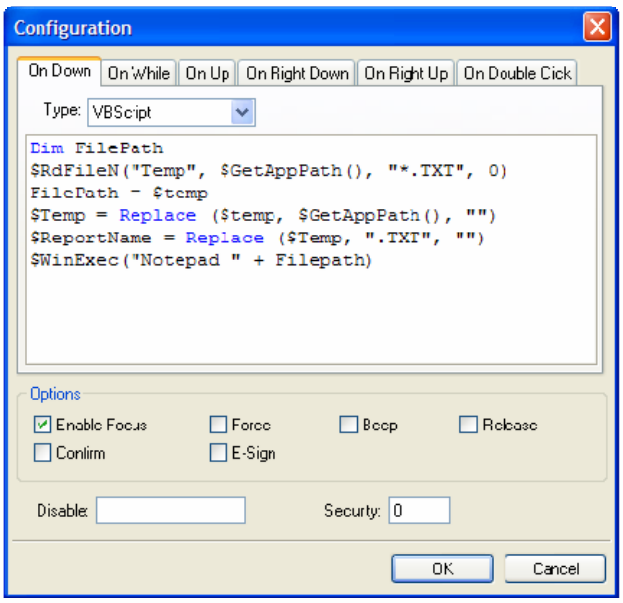
Creating the RecipeReports Screen Objects

☐ Double-click on the **text_input02** Object and when the *Object Properties* dialog displays, click on the **Expand** button and configure the **Symbol Properties** as follows:

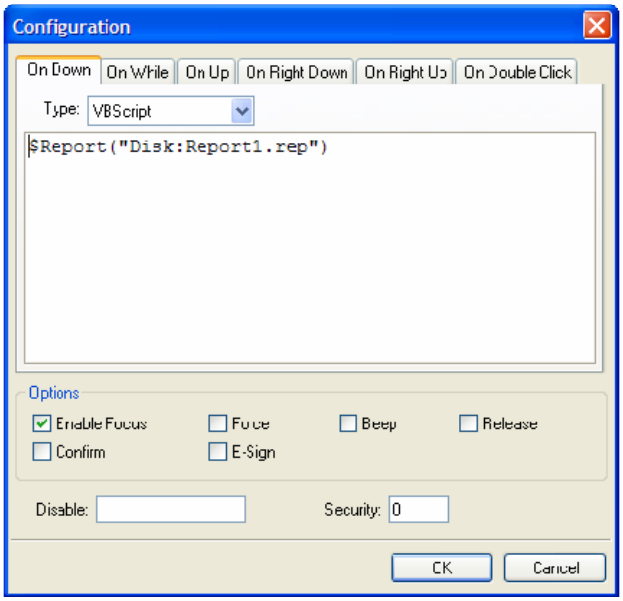


Configuring the Symbol Properties

- Add a Command Dynamic Property to the **Open Report** Button. Double-click the Button and configure the **Command** property as shown below. Be sure to leave a space after **Notepad**.
- Add a Command Dynamic Property to the **Save Report** Button. Double-click the Button and configure the **Command** property as follows:



Configuring the Open Report Button



Configuring the Save Report Button

Note:

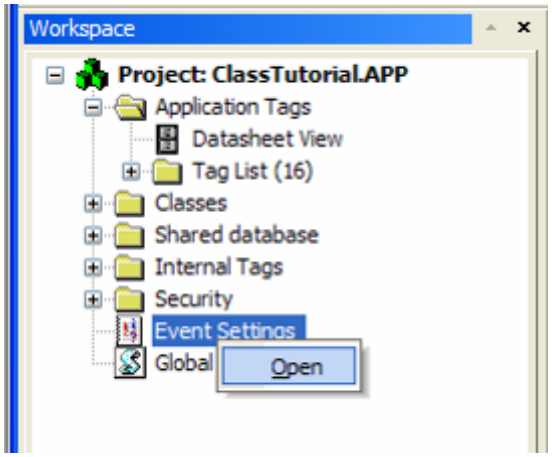
- The VBScript code in the **Open Report** button Command Dynamic executes the IWS built-in function **RdFileN** that looks for files of a specified file extension in a specified directory. Since the Report files were saved in Text format, we will look for the **.TXT** file extension.
- If you want to be able to search multiple directories, change the last parameter in the **RdFileN** function from a **0** to a **1.**
- The IWS built-in function **WinExec** executes a Windows command line. In this case, we are activating the Windows program **Notepad** and passing it the file name of the Report.
- RTF files cannot be opened with Windows Notepad. Windows Wordpad, Microsoft Word, or some other editor must be used instead.

Chapter 16. Events

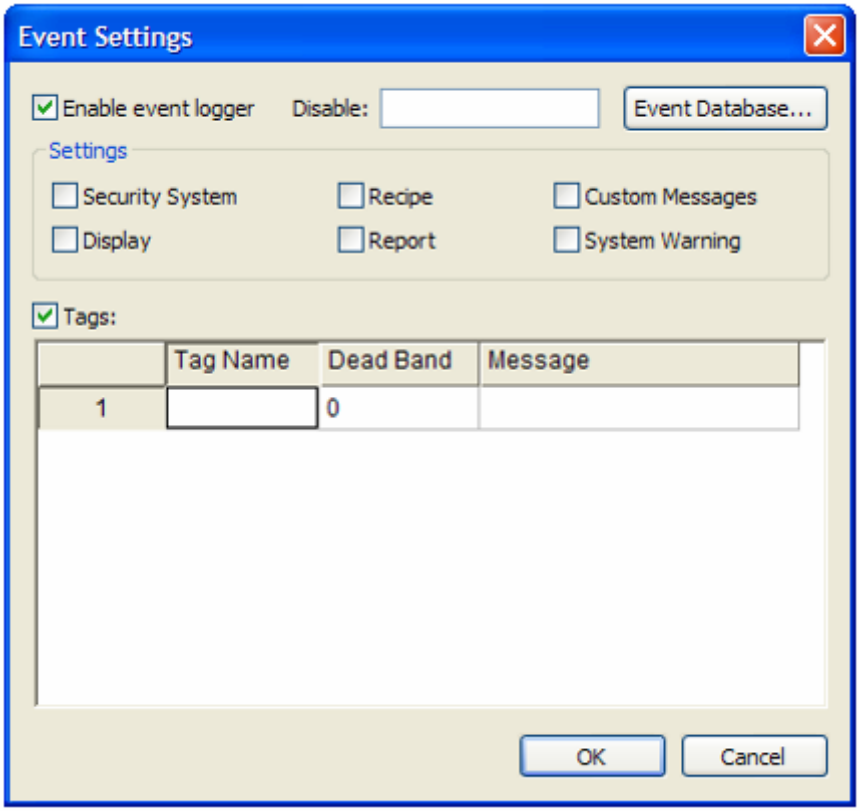
Use the steps to configure the Event Manager feature.

Exercise: Configuring Event Retrieval

☐ Select the **Database** tab. Right-click the **Event Settings** icon, and select **Open** from thepop-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** field (optional): Type a tag into this field. If the tag value is other than 0 (0=*false*), IWS automatically disables the Event Logger.
- **Event Database…** : Used to configure the Event Database
- **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).

Chapter 17. Using the Translation Tool

You can use the Runtime **Translation Tool** to change translate text on buttons or in any other text field from one language to another. All you have to do is create a **Translation** Worksheet and apply the IWS built-in translation function.

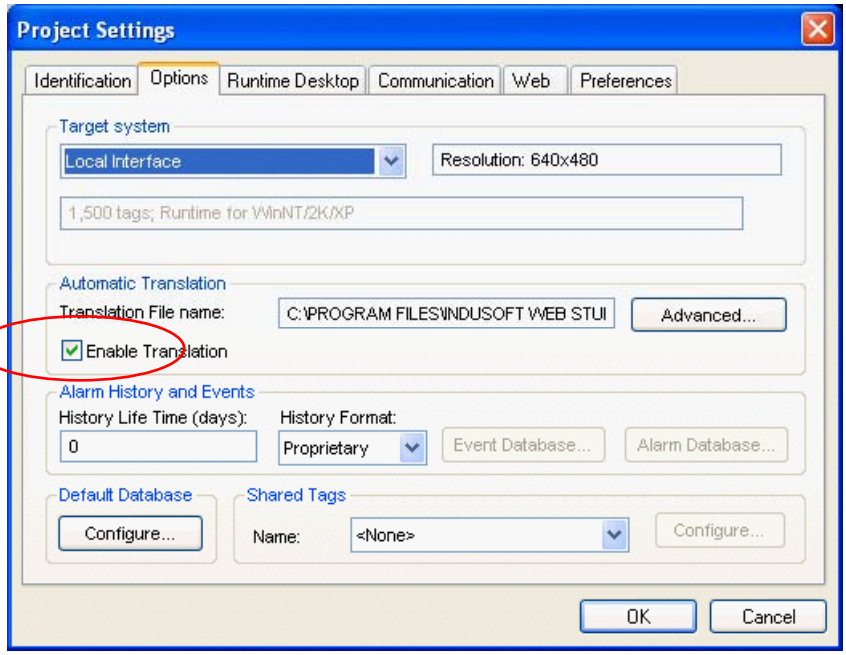
The translation function searches your application for all the text typed in the **Original** column (or the left most column) of the **Translation** Worksheet and then changes that text into the text typed in another column of the **Translation** Worksheet. The **Translation** Worksheet can support two or more languages. The **Translation Tool** ignores any text that has not been entered into the **Translation** worksheet.

In the following exercise, you will set up a **Translation** Worksheet and use the translation function to translate some text.

Exercise: Enabling External Translation

To successful translate text within an application, use the following procedure:

- ☐ Select **Project → Settings** from the main menu bar to open the *Project Settings* dialog.
- ☐ Select the **Options** tab, and click (*enable*) the **Enable Translation** check-box:



Project Settings Dialog

When this option is unchecked (clear), the Translation Tool is disabled and the text are displayed only as configured in the original application files.

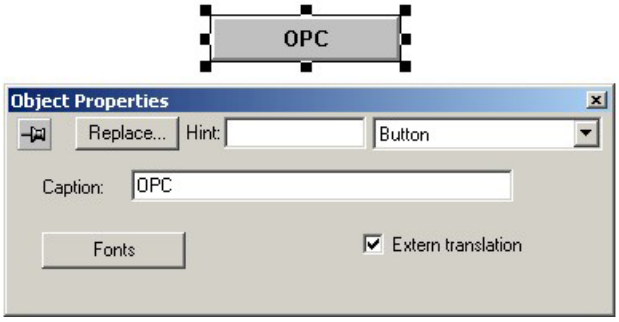
- ☐ Select a file in the **Translation File name:** field. The translation file configured in this field will be loaded by default when launching the application. This option is useful to configure the default translation file name when launching the application. The user can set a different language during the runtime by executing the SetTranslationFile() function.
- ☐ Click **OK** to close this dialog.

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 toobars.
- ☐ 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:

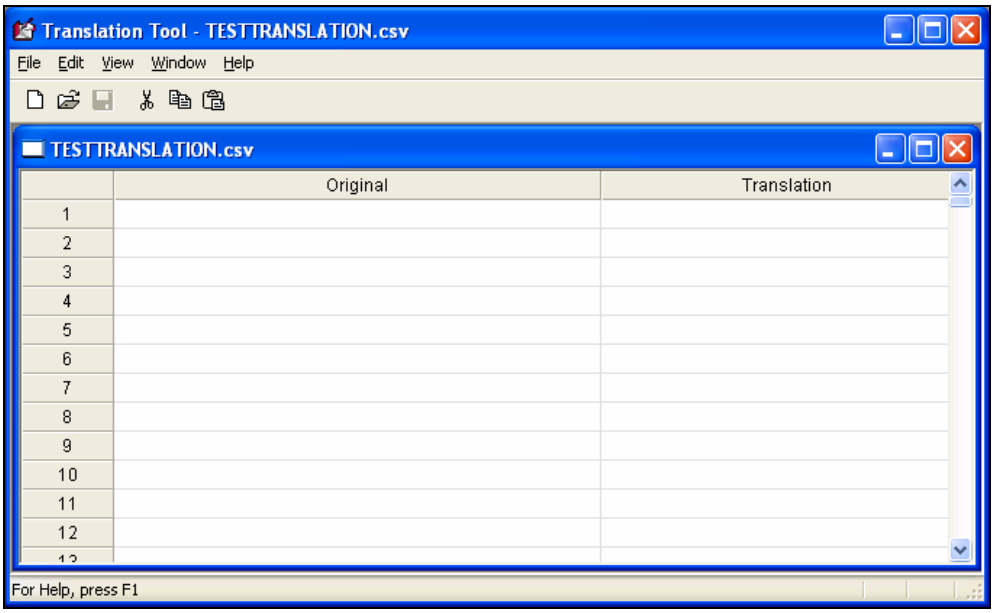


Translation Enabled for the OPC Button

Translation Editor

The Translation Editor can be launched by the **Tools > Translation Editor** menu option:

Select **Tools → Translation Editor** from the main menu bar to open a blank sheet.



Blank Worksheet

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.

Insert Column (shortcut F11): Delete the column currently selected.

After editing the translation file, 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).

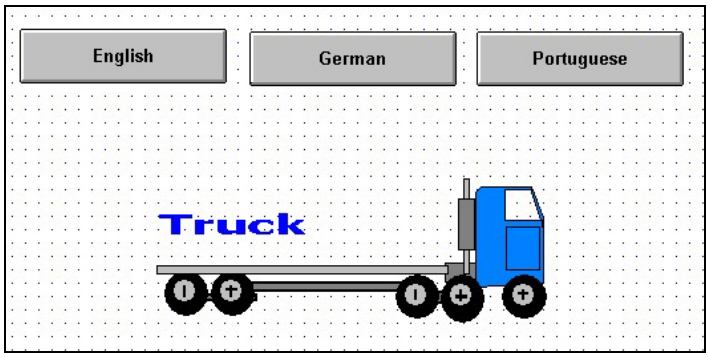
> **Note:**
> For legacy reasons, the *Translation Tool* supports translation files with the extension TRA. The translation files saved with this extension use the pipe character ("|") instead of the comma character (",") to separate the text between columns.

You can now proceed to the next section to create a *Translation Screen*.

Creating the Translation Screen

To create a *Translation Screen*, use the following steps:

Using the **Symbol Library** and your toolbars, create a new screen like the one in the following figure (save as **Translation.scr**):

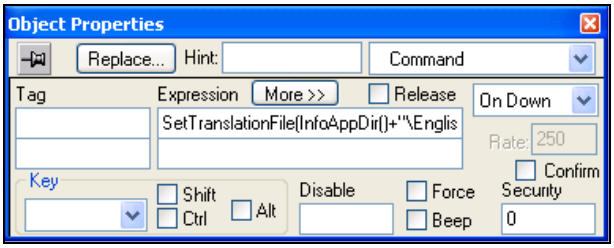


New Translation Screen

Double-click on the **English** button to open the *Object Properties* dialog.

Select **Command** from the drop-down menu and complete the fields as follows:

Expression field: **SetTranslationFile("*strFileName*")**, substituting the name of the translation file between the double quotes.



Configuring the English Button

> **Note:**
> The **SetTranslationFile** function defines the *Translation* worksheet to be used in the application.

Repeat the preceding steps to configure the **Portuguese** and **German** buttons.

Save the screen and execute the application to test the configurations.

Chapter 18. Configuring a Security System

In this part of the class you will define group and user security accounts (access privileges) for your application.

Enabling Security

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: When checked, enables the Security System.

Main Password button: Opens the *Security System Main Password* dialog so you can define passwords granting access to the security system.

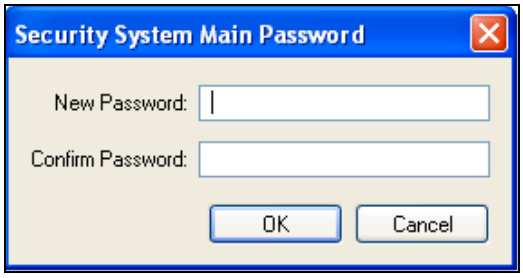
Use preferentially the Remote Security System check-box: When checked, enables security from a remote system.

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.

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.

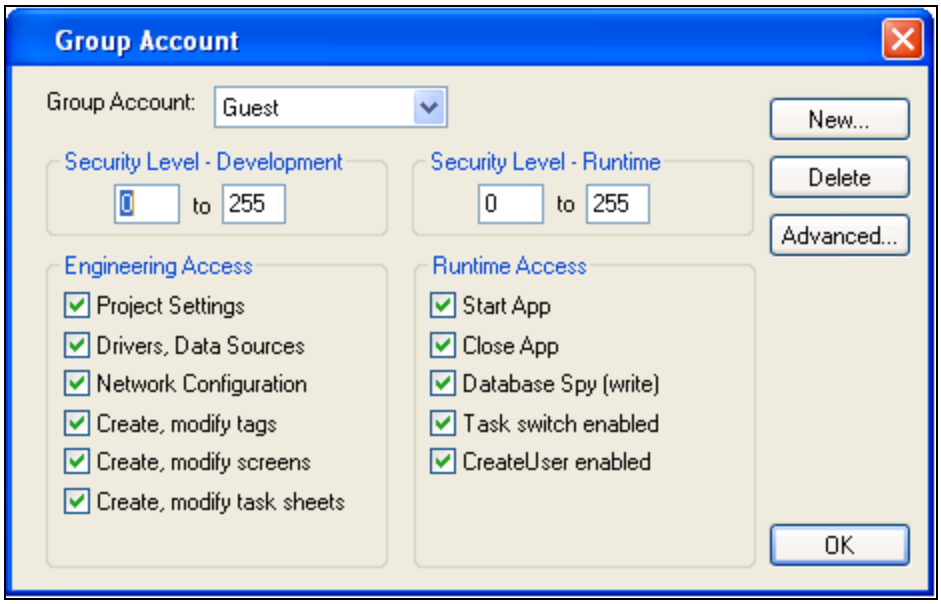
If the passwords are different, you will be prompted to type it again.

> **IMPORTANT!**
> After defining your password, you must use that password each time you access the Security System, so it is mandatory that you remember it.

Defining Group Accounts

The *Group Account* dialog enables you to create and maintain user groups, enable/disable operations, and set security level ranges for development and run-time 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 security level is not in the group security scale 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.

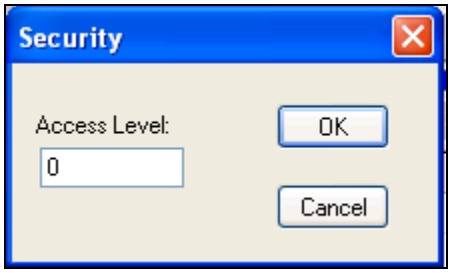
Delete button: Deletes the currently selected user group.

Advanced button: Calls the advanced settings, where the user can set the password minimum size, aging, e-sign timeout, lock up after a certain number of unsuccessful attempts, etc.

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,

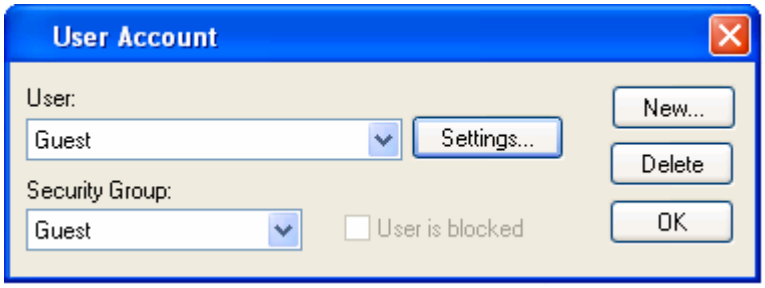
Only UserA can access *Math Worksheet 001*

Both users can access *Math Worksheet 002*
Only UserB can access *Math Worksheet 003*
Neither user can access *Math Worksheet 004*

Defining User Accounts

Click on the **User Account** button to open the *User Account* dialog. You can use this dialog to create and maintain user accounts for your application. (You defined your application users for each group using the *Group Account* dialog.)

