MET-TP 4th Course

A Guide to Indusoft SCADA

"Knowledge is Power"

Web-Ready InduSoft® Tools for Automation

ALMAWARED ENGINEERING & TRADING SAE

MET-TP 4th Course

A Guide to Indusoft SCADA

"Knowledge is Power"

ALMAWARED
ENGINEERING
& TRADING SAE

Table of Contents

1. Introduction

Introduction

Welcome to the fourth course in the **MET-TP** series. Al-**M**awared **E**ngineering and **T**rading **T