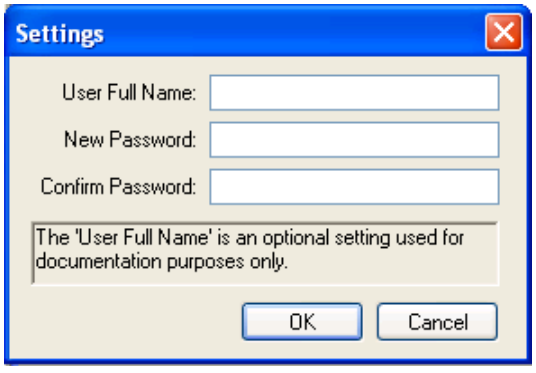
> **Note:**
> Alternatively, you can access the *User Account* dialog from the *Users* folder located in the *Security* folder on the **Database** tab, or by selecting **Insert →User** from the main menu bar.



User Account Dialog

Use the features on this dialog as follows:

User combo-box: Select from a list of application users.
Security Group combo-box: Select from a list of application groups.
New button: Open the *New User Account* dialog to create a new user.
Delete button: Delete the selected user.
Settings button: Open a *Settings* dialog to define user passwords.



Settings Dialog

SPECIFYING GUEST USERS

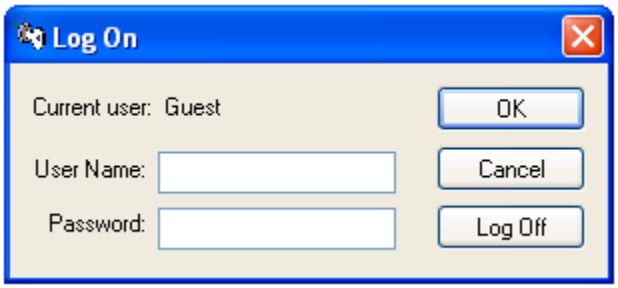
After initializing, a user is logged on as a *Guest* user (by *default*). If no other user logs on or the current user logs off, the *Guest* user is automatically logged on.

The Guest group has default privileges. Because the installation parameters of a Guest group leave all tasks enabled by default, you should change this parameter and set as few privileges as required for a start-up procedure.

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 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 **Log Off** is executed, the *Guest* user is automatically logged on.

Exercise: Creating Security Groups

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.

In this exercise, you will create three groups:

Operation

Maintenance

Development

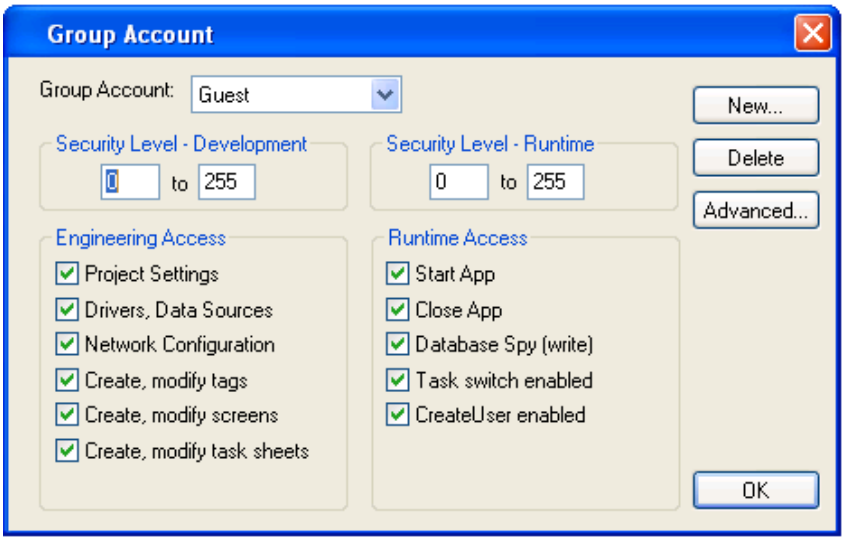
Use the following procedure to create these groups:

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 Security Group

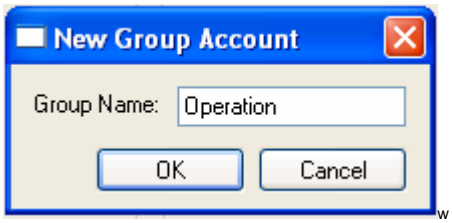
The *Group Account* dialog displays:



Group Account Dialog

Remember, you cannot delete the default group called *Guest*, so you must create a new group, as follows.

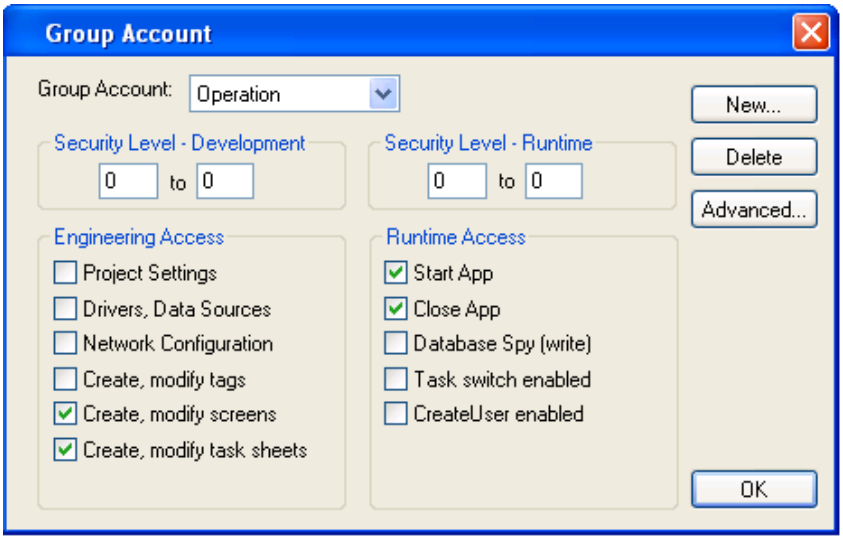
Click the **New** button, and when the *New Group Account* dialog displays, type a group name into the field provided. (For this class, type **Operation**.) Click **OK** to close the dialog.



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.

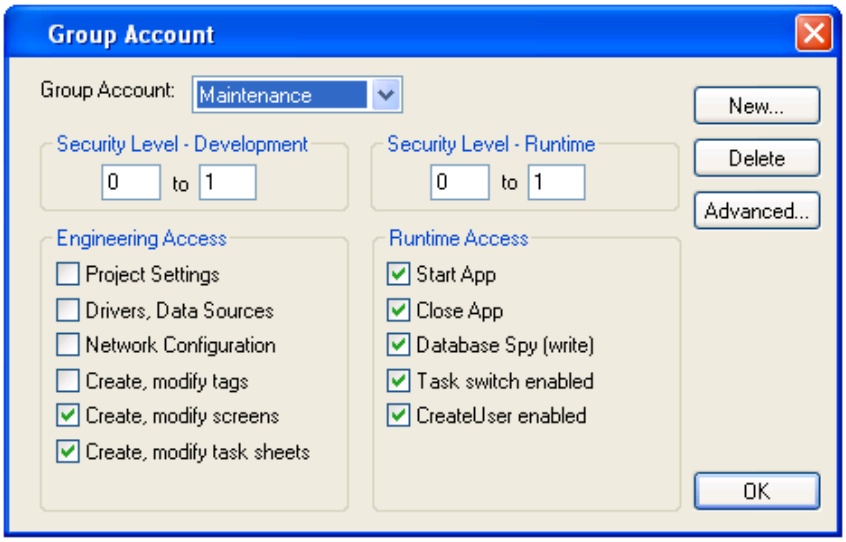
Configure the access rights for this group as shown:



Operation Access Rights

Click the **New** button again and create the Maintenance group. Click **OK** to close the *New Group Account* dialog.

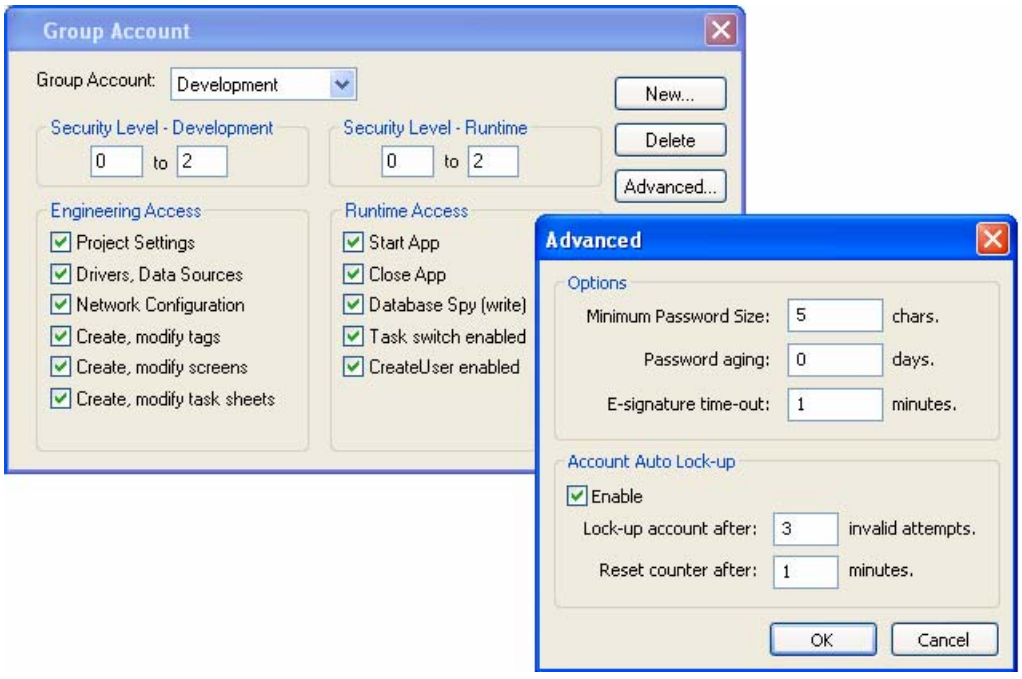
Select **Maintenance** from the **Group Account** combo-box and configure the access rights for the group as follows:



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. As this is a very important group, we will also configure the advanced settings.



Development Access Rights

Click **OK** to save this configuration.

Notes:

Each group has a Security Level range 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 run-time environment.

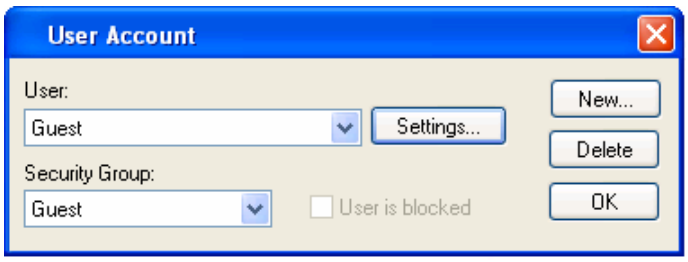
Next, you must create new users and associate these users to the group accounts you just created. Use the following steps:

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

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 class, 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.

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 **Password** 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:

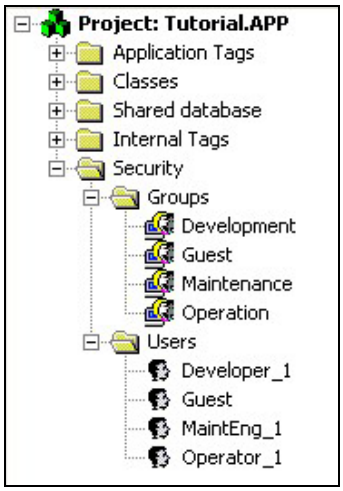
Click **New** and create the **Developer_1** user.

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:
In the last exercise, you created only one user for each group. However, it is possible to create many users for each group account. You also can create new users for an application using the **CreateUser** function, even during the runtime.

Exercise: Creating Alarm Groups

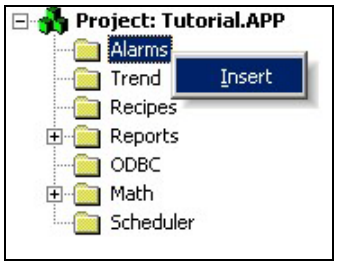
In this exercise, you will create alarm groups using the following procedure:

Create the following tags:

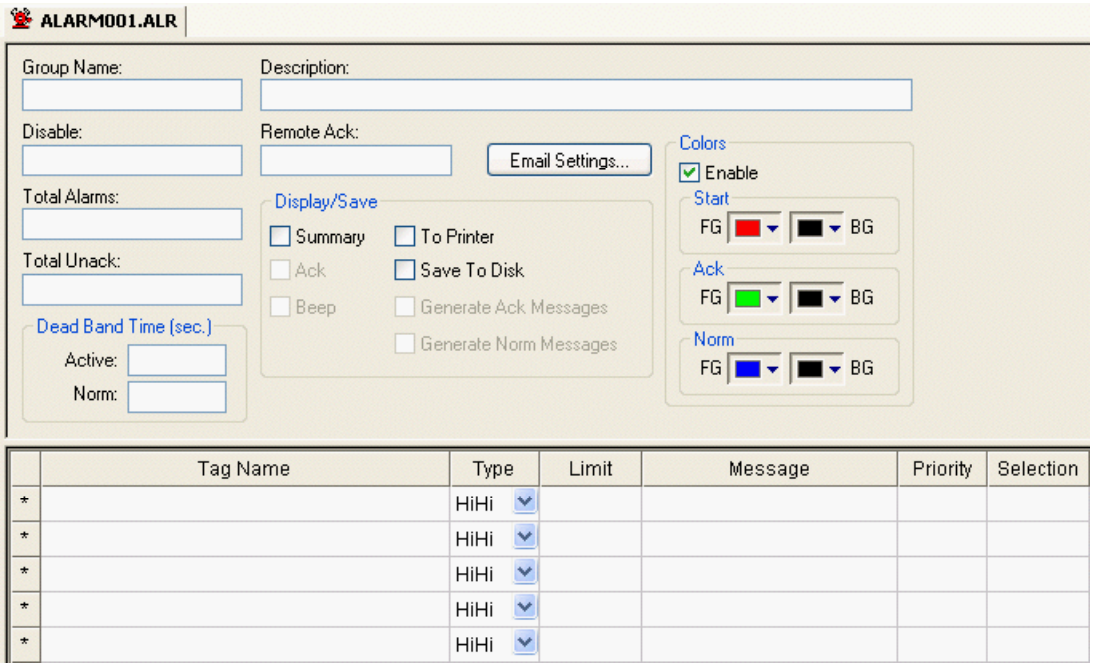
Tag Name	Size	Type
Alarm_Sel	2	String
View	2	Integer
Filters	2	Boolean

Select the **Tasks** tab and then right-click on the *Alarms* folder.

Select **Insert** from the pop-up menu to open an *Alarms* worksheet.



Opening an Alarm Worksheet



ALARM001.ALR Worksheet

When you create an *Alarm* worksheet, you specify which tags will be alarms, and specify what kind of alarm, limit, message, priority, filter selection, and message color properties you want the alarm object to have.

When you save this worksheet, IWS records all alarm occurrences to an ASCII file and saves that file in the *Alarms* subfolder of your application folder, with the **.hst** file extension.

Alarm Worksheet Header

This section explains the parameters provided in the Header section of the *Alarm* worksheet. You use these parameters to define common characteristics for all alarms in the group.

Group Name field: Type a name to distinguish alarm groups.

> ➲ **IMPORTANT!**
> Before changing the **Group Name** field, save the *Alarm* worksheet. You can lose alarm settings in an unsaved worksheet.

Description field (*optional*): Enter remarks for documentation purposes.

Disable field: Disables all alarms in a group.

You must add a tag to this field.

If the tag value is greater than zero, IWS disables the group and does not generate alarm messages.

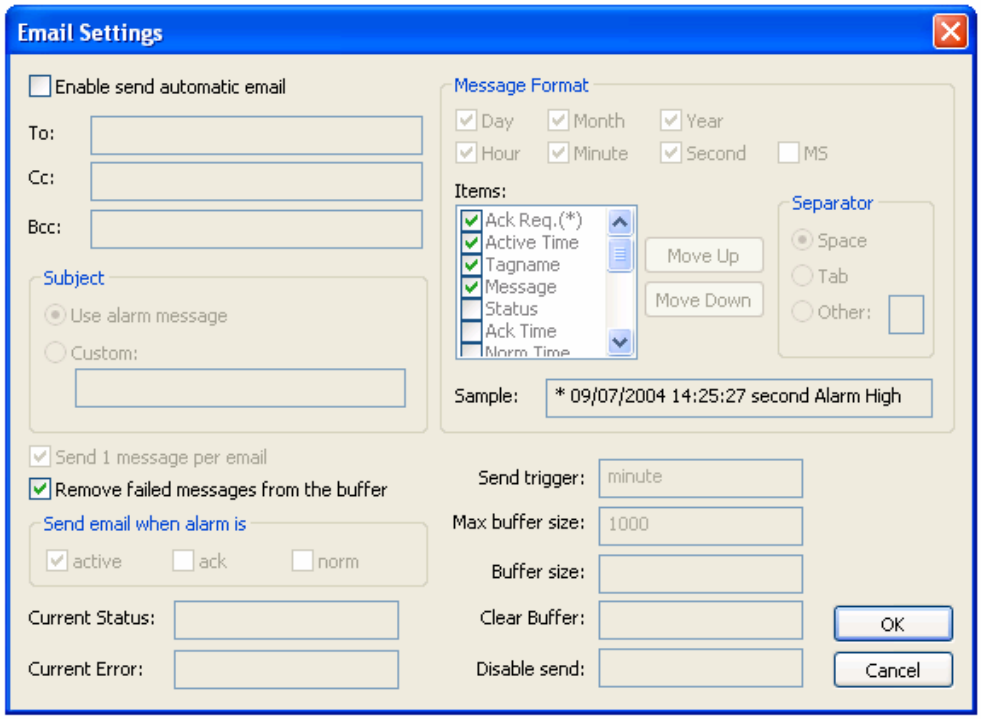
If you leave this field blank, IWS always enables the group.

Remote Ack field: Enter a tag to acknowledge alarms. Acknowledgment occurs when the tag value changes.

Email Settings button: Click to configure the application to send an email automatically to a designated person(s) when an alarm event occurs.

> ➲ **IMPORTANT!**
> To use this feature, you must be connected to the Internet (manually or using an automatic dial-up function) and you must have executed the **CNFEmail** function (described in the IWS *User Guide* "Appendix: *Studio Functions*") to configure the SMTP server, user name, password, and domain before trying to send an email.

The *Email Settings* dialog displays:



Email Settings Dialog

This dialog contains the following parameters:

- **Enable send automatic email** check-box: Click (*enable*) this box and type email addresses in the **To** (*required*), **Cc** (*optional*), and **Bcc** (*optional*) fields to automatically send an email message to a designated recipient(s) when an alarm occurs.
- **Use alarm message** button: Click (*enable*) this button to use the actual alarm message as the subject line of the email.
- **Custom** radio button and field: Click (*enable*) this button to enter and use your own subject line for the alarm email. Type the subject line text in the field provided.
- **Send 1 message by email** check-box (*available only when you select* **Custom**): Click (*enable*) this button to send each alarm message notification in a separate email. (For example, if there are three alarms, IWS will send three emails.)

  This parameter is disabled by default, which means IWS will send *all* alarm messages to the designated recipient in a single email.
- **Remove failed messages from the buffer** check-box: Click (*enable*) this button to remove alarm messages from the buffer when the alarm notification email fails (cannot be delivered).
- **Send email when alarm is** panel: Click (*enable*) one or more of the check-boxes in this area to send an email automatically when the alarm becomes active (**active**), when someone acknowledges the alarm (**ack**), and/or when the alarm is normalized (**norm**).
- **Current Status** field: Type a tag to receive the current status of the alarm.
- **Error** field: Type a tag to receive the error that caused the alarm.

> **Note:**
> See "**GetStatusSendEMailExt(optTagName)**" in IWS *User Guide* "Appendix: *Studio Functions*" for a description of the returned values for the **Current**

Status and **Error** fields.

- **Message Format** pane: Use the parameters in this area to format the outgoing email messages.
  * Click one or more of the check-boxes to include the **Day**, **Month**, **Year**, **Hours**, **Minutes**, **Seconds**, and/or **MS** (milliseconds) when the alarm event occurred.
  * Click one or more items in the **Items** list to include that alarm information in the email message.
  * Click the **Space**, **Tab**, or **Other** radio button to specify what kind of separator to use between elements of the alarm event message.
- **Send trigger** field: Type a tag in this field and when the tag's value changes, IWS checks all active alarm events. If there are any alarm events for which email notification has not been sent, IWS automatically sends an email notification message to the designated individual(s).
- **Max buffer size** field: Type a value to specify the maximum number of alarm messages to store in the buffer. If the number of messages exceeds this value, IWS uses the FIFO (first in-first out) algorithm to manage the buffer by deleting the oldest message whenever a new message occurs. The default buffer size is 16,000 messages. (***Note:*** This field also accepts a tag.)
- **Buffer size** field: Type a tag to display the number of alarm messages currently in the buffer. (IWS resets this number after sending the email notification.)
- **Clear Buffer** field: Type a tag to clear the buffer. When this tag changes value, IWS deletes all messages currently in the buffer.
- **Disable send** field: Type a tag to disable the send email feature. When this tag value is true (a value other than zero), IWS stops sending email and stops sending any existing or new alarm event messages to the buffer.

After configuring the **email** parameters, click **OK** to close the dialog and return to the *Alarms* worksheet.

- **Total Alarms** field: Enter an integer tag to receive a value denoting the total number of active alarms (acknowledged or unacknowledged) and unacknowledged normalized alarms.

  When an alarm *returns to the normalized state* and *has been acknowledged*, the IWS *Alarms* module no longer includes that tag in the total count.

- **Total UnAck** field: Enter an integer tag to receive a value denoting the total number of unacknowledged alarms, *regardless of the alarm state* (active or normalized).

  When an alarm *has been acknowledged*, the IWS *Alarms* module no longer includes that alarm in the total count.

The following table is provided to further illustrate how the IWS *Alarms* module counts alarm event messages for the **Total Alarms** and **Total UnAck** field totals:

Alarm State	Acknowledgement State	Counted for Total Alarms Tag?	Counted for Total UnAck Tag?
Active	Unacknowledged	Yes	Yes
	Acknowledged	Yes	No
Normalized	Unacknowledged	Yes	Yes
	Acknowledged	No	No

Note:
We recommend using unique tag names for the **Total Alarms** and **Total UnAck** fields for each *Alarms* worksheet.

- **Display/Save** area: Specify the following parameters.
  - **Summary** check-box: When selected, sends alarm messages to an alarm object on the screen.

    Caution:
    If you did not select the **Summary** option, the alarms for this group will not appear in the alarm objects in the screens and printer during execution.

  - **Ack** check-box: Demands the acknowledgment of the alarm messages. Only available if the **Summary** field is enabled.
  - **Beep** check-box: Sounds the beep until the alarm is acknowledged. Only available if the **Ack** and **Summary** fields are enabled.
  - **To Printer** check-box: Sends the each alarm messages of this group to the printer. You can use this option with a dot matrix printer only (or any printer that prints line by line).

    Caution:
    The **Printer** check-box should not be used with DeskJet or LaserJet printers because they will spend one entire leaf of paper for each alarm message. These printers are not able to print one line and then wait for the next printing command.

  - **Save to Disk** check-box**:** Sends the alarm messages of this group to a file on the hard disk. You must select this option if you want to have history alarm objects.
  - **Generate Ack Messages** check-box**:** Generates messages whenever the alarms of this group are acknowledged. Only available if the **Disk** or **Printer** fields are enabled.
  - **Generate Norm Message** check-box**:** Generates messages whenever the alarms of this group return to their normal state. Only available if the **Disk** or **Printer** fields are enabled.

– **Colors** area: Specify the following parameters to define colors of the alarm summaries for the alarm object. IWS shows each alarm object in the alarm message using the colors specified for its group.

Enable color check-box: Click (*check*) to specify colors.

* **Start** rectangle: Click **FG** to select a color for the text of the alarm messages and **BG** to select a color for the background of the alarm text.
* **Ack** color rectangle: Click **FG** to select a color for the text of the acknowledge messages and **BG** to select a color for the background of the acknowledge text.
* **Norm** color rectangle: Click **FG** to select a color for the text of the normal messages and **BG** to select a color for the background of the normal text.

When the *Color* dialog displays, click on a color to select it, and close the dialog.

Use the **Body** parameters on this worksheet as follows:

- **Tag Name** field: Type a tag to be monitored by the alarm group.
- **Type** drop-down list: Click to select one of the following alarm types. (You can change any of these fields in the run-time module. For additional information, see IWS *User Guide*, "Chapter 5: *Working with Tags*.")
  – **HiHi**: Alarm limit is too high; generate an alarm message when the tag value is equal to or greater than the **HiHi Limit** value.
  – **Hi**: Alarm limit is high; generate an alarm when the tag value is equal to or greater than the **Hi Limit** value.
  – **Lo**: Alarm limit is low; generate an alarm when the tag value is lower than or equal to the **Lo Limit** value.
  – **LoLo**: Alarm limit is too low; generate an alarm when the tag value is lower than or equal to the **LoLo Limit** value.
  – **Rate**: Determines the speed of the variation rate for a tag. If the variation speed is higher than the established one in this field, generate an alarm. The speed can be determined per second, minute, or hour.
  – **Deviation+**: Deviation for a higher value; generate an alarm when an augmentation in the tag value is equal to or higher than the established limit.
  – **Deviation–**: Deviation for a lower value; generate an alarm when a diminution in the tag value is equal to or higher than the established limit.
- **Limit** field: Type a value to limit the alarm generation.
- **Message** field: Type an alarm message to display.

> ➲ **Caution:**
> Alarm messages can contain any system tag using the syntax:
> **message {tag_name}**

- **Priority** field: Type an integer number (0 to 255) to indicate the priority within a group. Tags with a higher priority must have a higher priority value.

- **Selection** field: Type a string to filter in the alarm summary objects.

> **Caution:**
> The **Selection** field must have a string with a maximum of 7 characters (the other characters will not be considered).

- **Alarm summary**: When you enable the alarm history file for a group, IWS saves the file as **AL*yymmdd*.ALH** in the application's **\app\ALARM** directory.

  Where ***yymmdd*** refers to the year, month, and day the file was created.

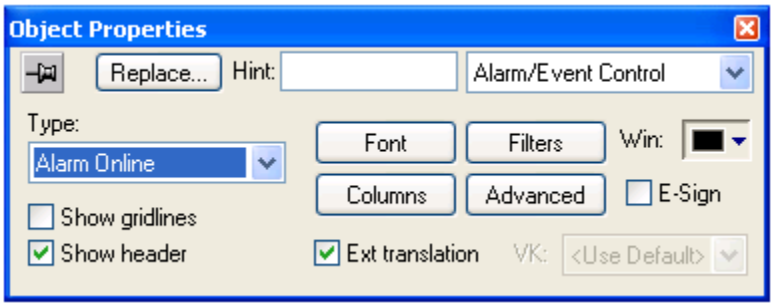
Exercise: Creating On-Line Alarm Screens

To begin this exercise:

Open the *Standard.scr* screen and save it as **AlarmOnLine.scr**.

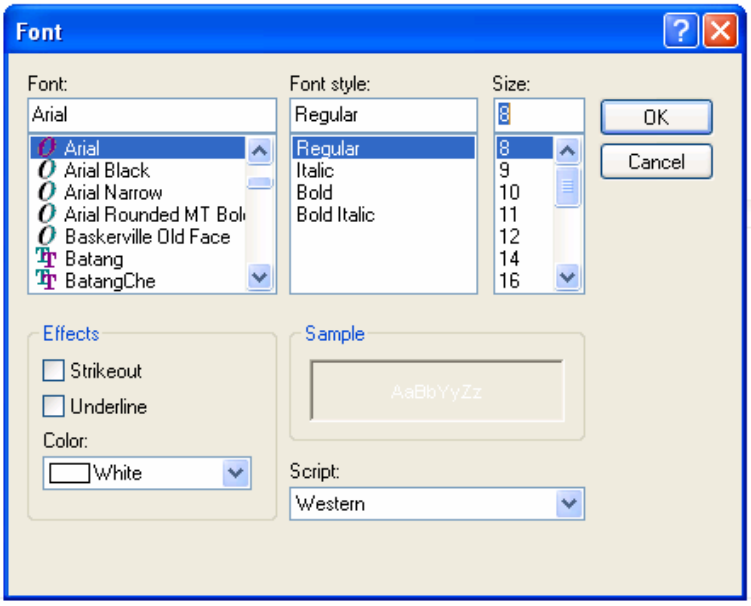
Click the **Alarm/Event Object** button ( ) and create an *Alarm* object on the screen.

Double-click the *Alarm* object to open the *Object Properties* dialog:



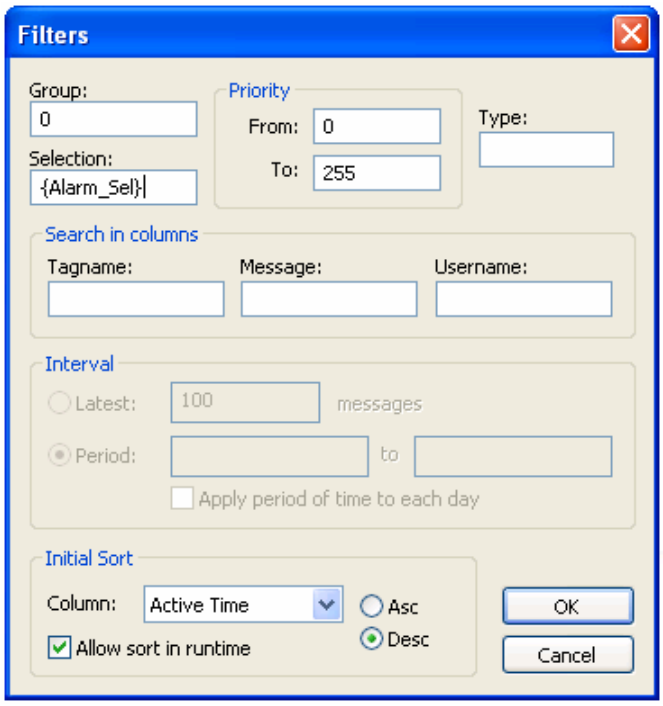
Object Properties: Alarm

Configure the *Alarm* object. (Being sure to select the **Alarm Online** type and set the font parameters to Arial, 8 point, and White.)



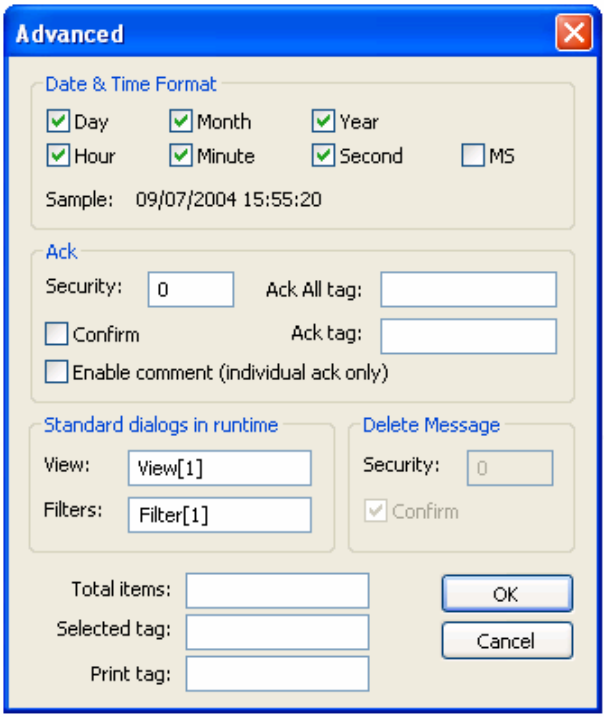
Configuring the Alarm Object

Click the **Filters** button on the *Object Properties* dialog to configure the alarm filter as follows:



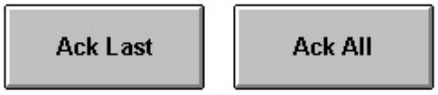
Configuring the Alarm Filter

Click the **Advanced** button on the *Object Properties* dialog to configure the alarm Advanced Settings as follows:



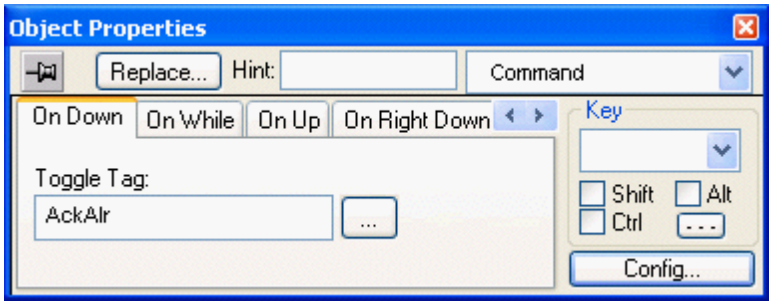
Configuring the Alarm Advanced Settings

Create two buttons on your screen to acknowledge these alarms:



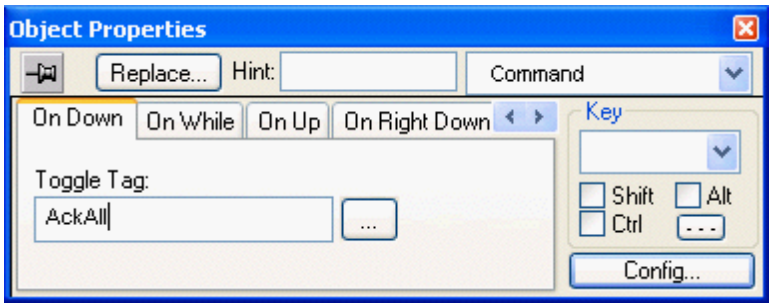
New Alarm Buttons

– The first button will toggle the value of the internal tag AckAlr, to acknowledge the last alarm that occurred. Give the button the caption **Ack Last**, add the **Command** property to the button, and configure the command as shown:



Configuring the Ack Last Button

– The second button will toggle the value of the internal tag AckAll, to acknowledge all alarms that occurred. Give the button the caption **Ack All**, add the **Command** property to the button, and configure the command as shown:



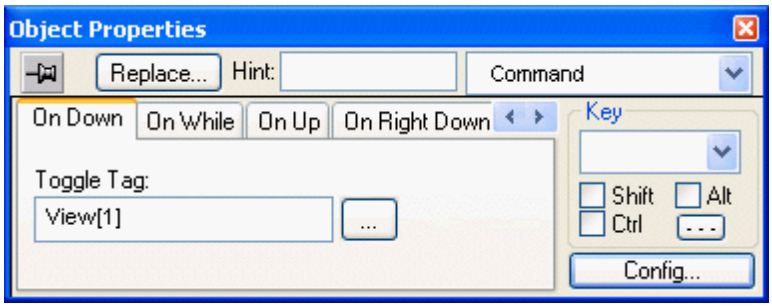
Configuring the Ack All Button

Draw the text and objects as shown below and configure them with the **Alarm_Sel** tag to sort the alarm messages shown in the alarm object.

Create another two buttons on your screen, one to filter the alarms and another for selecting columns during the runtime:

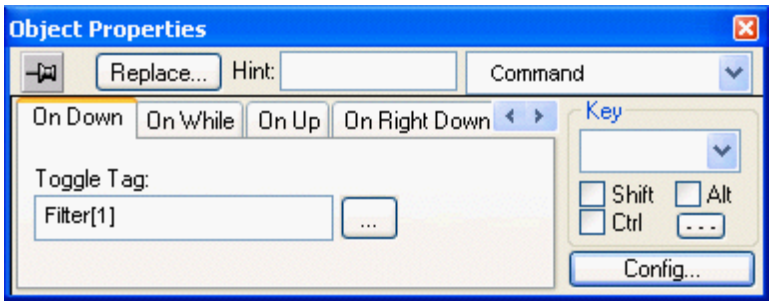
Alarm Buttons

– The first button will toggle the value of the tag View[1], to allow the user to select which columns will be shown during runtime. Give the button the caption **Column**, add the **Command** property to the button, and configure the command as shown:



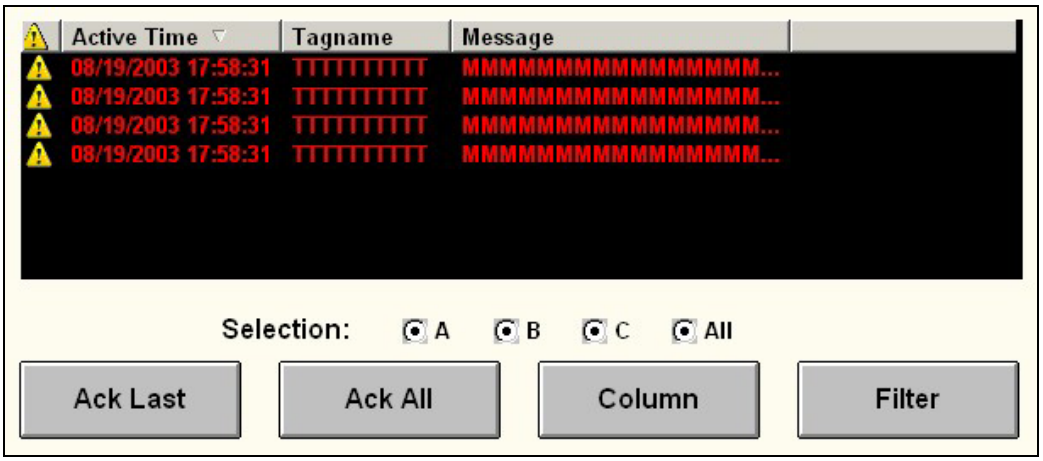
Configuring the Column Button

– The second button will toggle the value of the tag Filter[1], to open the window that allows the user to dynamically filter the current alarms by group, time, type, etc. Give the button the caption **Filter**, add the **Command** property to the button, and configure the command as shown:



Configuring the Column Button

Your screen should look like the following:



Finished Online Alarm Screen

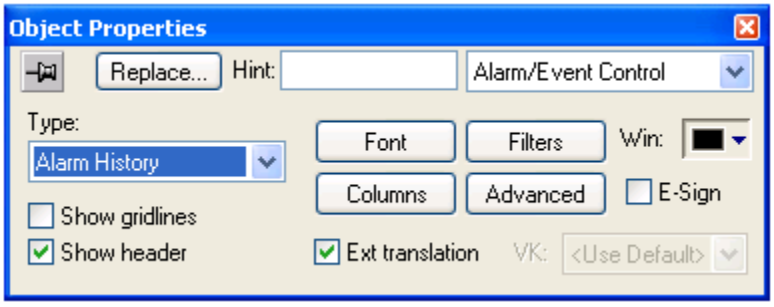
Save the screen and continue to the next section.

Exercise: Creating the Historical Alarm Screen

To create a *Historical Alarm* screen, use the following procedure:

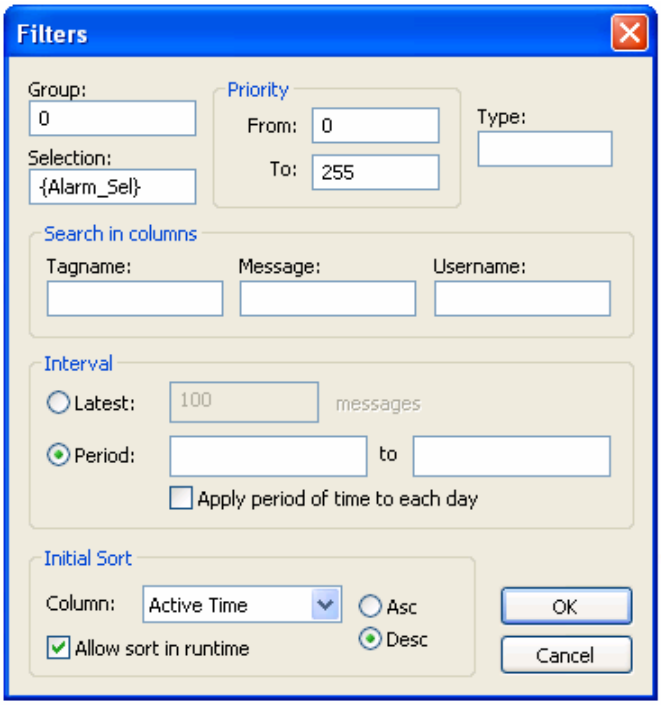
Open the *AlarmOnLine* screen and save it as **AlarmHistory.scr**.

Double-click the object to open the *Object Properties* dialog and configure the object using the following figure. Be sure to select the **Alarm History** option.



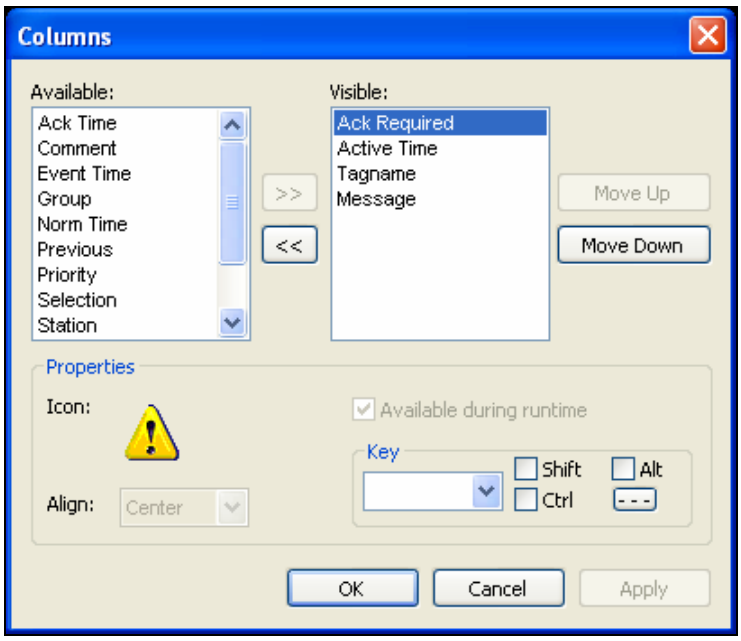
Alarm History Object Properties

Click the **Filter** button and use the following figure to configure the filter**:**



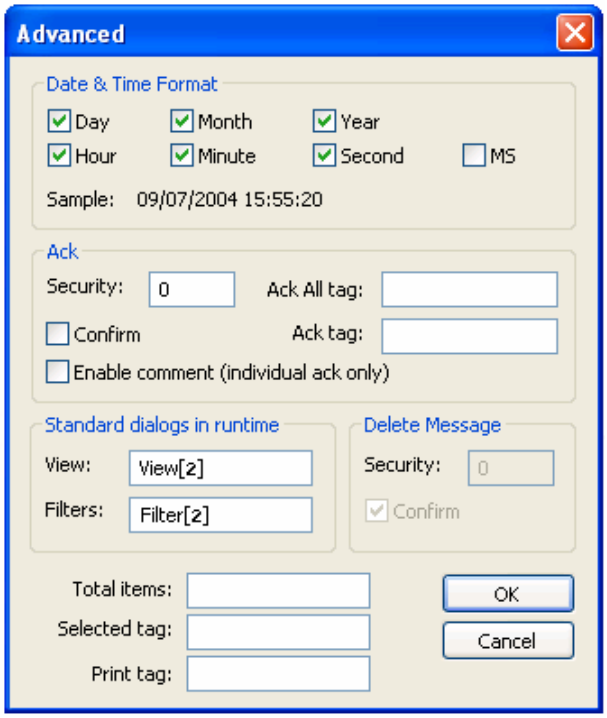
Configuring the Alarm History Filters

Click the **Column** button and use the following figure to configure the filter**:**



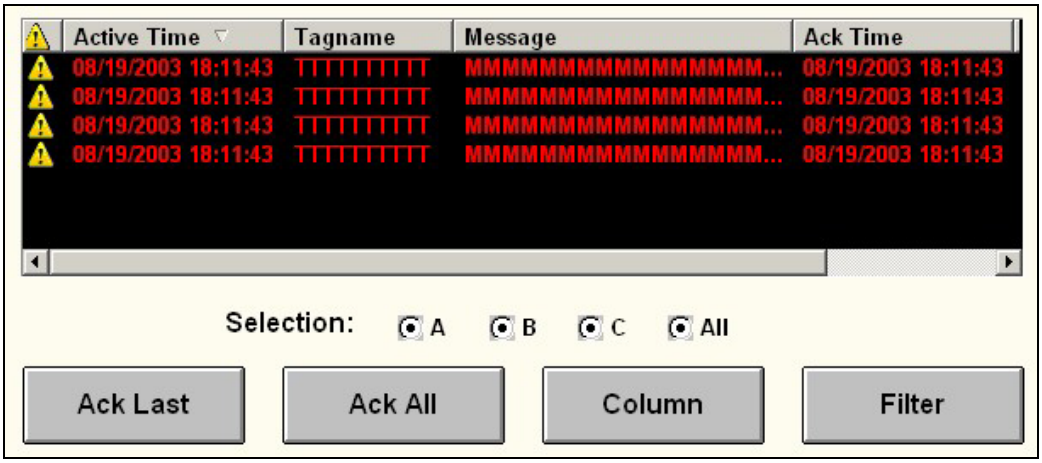
Configuring the Alarm History Columns

Click the **Advanced** button and use the following figure to configure the filter:



Configuring the Alarm History Advanced Settings

Your screen should now look like the following figure:



Completed Historic Alarm Screen

Save the screen and run the runtime.

Check the color differences according to the alarm states.

Notes

Chapter 19. Communications

Communications (for this class) means to read values from a devices' memory (e.g. Registers) to application variables called *tags*, or write values from the application tags, to the devices' memory.

In this part of the class you will learn how to set up PLC, OPC and TCP/IP communications.

Using PLC Drivers

The driver is a part of the InduSoft Web Studio Software, and its function is to establish the communication between different devices and InduSoft Web Studio software.

The following is a sample of the drivers available to the WinNT/2000 and CE operating systems:

Allen Bradley: DF1
Siemens: S5 – AS511 PG Port
Profibus DP Master and Slave (Hilscher)
Allen Bradley: ControlNet Slave
OMRON: Host Link
GE FANUC: SNP, 90-30 90-70 Series
Modbus: Schneider 984 series
Profibus DP Master
Cutler-Hammer: D50 – D300 Series
Hitachi: H series
Toshiba: Prosec T1/T2

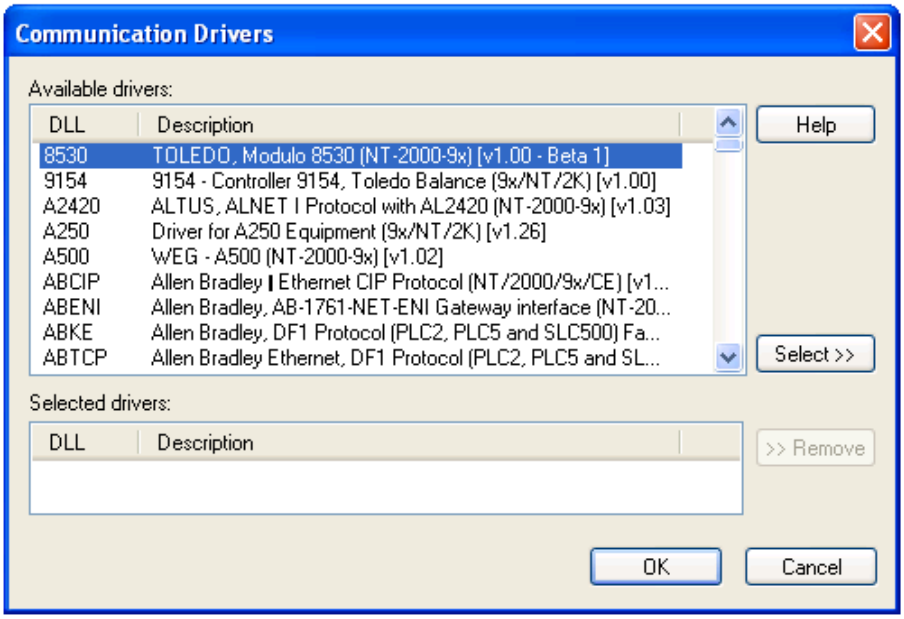
Configuring a PLC Driver

When you installed InduSoft Web Studio, you also installed all the drivers.

Use the following process to configure any driver:

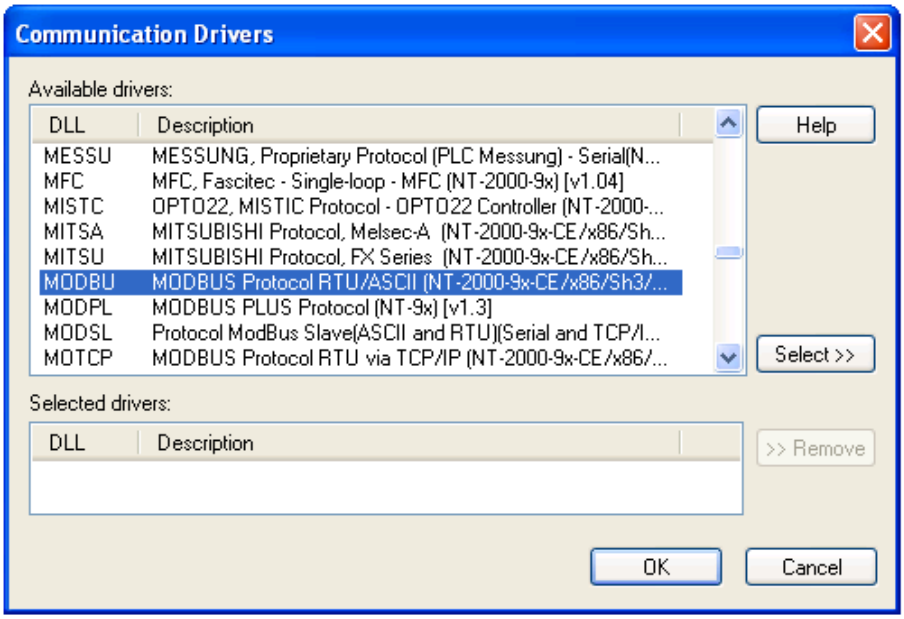
Select the **Comm** tab (in the *Workspace*) and right-click on the *Drivers* folder.

Select **Add/Remove drivers** from the pop-up menu to open the *Communication Drivers* dialog.



Communication Drivers Dialog

Select a driver (for this exercise, select the **MODBU Protocol ModBus** driver) from the **Available Drivers** list pane, click the **Select** button, and then click **OK** to close the dialog.

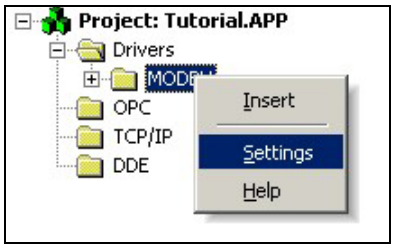


Selecting the MODBUS Driver

Next, configure the serial channel and other parameters for the driver.

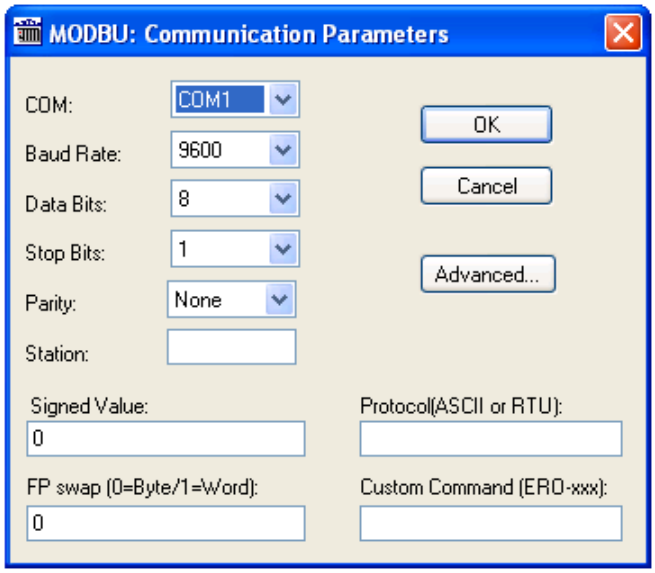
> **Note:**
> The values you configure will be stored immediately in a system-related file. *You cannot save or cancel this operation.* Any changes you make will take effect only after the driver is initialized. Consequently, if the driver is running and you change any of the driver parameters, your changes will not take effect until you close and reopen the driver.

Open the *Drivers* folder in the *Workspace* and right-click on the *MODBU* folder. Select **Settings** from the pop-up.



Select the Setting Option

When the *Communications Parameters* dialog displays, set the parameters specified in the following table.



MODBUS Communications Parameters Dialog

Parameter	Default Value	Valid Values	Description
COM	COM1	COM1 to COM8	Serial port of the PC used to communicate with the device (if it is a serial driver).
Baud Rate	19200	110 to 57600bps	Communication data rate
Data Bits	8	5 to 8	Number of data bits used in the protocol
Stop Bits	1	1or 2	Number of stop bits used in the protocol
Parity	Odd	even odd none space mark	Protocol's parity
Station	0	0	Number, computer name, or network unit if the protocol requires it.

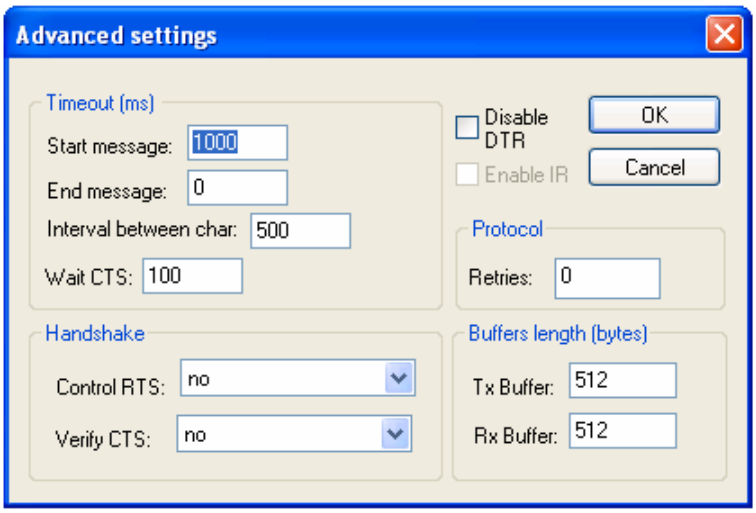
Note:
You *must* configure your device with the *same* values defined in the *Communication Parameters* dialog.

The four fields at the bottom of this dialog are different for every driver and are configured with different functions for each driver. For the ModBus driver, the fields are **Signed Value**, **FP swap (0=Byte/1=Word)**, **Protocol (ASCII or RTU)**, and **Custom Command (ERO-xxx)**.

If you type an invalid entry in these fields, IWS will accept the value, but when you try to close the *Communication Parameters* dialog, an error message displays and prevents you from closing the dialog.

SPECIFYING THE ADVANCED SETTINGS

Clicking the **Advanced** button in the *Communication Parameters* dialog makes it possible to configure other parameters for the serial communication (as noted in the following table):



MODBU Driver Advanced Settings

Parameter	Default Value	Valid Values	Description
Start message (ms)	1000	0 to 10000	Maximum time to receive the beginning of the answer from the device (time-out time)
End message (ms)	0	0 to 10000	Maximum time to receive the end of the answer from the device since the beginning of the answer. (***Note***: Entering a zero value means the driver will not check these times)
Interval between char	500	0 to 10000	Maximum time between characters sent from the device
Wait CTS (ms)	100	0 to 10000	Maximum time to receive the *CTS* (Clear to Send) signal after setting the *RTS* (Request to Send) signal (***Note***: Valid only if you specify **Yes** for the **Verify CTS**

Parameter	Default Value	Valid Values	Description
			parameter).
Control RTS	No	•no •yes •yes+echo	Define if the *RTS* (Request to Send) handshake signal must be set before a communication and if it will have echo in the communication.
Verify CTS	No	•no •yes	Define if driver must wait for a *CTS* (Clear to Send) handshake signal before sending a message.
Disable DTR	Not checked	•Not checked •Checked	If checked (*enabled*), the driver will not set the DTR signal before starting communication.
Retries	0	0 to 5	Communication retries number by each tag configured in the driver worksheets, in failure case.
Tx Buffer (bytes)	512	0 to 512	Maximum size of information buffer to be sent from the driver.
Rx Buffer (bytes)	512	0 to 512	Maximum size of information buffer to be received from the host.

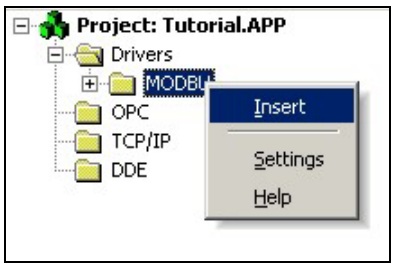
Note:
Generally, you change these parameters only when using a *DCE* (Data Communication Equipment) converter (232/485, for example), modem, and so forth between the PC where the driver is running and the Host. You must know the features of the DCE before setting the *Advanced* parameters. The advanced communication parameters are the same for all the *Driver Configuration Worksheets*.

Adding a New Driver Worksheet

Use the following instructions to create a new *Driver* worksheet:

Right-click on the *Drivers* folder, and select **Insert** from the pop-up.
(For this example, we will create a new *Modbu Driver Worksheet*)

Alternatively, you could select **Insert → Document → *XXXX Driver Worksheet***
(where ***XXXX*** is the name of the driver).



Inserting a MODBU Driver Worksheet

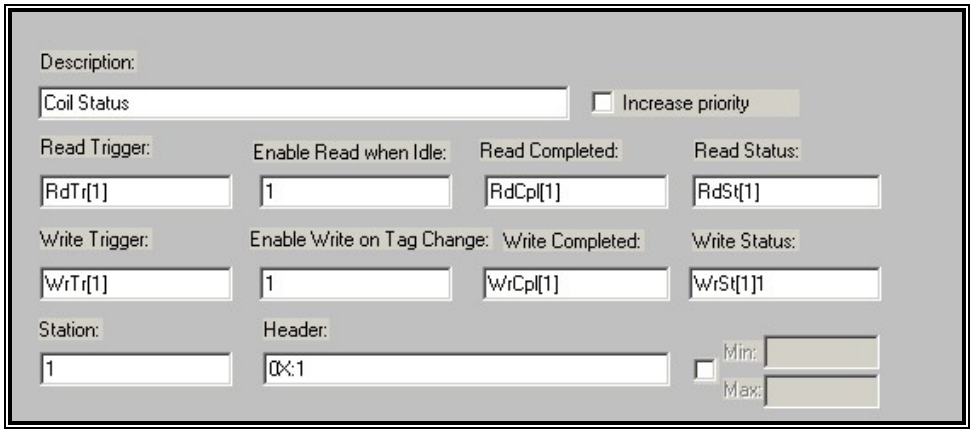
Driver worksheets contain two sections:

Header: Contains all the information about the read and write 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.



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 are 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 the tag value 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 tag changes value.

> ➲ **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 isany kind of hole in the worksheet that can cause problems for the system or the PLC's program.

Enable Write on Tag Change: Contains a tag that, when its value is greater than zero, IWS writes the changed tag in the body's worksheet, when the tag value is different from the write trigger.

Write Complete: 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 *Application* 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.

In our example case, let's use the MODBUS syntax, which is:

<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 run-time 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 *Application* 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 standard **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 check (*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.

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.

Note:
For read operations:
<tag> =((<value in the equipment>) / Div) + Add

For write operations:
<value in the equipment> = (<tag> - Add) * 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.

Exercise: Preparing the Application for Driver Runtime

Use the following steps to specify *Header* tags:

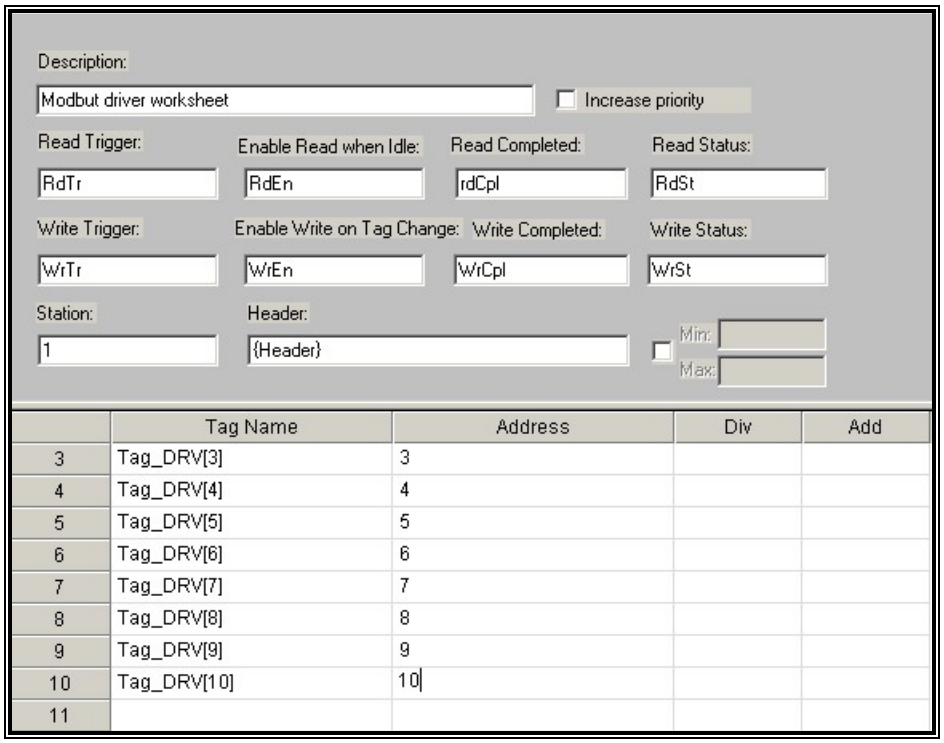
Specify the following tags in the *Driver* worksheet **Header** fields. All of the tags will be arrays, and you must type each element on each worksheet.

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, not in the test's beginning, on the "Header" field
Header	0	String	String tag that will be, not in the test's beginning, on the "Station" field

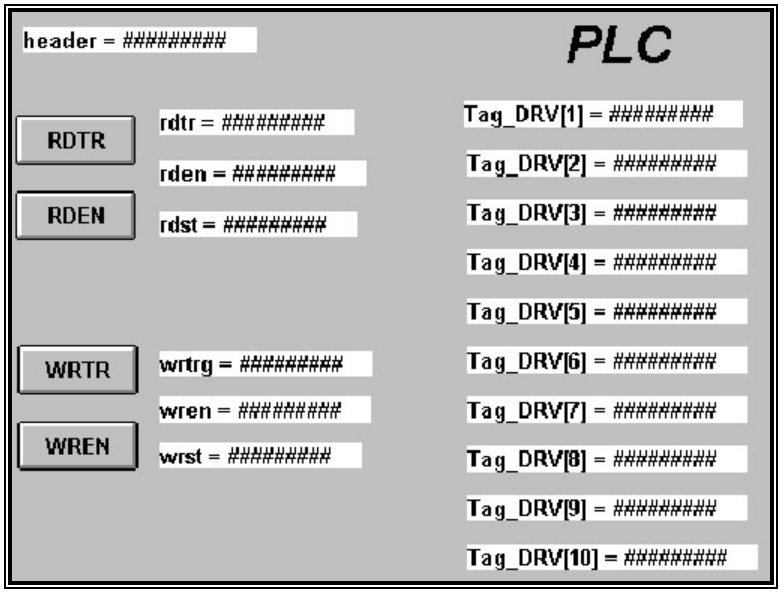
Specify **TAG_DRV**, size 10 for the communication tags.

Configure a *Driver* worksheet and a *PLC Driver* screen to look like the following figure:



	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 MODBUS Driver Worksheet



PLC Driver Screen

Running the Application and Monitoring the Driver

Now you are ready to run the application and to monitor your driver. Use the following tests.

Test 1: Using the Read Trigger

Change the value of the Tag **RdTr[1]**.

Verify if a read event occurred by checking if **RdCpl[1]** value has toggled. If so, check whether the values are the same as the PLC, to all the areas.

Also check with the Tag **RdCpl[1]** changes its value each times a read event is finished, and if the Tag **RdSt[1]** keeps its null value.

Test 2: Using Enable Read When Idle

Set the value of the tag **RdEn[1]** tag to **1**.

Verify that events are being read continuously. If so, check whether the values are the same as the PLC in all the areas.

Also check that the **RdCpl[1]** tag changes values each time a read event is finished, and that the **RdSt[1]** tag maintains its null value.

Test 3: Write Trigger

Change the value of the **WrTr[1]** tag.

Verify that there is a write event. If so, check whether the values are the same as the PLC, in all the areas using the PLC programming software.

Also check that the **WrCpl[1]** tag changes its value each time a write event is finished, and that the **WrSt[1]** tag keeps its null value.

Test 4: Write on Tag Change

Set the value of the **WrEn[1]** tag to **1**.

Change the value of the **TAG_IR[1].W** tag and the **TAG_IR[1].b0** tag.

Verify that a write event occurred for each change. If so, check that the new value is in the PLC and if the other operators on the same worksheet have not been written to the PLC.

Also check with the **WrCpl[1]** tag changed its value each time a write event is finished, and if the **WrSt[1]** tag kept its null value.

OPC Communication

The InduSoft Web Studio *OPC Client* module enables the InduSoft Web Studio system to communicate with any device that implements an OPC Server. This module implements the OPC standard as described in the document "OLE for Process Control Data Access Standard," available at the site *http://www.opcfoundation.com*.

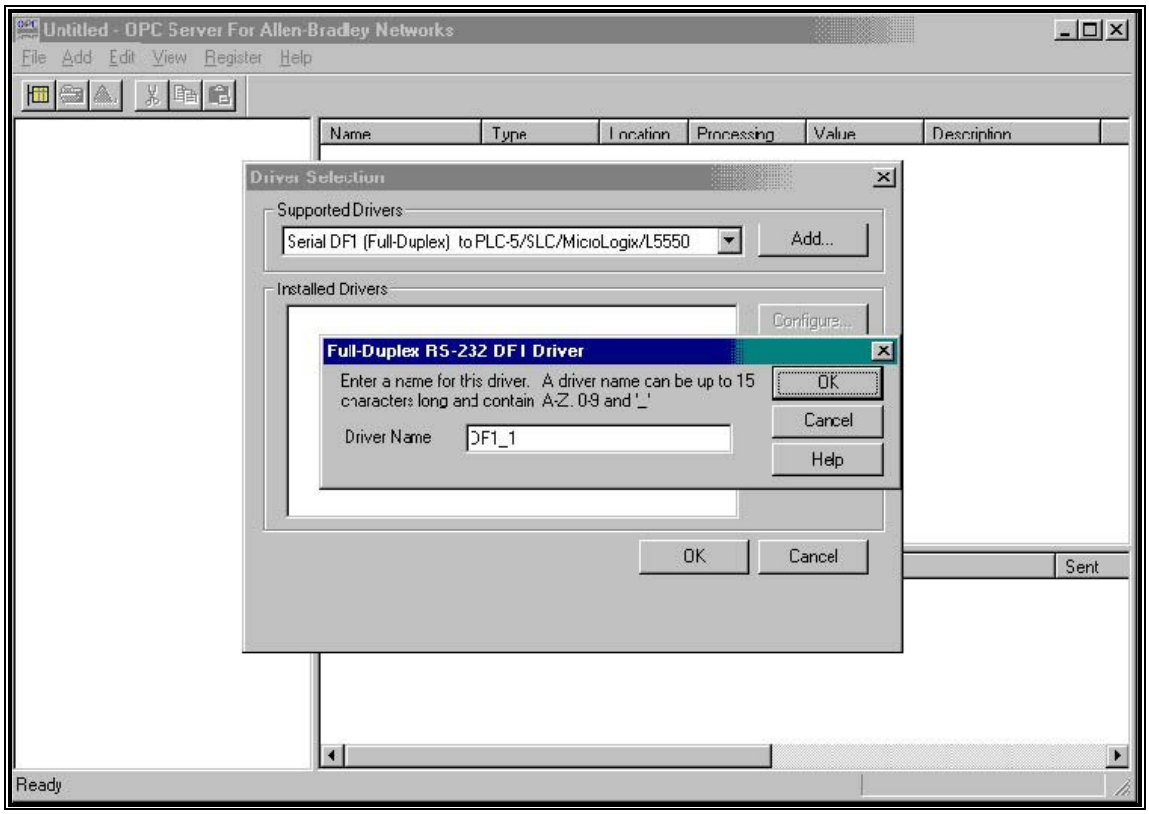
Before you start using the OPC Client Configurator, you must install the OPC Server.

For this class, install the *INGEAR Allen Bradley OPC Server*.

After installing the server, you must configure an *OPC Server* database, and name it *INGEAR AB OPC Server*.

Click **Add** → **New Device**. You will be prompted to create a new driver. Accept it.

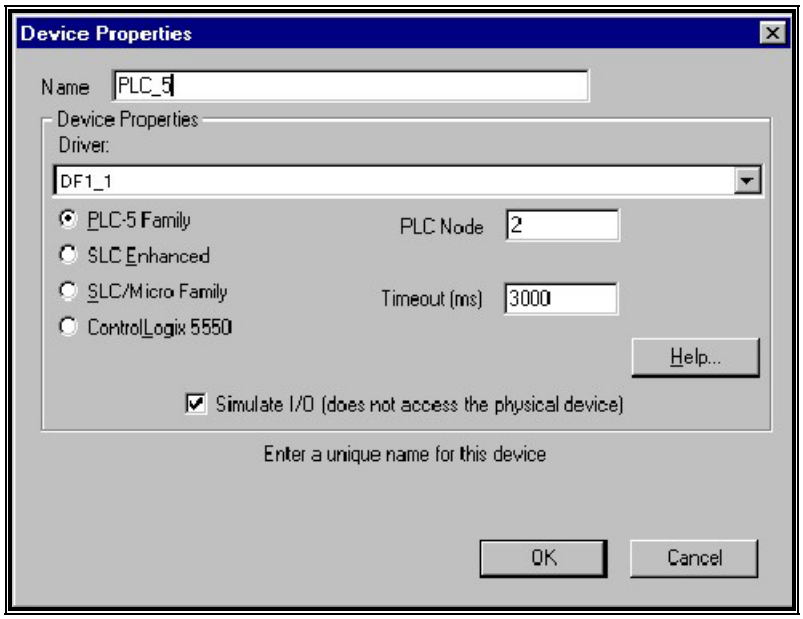
The first option is the **DF1 Serial**. Because the communication method does not matter (we are going to simulate a communication), you can choose the **DF1**.



Adding a New Driver

Accept the next steps, such as driver name and communication settings.

Configure the *Device Properties* dialog as follows. Note that the **Simulate I/O** check-box is checked (*enabled*):

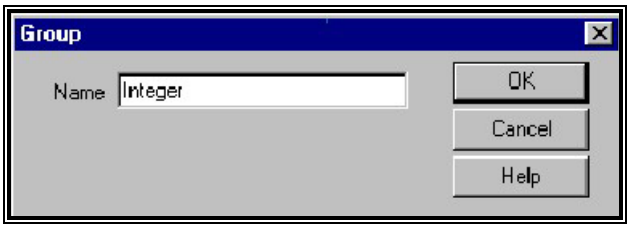


Device Properties Dialog

Now you can add the new OPC Server items.

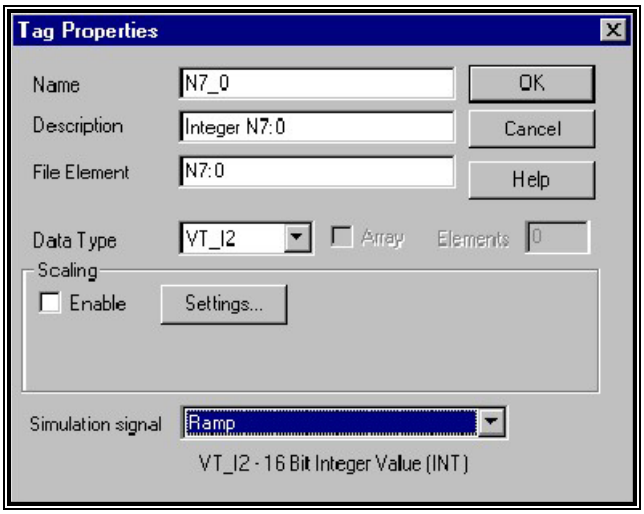
You must tell the OPC Server with which PLC address to communicate. Organize it in groups, sub-groups, and so forth. For this example, just create two groups.

Select **Add → New Group** and specify the **Name** Integer.



Click **Add → New Tag**. (***Note*:** New tag in this case indicates an OPC Server item; it has nothing to do with InduSoft Web Studio tags.)

Configure the new tag as follows:



Tag Properties Dialog

Create at least two more tags as part of the *Integer* group, with elements such as **T4:0.ACC**, **C5:0.PRE**, and so forth.

Create a new group, called *Boolean* and create new tags such as **B3:0/200**, **I:0/5**, **N7:100/4**, and so forth.

Save your work in the *FirstTutorial* folder, as shown in the following figure:

Click the **View Monitor** option.

Now that you created an OPC Server database, you can start configuring the InduSoft Web Studio *OPC Client Worksheets*.

Configuring the InduSoft Web Studio OPC Client Worksheet

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*.

Before you can use a control (or a COM Server), you must create specific entries in the Windows registry to declare its availability to the OLE client application.

Select the **Database** tab and create a new set of tags to communicate with the OPC Server, as shown in the following figure (review page **Error! Bookmark not defined.** if necessary):

Application Tags

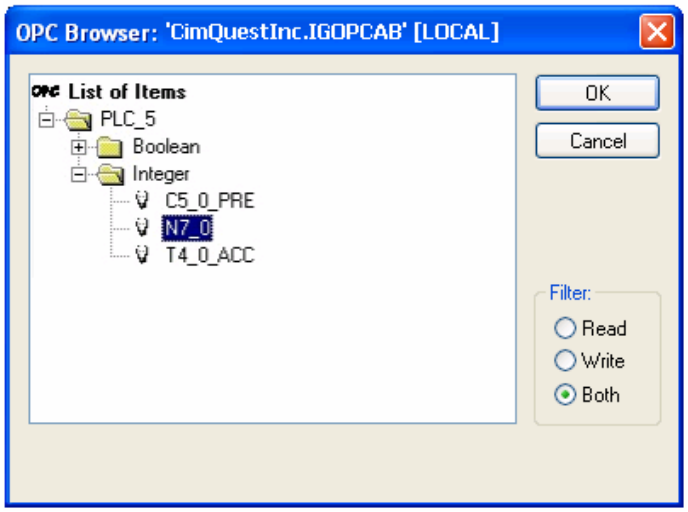
	Name	Size	Type	Description	Scope
9	OPC_Status	0	Integer	OPC Server status	Server
10	OPC_N7_0	0	Integer		Server
11	OPC_T4_0_ACC	0	Integer		Server
12	OPC_C5_0_PRE	0	Integer		Server
13	OPC_B3_200	0	Boolean		Server
14	OPC_I_0	0	Boolean		Server
15	OPC_N7_100_4	0	Boolean		Server
*			Integer		Server

Creating Tags to Communicate with OPC Server

In the *OPC Client* worksheet, type **OPC_Status** in the **OPC Status** field.

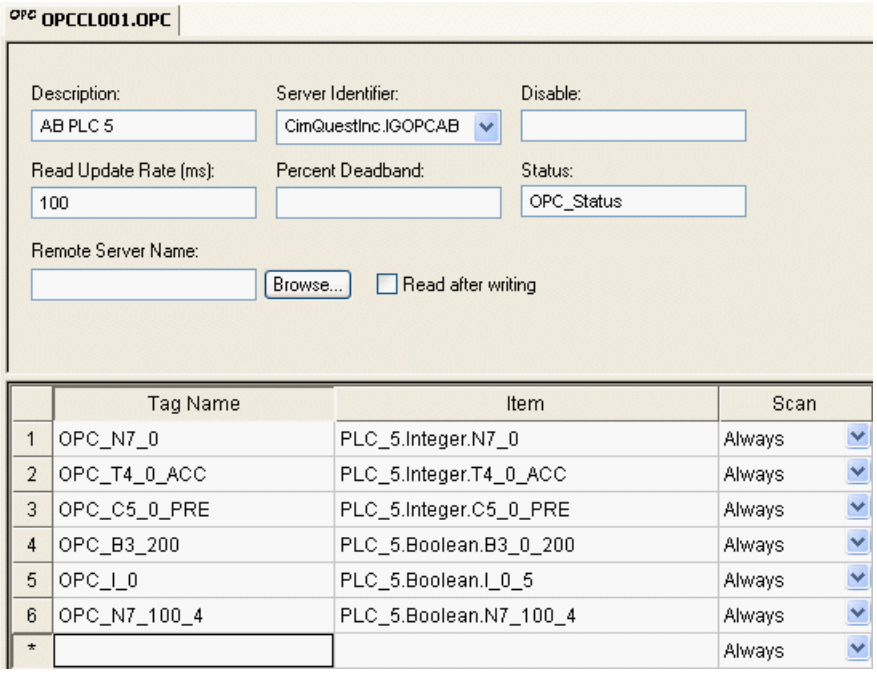
In the first **Tag Name** column row, type **OPC_N7_0**.

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.



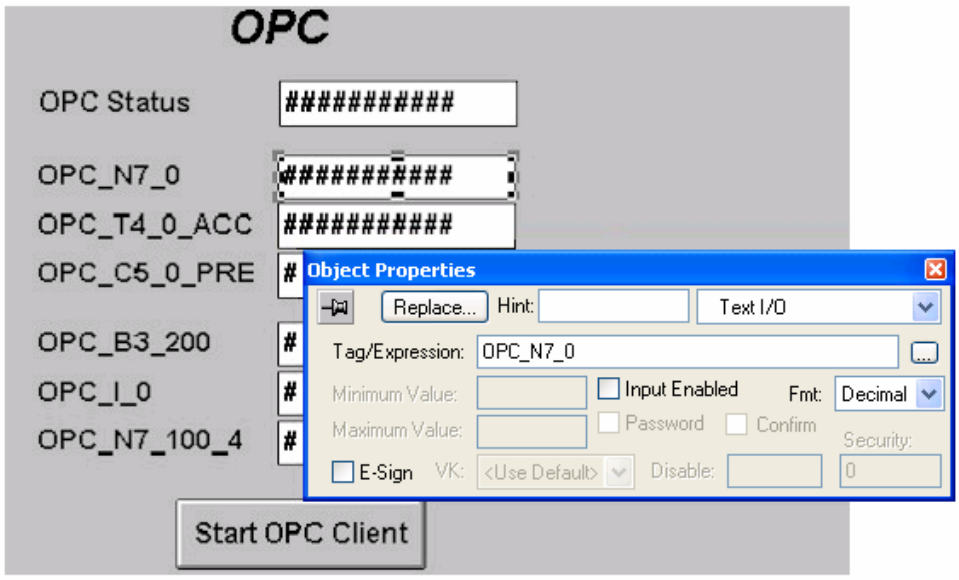
Select N7_0

Your *OPC Client* worksheet should look like the following:



OPC Client Worksheet

Your *OPC Client* screen should look like the following figure:



OPC Client Screen

All the OPC Clients start their OPC Server when starting up. As this OPC Server is just a demo and it opens a dialog, it can cause some errors during the Start Up. That is why we have a push button to start the **OPC Client Runtime** task.

Run the application and check the OPC screen behavior with the OPC Server values.

InduSoft Web Studio allows you to save your application screens in HTML format and export them to Internet Browsers (Internet Explorer). We will discuss this procedure in Chapter 20.

TCP/IP Communication

The IWS *TCP/IP Client/Server* modules enable two or more InduSoft applications to keep their databases synchronized. These modules use 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 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. 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 the Client

On the client machine, you must use the *TCP/IP Client Configuration* program 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 Configuration* 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.

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.

RUNNING THE TCP/IP CLIENT MODULE

You can choose to run the *TCP/IP Client* module automatically or manually. Select **Project → Status**, **Runtime Tasks Table**, and enter **TCP/IP Client**.

After running this program, a small icon will display in your system tray.

Setting Custom Parameters

You can configure the following parameters for the **Application Configuration** (**.app**) file:

[TCP] Port: TCP/IP port number. Default is **1234**. This parameter should be the same in both the client and server machines.

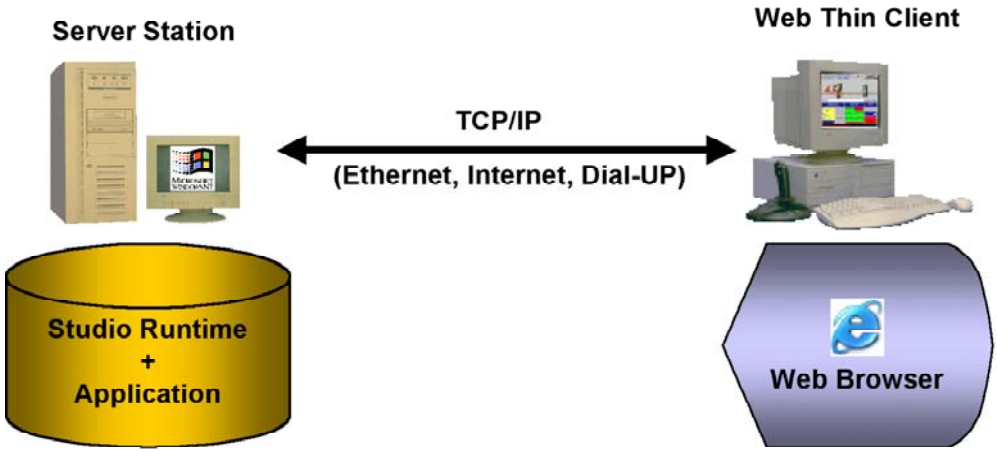
SendPeriod: Time in milliseconds before the client/server module will update the tag values of the other machine. Default is **250**.

ConnectRetryTimeout: Time in seconds before the client should retry to connect to the server. Default is **30**. Only the client module uses the **ConnectionRetryTimeout**.

Chapter 20. Running Your Web-Based Application

In this section, you will learn how to configure an application to run on the Web by saving it in HTML format and then exporting the file to a browser.

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).

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/2K/XP:

  Copy the files **ISSymbolReg.exe** and **ISSymbol.cab** from the **\BIN** sub-folder of **Indusoft Web Studio v6.1** and paste them in any directory of the Web Thin Client station. Make sure that both files are stored in the same directory.

  Run **ISSymbolReg.exe** to register ISSymbol control in the Web Thin Client station.

- Windows 9x/ME:

  Copy the files **ISSymbolReg.exe** and **ISSymbolA.cab** from the **\BIN** sub-folder of **Indusoft Web Studio v6.1** and paste them in any directory of the Web Thin Client station. Make sure that both files are stored in the same directory.

  Run **ISSymbolReg.exe** to register ISSymbol control in the Web Thin Client station.

  Caution:
  Windows 9x/ME do not support UNICODE characters. Therefore, UNICODE fonts will not be properly displayed on Web Thin Clients running under Windows 9x/ME.

- Windows CE:

  Copy the file **ISSymbolCE.ocx** and **IndHTTP.dll** from the **\Redist\<OS Version>\<Processor Type>** sub-folder of **Indusoft Web Studio v6.1**, and paste them in any directory of the Web Thin Client station.

  Execute the following command from the Prompt window: **regsvrce.exe "\<ISSymbolPath>\ISSymbolCE.ocx"** (e.g. **regsvrce.exe "\Storage Card\ISSymbolCE.ocx"**)

Save the registry settings to keep **ISSymbolCE.ocx** registered when you reboot the Windows CE device.

- Windows CE PocketPC:

  Copy the files **RegSvrCE.exe, ISSymbolCE.ocx** and **ndHTTP.dll** from the **\Redist\<OS Version>\<Processor Type>** sub-folder of **Indusoft Web Studio v6.1** and paste them in any directory of the Web Thin Client station. Make sure that both files are stored in the same directory.

  Execute the **RegSvrCE.exe** program on the Web Thin Client device. To register **ISSymbolCE.ocx**, do the following:

  - Select the **\<ISSymbolPath>\ISSymbolCE.ocx** file
  - Select the option *Register*
  - Click the *OK* button

> **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 encrypted 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.

Configuring a Web-Based Application

The main steps to configure a web-based application with IWS are described below:

1. **Configure the web server**: The web server is a HTTP server driver that is able to provide files to remote stations via the HTTP protocol over TCP/IP. IWS supports any web server; however, if your architecture demands the *Web Gateway* designed by Indusoft, the web server must be IIS (Internet Information Services) from Microsoft.

   Configuring a web server for an IWS application is basically making sure that the web server is running and assigning a home directory (web root) to it. Usually, the home directory should be configured with the path for the **\Web** sub-folder of the application. Consult your web server's documentation for further information about its configuration.

   Notes:
   - InduSoft provides a web server for Windows NT/2K/XP (**NTWebServer.exe**) that is stored in the **\BIN** sub-folder of IWS after installation. Furthermore, InduSoft provides a web server for Windows CE (**CEWebServer.exe**) that is stored in the **\Redist\<WinCE version>\<Processor Type>** sub-folder of IWS after installation. The home directory (web root) for the web servers provided by InduSoft is the directory from where they are executed. Therefore, InduSoft recommends copying these web servers to the **\Web** sub-folder of your application before running them.
   - NTWebServer and CEWebServer were designed primarily for simple tests and/or for demos. InduSoft recommends using third-party commercial Web Servers such as IIS (Internet Information Services) from Microsoft or Apache (for Linux) in real-world applications.

2. **Configure the web settings in the IWS application**: The web settings for the IWS application are configured in the **Project Settings → Web** dialog window.
3. **Save the screens as HTML**: The screens that should be available for the Web Thin Client stations must be saved as HTML. To do so, open the screen on the development environment and execute the **File→Save as HTML** command from the menu. Use the **File→Save Screen Group as HTML** menu option to save screen groups (***.sg**) as HTML and make them available for the Web Thin Clients. After you save any screen as HTML once, the web files for this screen are automatically updated whenever it is saved again (**File→Save**).

   Tip:
   If you want to make all screens from your application available for the Web Thin Client stations, execute the **File→Save All as HTML** menu option.

   Caution:
   After changing any setting in the **Project Settings** dialog window and/or in the **Tags Database**, you must execute the **Tools→Verify Application** command to update the web files with the new settings.

4. **Run the application on the server**: Make sure the TCP/IP Server module is running on the data server. The TCP/IP Server module is embedded in IWS, and it is automatically executed whenever you run any application for the Windows CE operating system. For Windows NT/2K/XP, you can configure the TCP/IP Server module to be executed automatically when the application is started,

using the **Project Status→Execution Tasks** dialog. The TCP/IP Server module is the data server for the remote Web Thin Client station(s).

5. Depending on your architecture, you may need to run the *Web Gateway* designed by Indusoft in the web server station.

Typical 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**: Software that implements the HTTP protocol (server) over TCP/IP; e.g. web server from the IIS from Microsoft.
- **Server station**: Computer or device running IWS and a web server. The IWS application must be stored in this computer.
- **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).

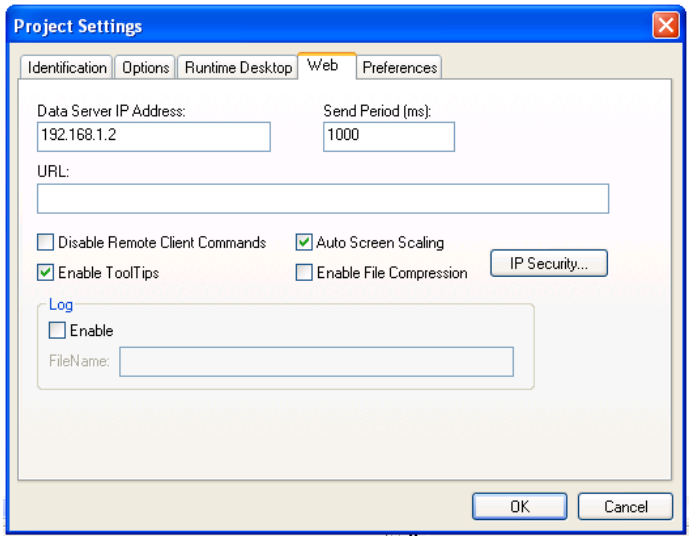
Configuration example:

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**

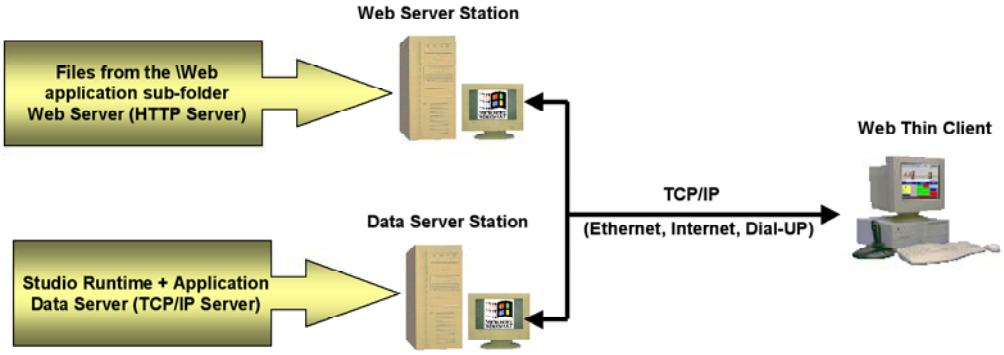
The **Project Settings → Web** interface must be configured as follows:



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 traffice 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).

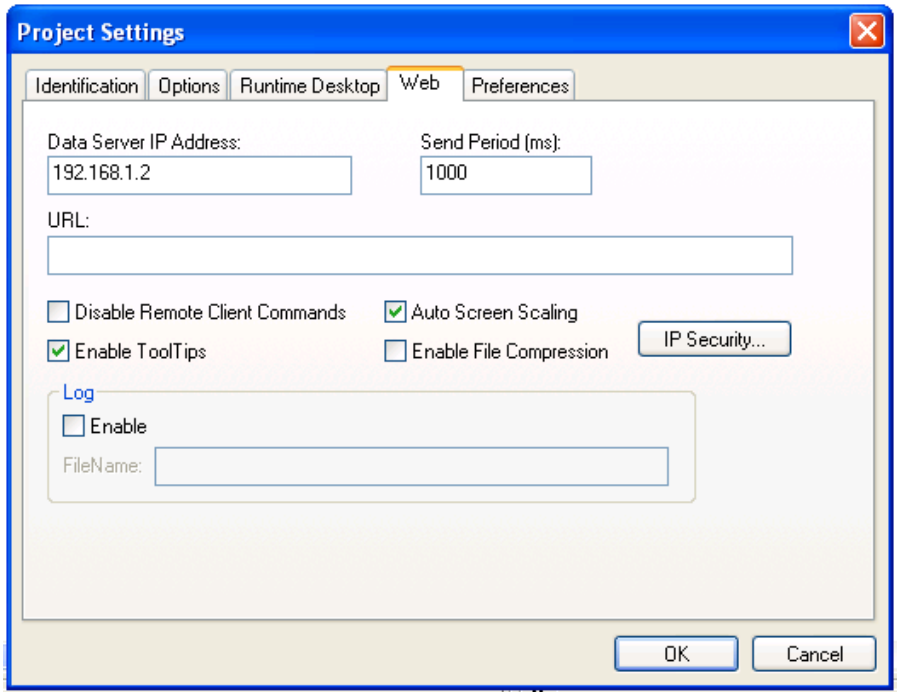
Configuration Example:

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

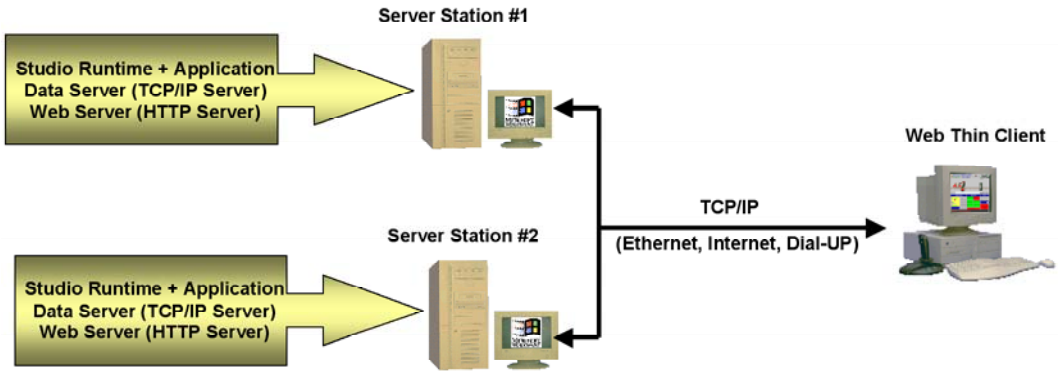
You must type the following address on the remote web browser to access a screen (e.g. **myscreen**) from the server: **http://192.168.1.1/myscreen.html**

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:

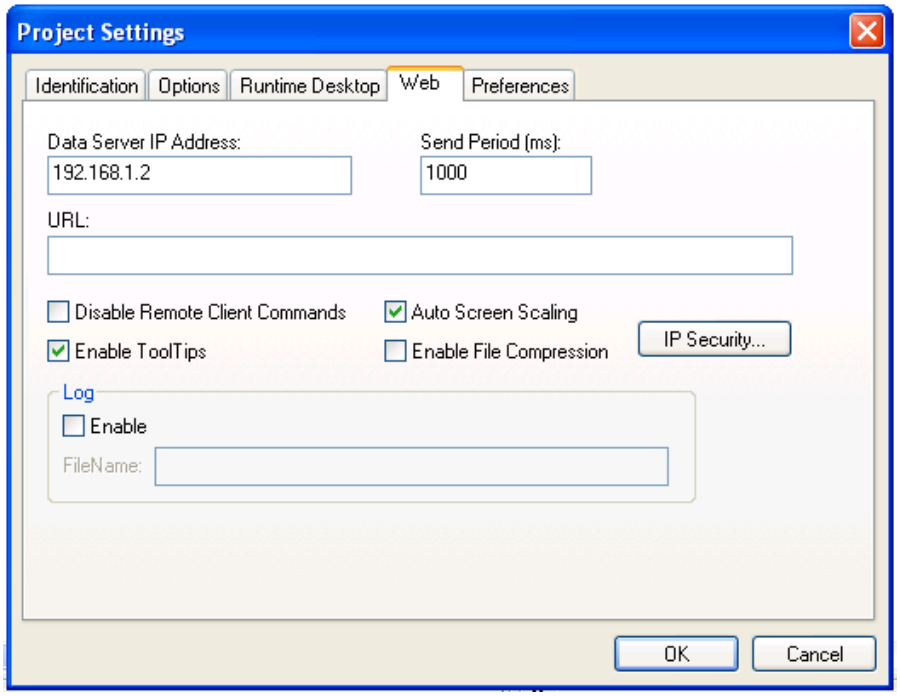
- IP address of server station #1 on the network: **192.168.1.1**
- IP address of server station #2 on the network: **192.168.1.2**
- Home directory of the web server (HTTP server) on server station #1: **\Web** sub-folder of the application stored on server station #1.
- Home directory of the web server (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**

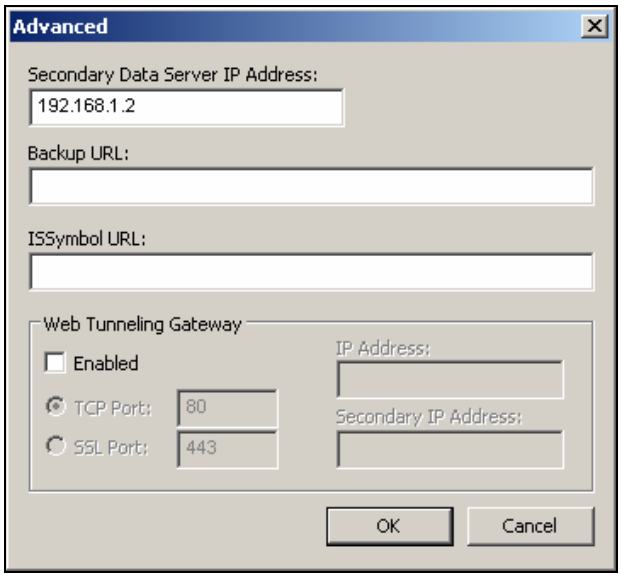
The user must type the following address on the remote web browser to access a screen (e.g. **myscreen**) from server station #2: **http://192.168.1.2/myscreen.html**

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.*

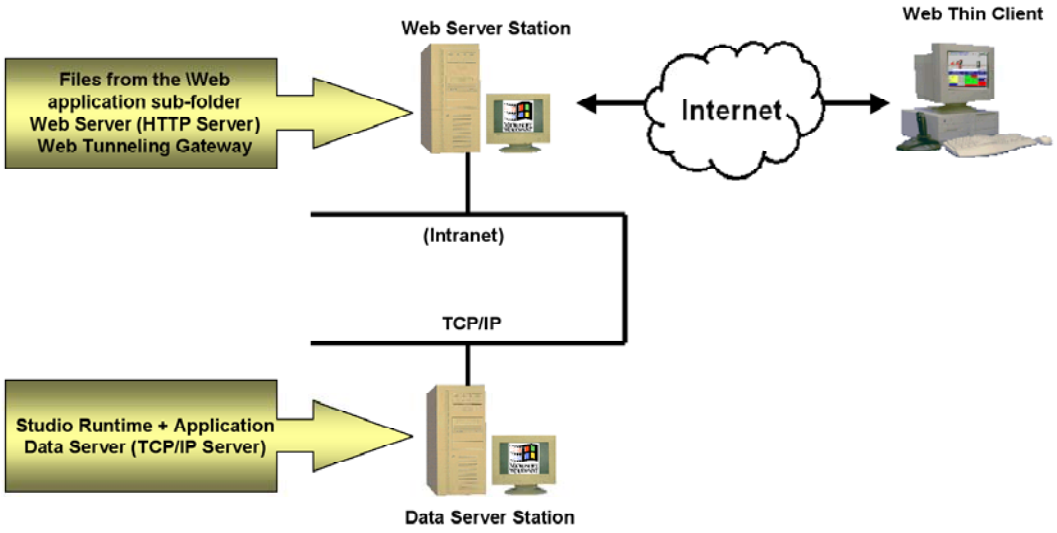


Project Settings → Web Interface



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.

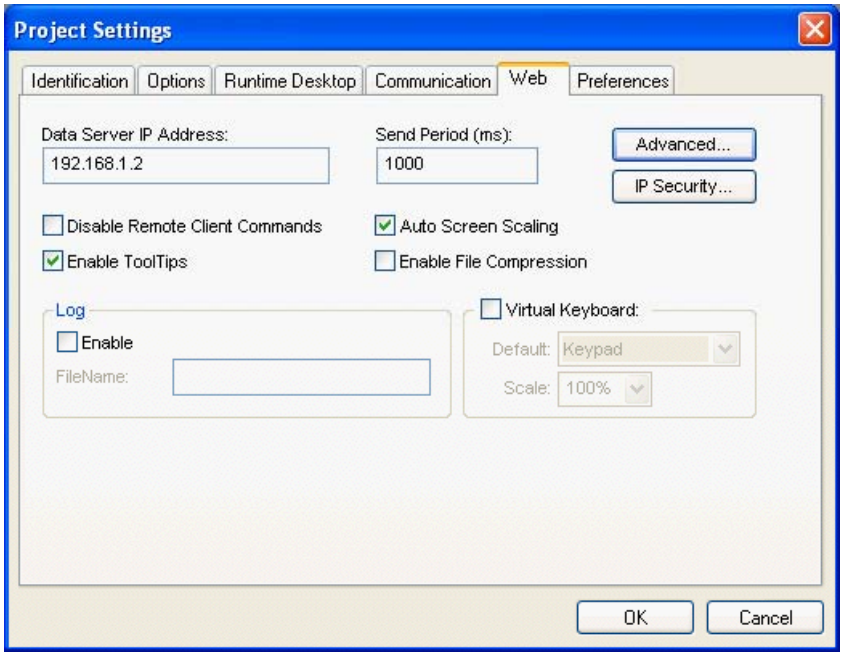
Configuration example:

This example is based on the following premises:

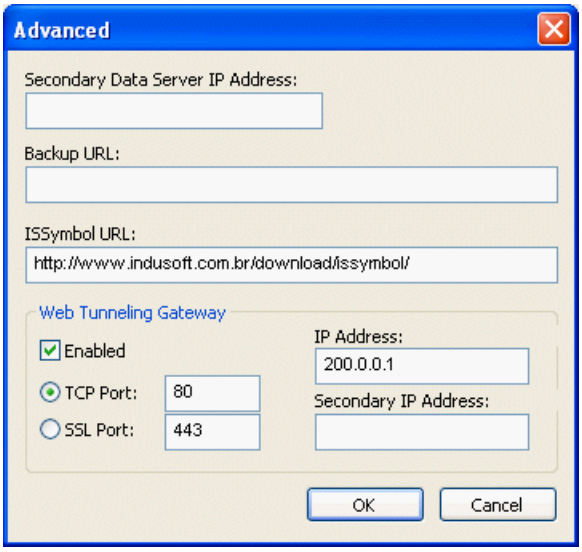
- IP address of the web server station (internet): **200.0.0.1**
- IP address of the web server station (intranet): **192.168.1.1**
- IP address of the data server station on the intranet: **192.168.1.2**
- Home directory of the web server (HTTP server) on the web server station: **\Web** sub-folder 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

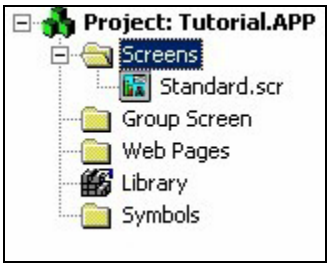
Note:
If your web server is able to provide files via HTTPS (SSL – Secure Socket Layer), you can select this option on the *Advanced* dialog window from the **Project Settings → Web** interface.

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.

Exercise: Viewing Your Application on the Web

To view your application, use the following steps:

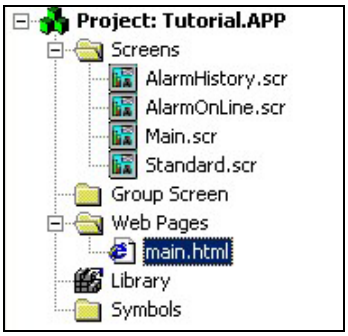
Expand the *Screens* folder and double-click on your *Main.scr* screen.



Selecting a Screen

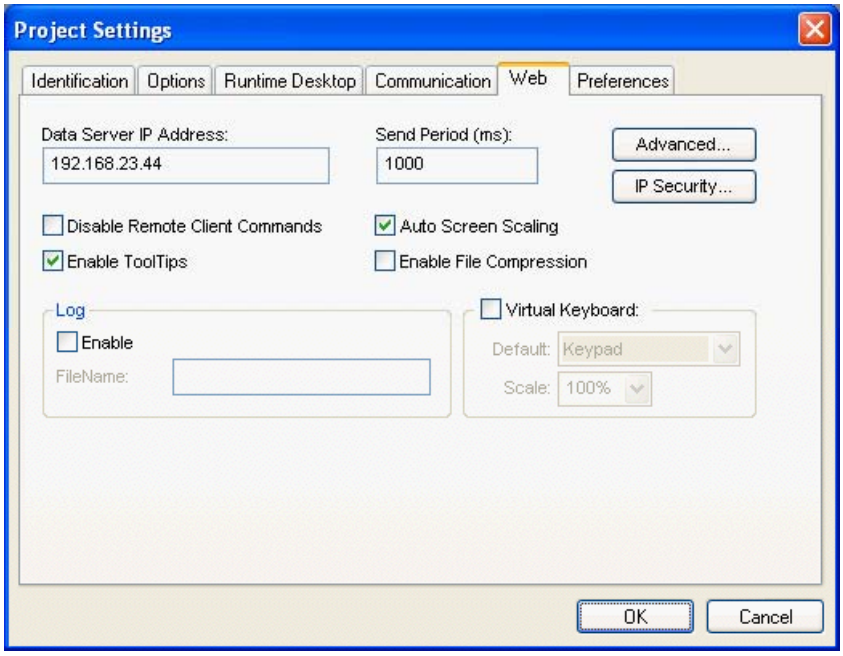
Select **File →Save as HTML** to save the screen in HTML format.

Web files are stored in the *Web* folder, so open the folder and verify that you saved the *Main* screen successfully. You should see **main.html**.



main.html

Select **Project→Settings** from main menu bar, and then select the **Web** tab.



Open the Project Settings Dialog

Configure the **Data Server IP Address** to use the IP Address of the Server station (computer on which you are running) at runtime.

The Web Thin Client station exchanges on-line data (tag values) with the station specified in this field.

Type the URL path to your **main.html** file (in the *Web* folder) into the **URL** field.

The URL depends on the Home directory configured in the Web Server of your Server station.

> **Note:**
> Microsoft provides Web Servers for any Microsoft operating system. Consult your Microsoft documentation about installing and configuring a Web Server.

After configuring the Web settings, click **OK** to close the *Project Settings* dialog.

Close all screens in the screen editor (**Window → Close All**) and execute the **Tools →Verify Application** command to update the Web settings to your Web page.

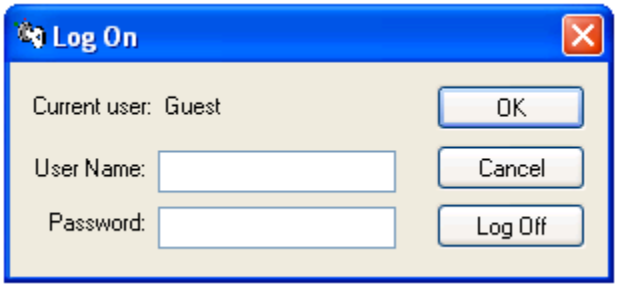
> **Caution:**
> You must execute the **Tools →Verify Application** again after changing any settings in the **Project Settings** menu.

To test your Web-based application, use the following steps:

Click the **Run Application** button ( ) to execute the application locally on the Server station.

Open an Internet Browser (Microsoft *Internet Explorer* or *Netscape*) and type the URL address to open your main.html screen from the Server station.

When the *Log On* dialog displays in the Browser, type **guest** into the **User Name** field, then click **OK** to open the main.html screen in the Browser.



Log On Dialog

Notice that you can modify the level of any tank locally (Server station) using the Viewer run-time module or remotely (Web Thin Client) using the Browser.

Note:
A Web Thin Client requires an ActiveX component (**ISSymbol.ocx**) to handle screens on the Browser. If you connect the Web Thin Client to the Internet, this component is downloaded and registered automatically.

Otherwise, you must copy the **ISSymbol.cab** from the *BIN* folder and paste it into the **\<OSPath>\System32** directory on the Web Thin Client workstation. Use the *WinZip* utility to unzip (*extract*) the files from **ISSymbol.cab** into the **\<OSPath>\System32** directory and register the **ISSymbol.ocx** using the **regsvr32 ISSymbol.ocx** command.

NOTES

Chapter 21. Managing Applications Remotely

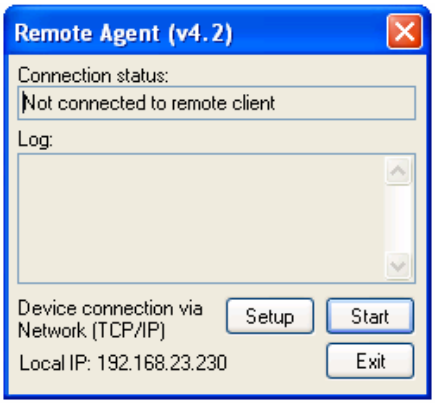
After configuring an application and testing it locally (on your development workstation), you can download the application to a remote run-time workstation that is running under Windows NT/2000/XP or CEView under Windows CE.

Exercise: Configuring the Remote Agent

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 NT/2000/XP computers:
>
> **\InduSoft Web Studio\Redist\CEView\<Processor Type>\BIN**
>
> The file should be located in the **\<non-volatile>** folder on WinCE device.

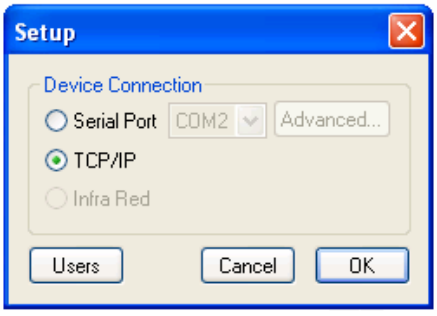
Run the **CEServer.exe** to launch the *Remote Agent* dialog on the remote workstation.



Remote Agent Dialog

Click the **Setup** button in the *Remote Agent* dialog on the run-time workstation.

When the *Setup* dialog opens, specify the device connection method (**Serial** or **TCP/IP**) for connecting to the development workstation.

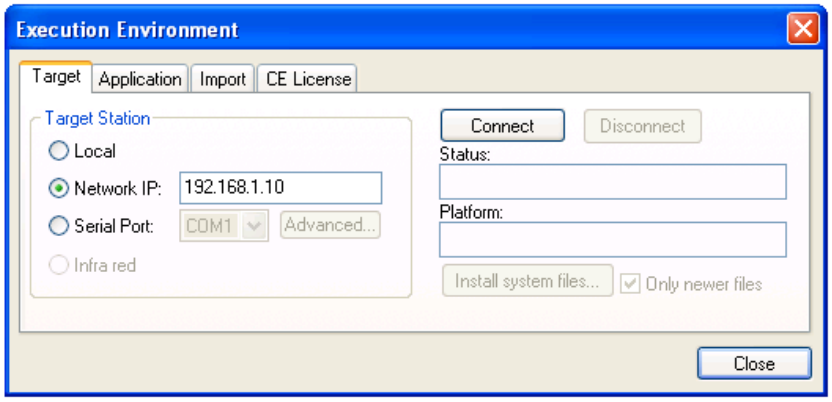


Remote Agent Setup Dialog

> **Note:**
> We recommend using TCP/IP instead of Serial Link for performance reasons.

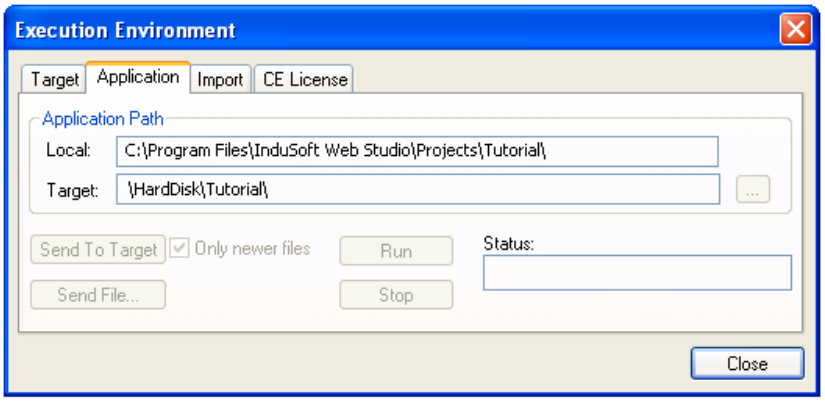
Click **OK** to close the *Setup* dialog, but leave the *Remote Agent* program running in the remote workstation.

Select **Project** → **Execution Environment** from IWS (on the development workstation).



Execution Environment Dialog

Specify a link type for the **Target Station** (**Network IP** or **Serial Port**). If you select **Network IP**, type the IP address of the remote workstation into the text box.



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.

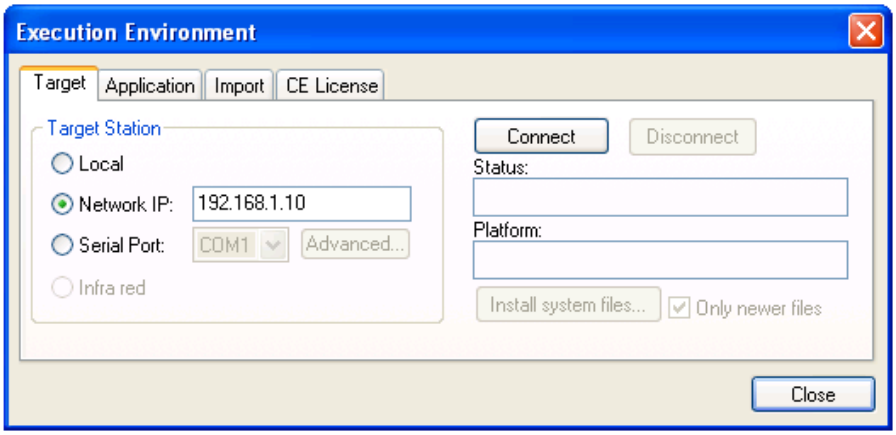
After downloading all application files, click the **Run** button to execute the application in the remote target workstation.

Downloading a CEView Application

You must use the *Project Settings* dialog to configure the **CEServer.EXE** for TCP/IP communications before running your CE unit.

In the development environment (on the development workstation), click the **Execution Enviornment** button.

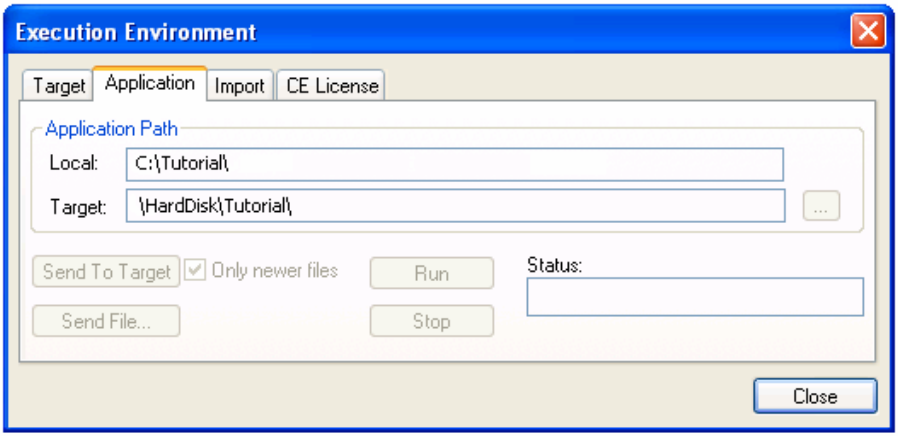
When the *Execution Environment* dialog displays, type the IP address of the CE unit, and then click the **Connect** button.



Execution Environment Dialog

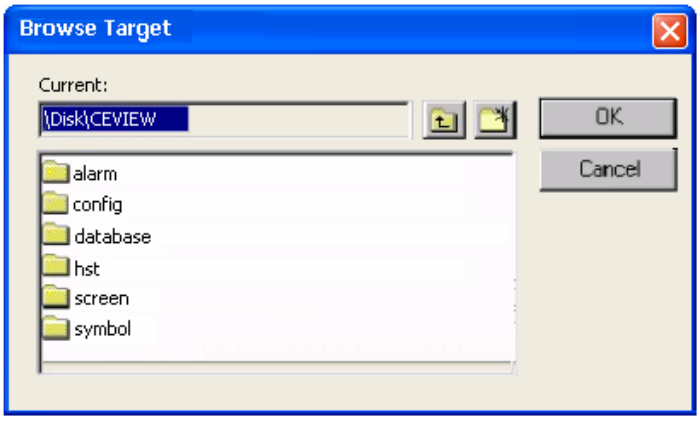
When the **Status** field display indicates that you are connected, click the **Install system files** button to install the CEView on the remote workstation.

After installing CEView, click the **Application** tab.



Click the Application Tab

Click the **Browse** button, located next to the **Target** field, when the *Browse Target* dialog displays, and choose the remote directory where you will download the application.



Select the Remote Directory

Click the **Send to Target** button to download the application. ***Note*:** The CE unit must be running **CEServer.exe**.

After downloading the application, you can click the **Run** button to start the application and click **Stop** to stop the application while connected to the CE unit.

The **Status** field displays the last operation status.

Appendix A. Database Interface

IWS supports ADO.NET to provide intuitive, simple, flexible and powerful interface with standard technologies from MCDA (Microsoft Common Database Access) 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 MCDA compatible (please see table x for 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**: The powerful grid object can query and change information in a Relational Database.
- **Web**: Because the items listed below are already available in IWS Web interface, one can deploy an application that stores/save 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	Provider
Microsoft SQL Server	ADO.NET Provider for SQL Server
Microsoft Access	ADO.NET Provider for OLE DB
Microsoft Excel	ADO.NET Provider for OLE DB
Oracle	ADO.NET Provider for Oracle
Sybase	ADO.NET Provider for Sybase
MySQL	ADO.NET Provider for OLE DB

Conformance Test Table

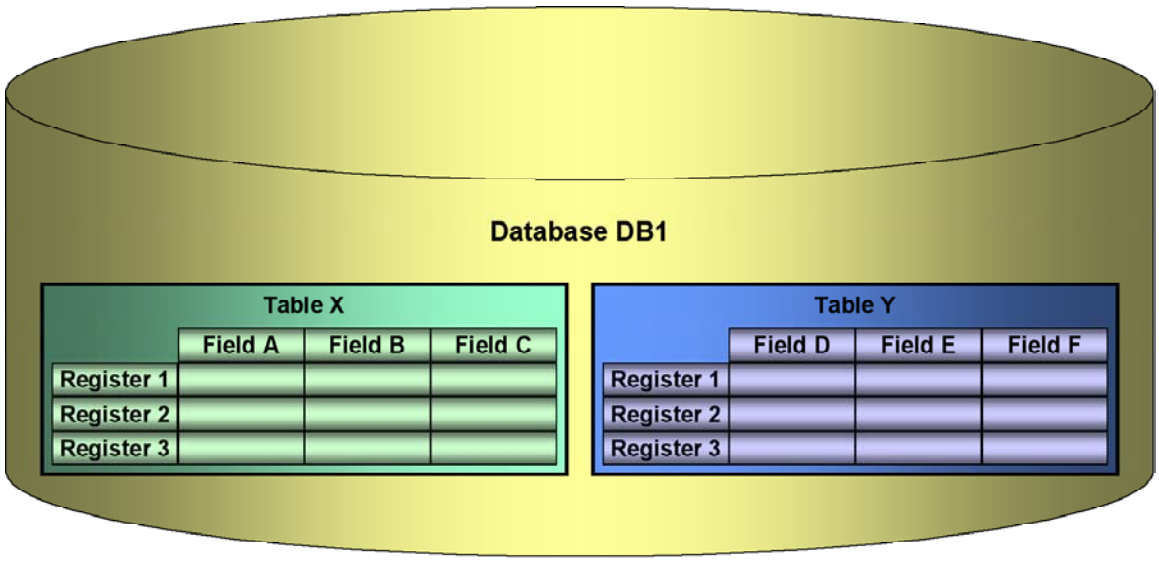
General Concepts

This section describes databases, database providers and the way IWS interfaces with different databases.

SQL Relational Databases

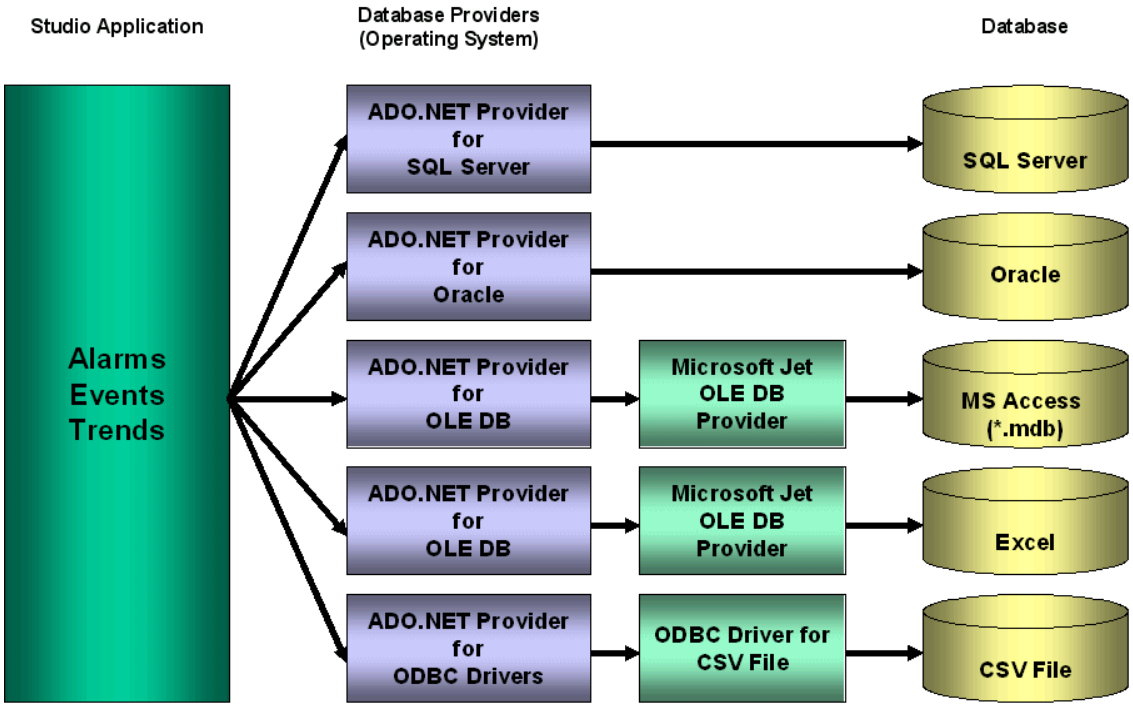
A SQL Relational Database is a set of information stored in tables with fields and registers, which support SQL commands.

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 OLE 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 *Appendix A* 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* one can get access to the database; the Microsoft Jet OLE DB provider is used to get 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.

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 Studio database gateway interface). Therefore, the application settings are mostly uniform, regardless of the specific database chosen by you, the end-user.

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**	**File Format**: Text (UNICODE). IWS uses the vertical bar character (|) to separate the fields. **Default Path**: …\<Application Path>\Alarm\ALYYMMDD.ALH , where: YY = Two last digits of the year MM = Month DD = Day.
	Database	**Database Type**: Chosen by the user **Default Table Name**: AlarmHistory
Events	**Proprietary**	**File Format**: Text (UNICODE). IWS uses the vertical bar character (|) to separate the fields. **Default Path**: …\<Application Path>\Alarm\EVYYDDMM.EVT , where: YY = Two last digits of the year MM = Month DD = Day.
	Database	**Database Type**: Chosen by the user **Default Table Name**: EventHistory
Trend	**Proprietary**	**File Format**: Binary **Default Path**: …\<Application Path>\Hst\GGYYDDMM.HST , where: GG = Trend group number (in hexadecimal format) YY = Two last digits of the year MM = Month DD = Day.
	Database	**Database Type**: Chosen by the user **Default Table Name**: TRENDGGG (GGG = Trend Worksheet Number – e.g.: TREND001 for the Trend Worksheet 001)

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.

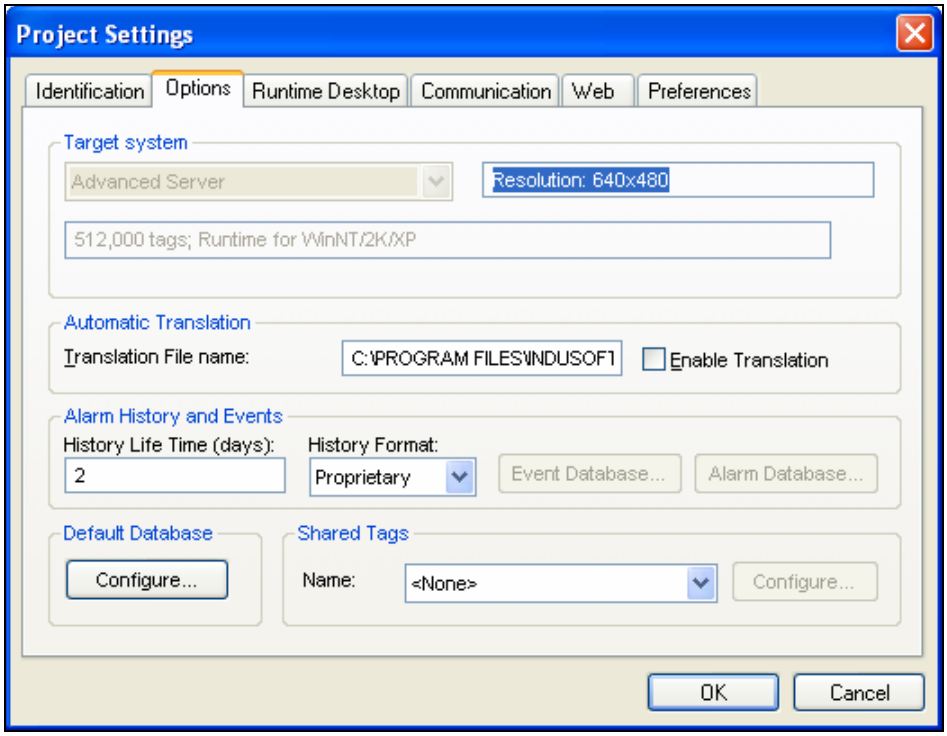
> **Note:**
> The Primary and Secondary can be different types of databases. However, they must have the same fields.

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).

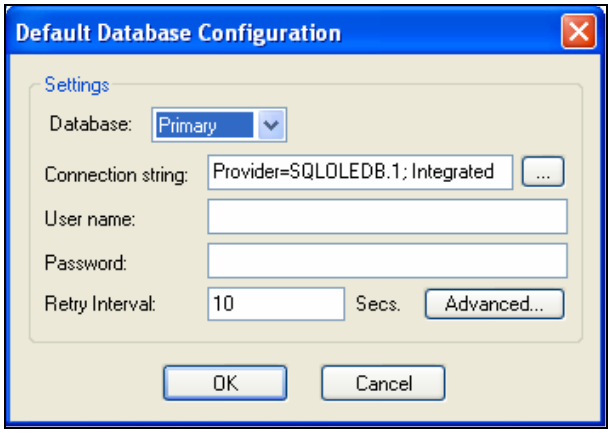
Default Database

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:



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 Studio database gateway (running in a remote computer) via TCP/IP. The Gateway implements the interface with the Database through the Database Provider available in the computer where it is running.

The Studio database gateway does not require complex configuration. Just copy the files STADOSvr.exe and StudioADO.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 topic “Studio Database Gateway” for information on how to configure the Studio Database Gateway advanced settings.

> **Tip:**
>
> The Studio 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	▪ 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	▪ 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	▪ 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.

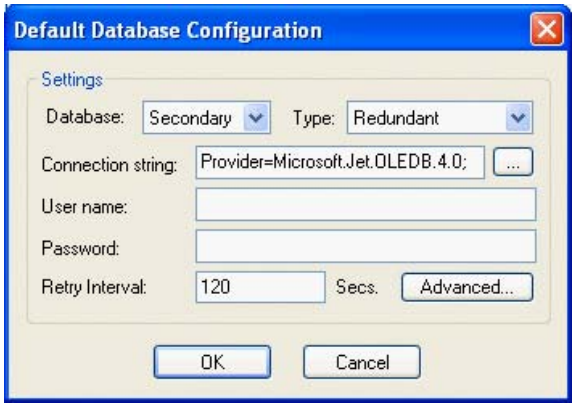
Note:

- 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.

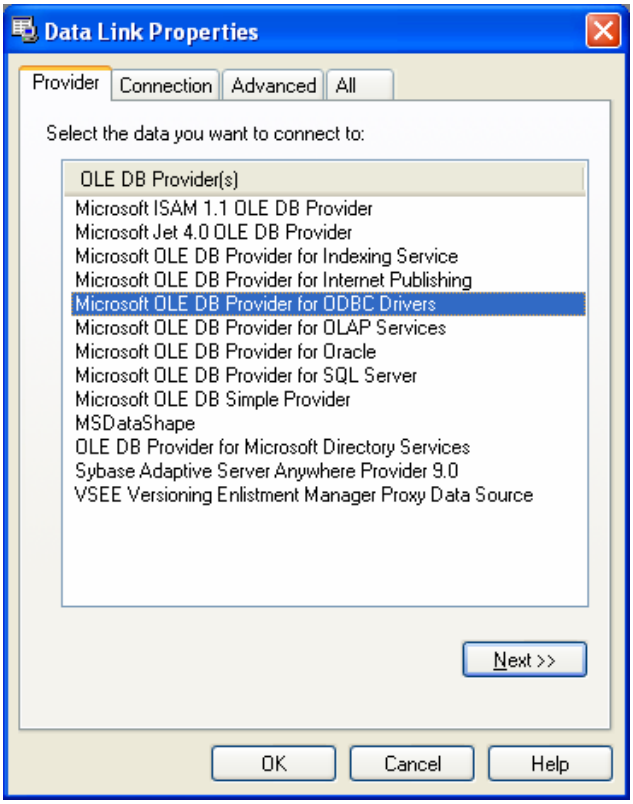
The following picture illustrates the Database Configuration dialog:



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.

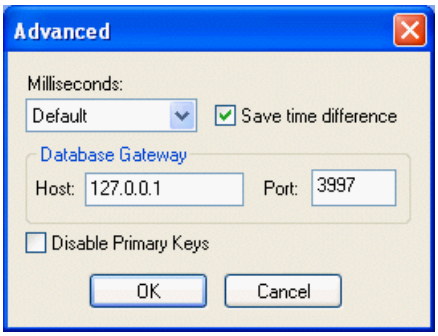
- **Connection string** field: This field defines the database where IWS will write and read values as well as the main parameters used when connecting to the database. Instead of writing the Connection string manually, you can press the browse button (…) and select the database type from the **Data 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.
    - **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 the *Studio Database Gateway* section 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 the *Studio Database Gateway* section for information about how to configure the Studio ADO Gateway.

> ⇨ **Tip:** The fields Table, Connection String, User Name, Password and Host name can have tags between curly brackets if you need dynamic values.

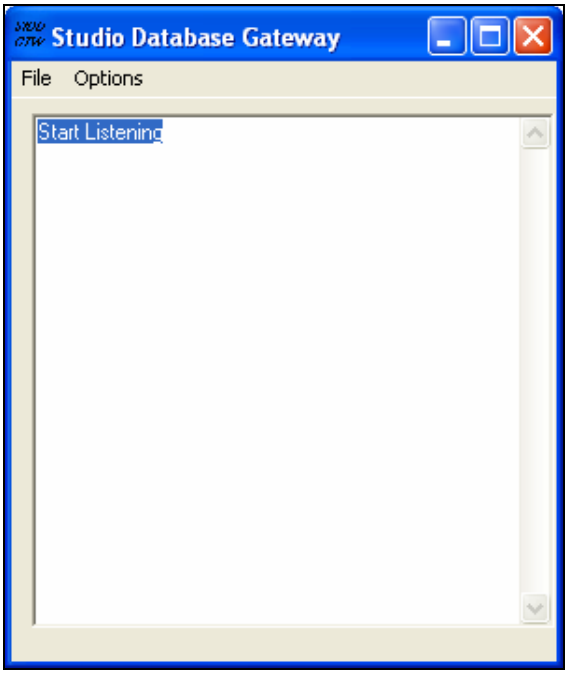
Studio Database Gateway

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) 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:

By right clicking 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 **IWS LogWin** window. The messages are reported by exceptions generated by the ADO.NET Provider. (For troubleshooting answers, please consult the ADO.NET provider documentation.)

The Studio Database Gateway has Advanced Settings that are configured in the StudioADO.ini file. If you are having problems interfacing with a specific database, you will probably need to change some of these settings or add new providers to the file. The following parameters are available:

Parameter	Range of Values	Description
SaveMSec	1 - Disable 2 - Enable 3 - Separate Column	This setting specify the default behavior for the provider when saving milliseconds. The default can be changed on the Advanced Settings in the Database Configuration Dialogs.
Assembly	Any string that contains a .Net Framework assembly	Assembly option for all providers. The assembly has all the classes required to interface with the database. Most of the providers are inside the System.Data assembly.
ConnectionClass	Any connection class inside the assembly	The Connection Class is the one that implements the System.Data.IDbConnection interface.
AdapterClass	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 define 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
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 (?)
ValueAddNumber	0 or 1	Indicates whether a sequencial 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.

Parameter	Range of Values	Description
BoolType	Any string representing a valid datatype for the database	When trying to create columns to store boolean values, the datatype specified on this parameter will be used. You need to make sure that the datatype specified is able to save boolean values.
IntegerType	Any string representing a valid datatype for the database	When trying to create columns to store integer values, the datatype specified on this parameter will be used. You need to make sure that the datatype specified here is able to store 32 bit values.
RealType	Any string representing a valid datatype for the database	When trying to create columns to store real values, the datatype specified on this parameter will be used. You need to make sure that the datatype specified here is able to store 64 real values.
StringType	Any string representing a valid datatype for the database	When trying to create columns to store string values, the datatype specified on this parameter will be used. You need to make sure that the datatype specified is able to save the number of characters that you are willing to save on your application.

A single section called [Providers] has all the parameters inside it. 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
DefaultColumnDelimiterPrefix=[
DefaultColumnDelimiterSuffix=]
DefaultValueString=Value
DefaultValueAddNumeber=1
DefaultTypeBool=INTEGER
DefaultTypeInteger=INTEGER
DefaultTypeReal=REAL
DefaultTypeString=VARCHAR(255)
DefaultTypeTimeStamp=DATETIME
```

The next item on the file lists the amount of providers:

```
Count=5
```

The providers are identified by the “Provider” parameter followed by a number. When connecting to a database, the Provider parameter in the connection string is compared to the provider’s identification, in order to determine which provider will be used. If there is no provider with the value on the connection string, all the default values are assumed. Besides its identification, each provider can have its own value per each parameter. Again, if no value is specified, the default is used. Below is an example with five providers:

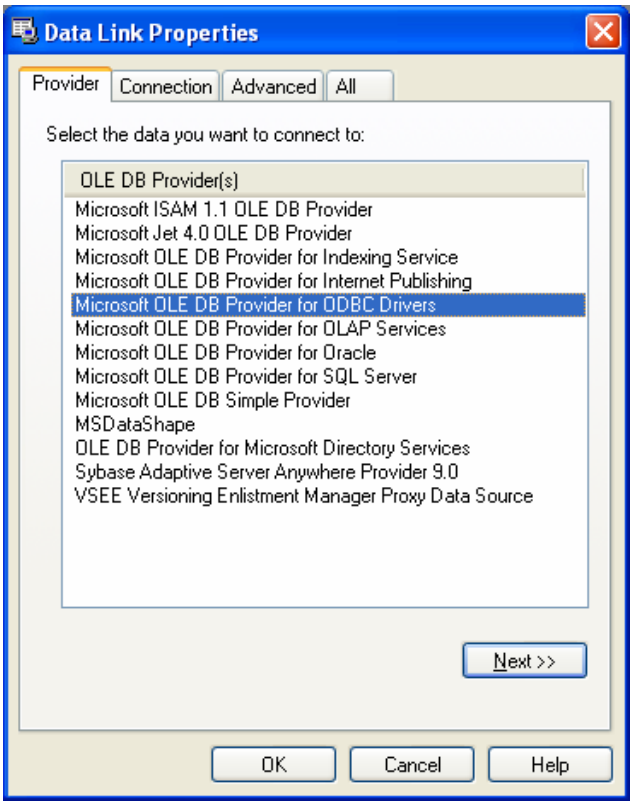
```
Provider1=MICROSOFT.JET.OLEDB
SaveMSec1=3

Provider2=SQLOLEDB
ConnectionClass2=System.Data.SqlClient.SqlConnection
DataAdapterClass2=System.Data.SqlClient.SqlDataAdapter
CommandBuilderClass2=System.Data.SqlClient.SqlCommandBuilder

Provider3=MSDASQL
ConnectionClass3=System.Data.Odbc.OdbcConnection
DataAdapterClass3=System.Data.Odbc.OdbcDataAdapter
CommandBuilderClass3=System.Data.Odbc.OdbcCommandBuilder

Provider4=SQLOLEDB
Assembly4=System.Data.OracleClient.OracleClient
ConnectionClass4=System.Data.OracleClient.OracleConnection
DataAdapterClass4=System.Data.OracleClient.OracleDataAdapter
CommandBuilderClass4=System.Data.OracleClient.OracleCommandBuilder

Provider5=ASAPROV
Assembly5=iAnywhere.Data.AsaClient
ConnectionClass5=iAnywhere.Data.AsaClient.AsaConnection
DataAdapterClass5=iAnywhere.Data.AsaClient.AsaDataAdapter
CommandBuilderClass5=iAnywhere.Data.AsaClient.AsaCommandBuilder
```

Using ODBC Databases

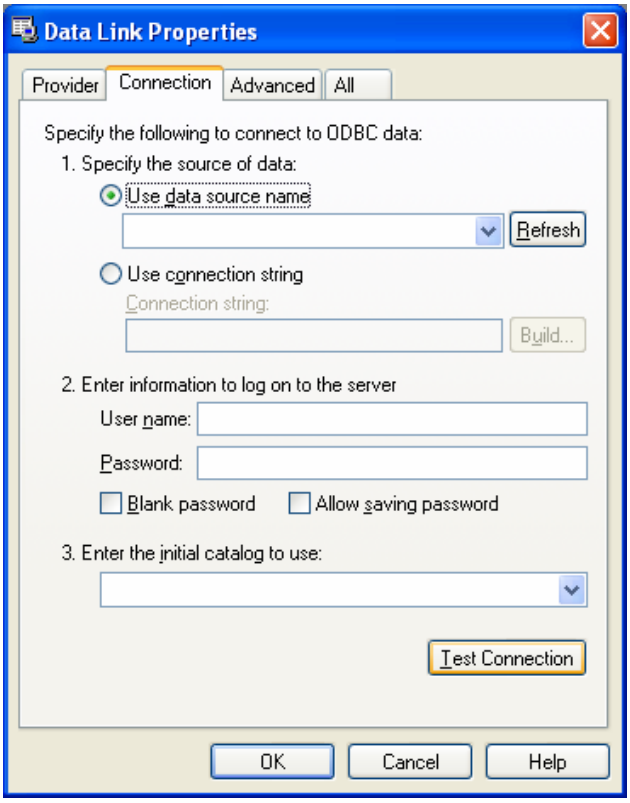
Almost every database provides an ODBC interface that can be used to interface with it. The database features provided by IWS can be used with ODBC drivers through the ADO.NET interface for ODBC. In order to use this capability, you must use Microsoft *.NET Framework 1.1* or higher.

> **Note:**
> Microsoft *.NET Framework 1.1* is automatically installed 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.

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
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.
Tag	String	Tag Name
Message	String	Alarm message
Ack	Boolean	0: Indicates the alarm was acknowledged or does not require acknowledgment 1: Indicates the alarm was not acknowledged
Active	Boolean	0: Indicates the alarm is not active 1: Indicates the alarm is active
Value	Real	Tag value when the alarm event occurred
Group	Integer	Alarm Group Number
Priority	Integer	Alarm Priority Number
Selection	String	Alarm Selection value
Type	Integer	1: HiHi 2: Hi(On) 4: Lo(Off) 8: LoLo 16: Rate(Change) 32: Deviation+ 64: Deviation-
Ack_Req	Boolean	0: Requires acknowledgement (Ack) 1: Does not require acknowledgement
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.
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.

Field Name	Data Type	Remarks
User	String	User logged when the alarm event occurred. This field only exists for Version >=1
Comment	String	Comment (optional) typed by the operator when the alarm was acknowledged. This field only exists for Version >=1
User_Full	String	Full name of the user logged when the alarm event occurred. This field only exists for Version >=2
Station	String	Name of the station (computer) where the alarm event occurred. This field only exists for Version >=2
Previous_Value	Real	Tag value that occurred before the alarm event. This field only exists for Version >=3
Deleted	Boolean	0: Alarm message was not deleted 1: Alarm 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 >=3
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 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
TimeStamp	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.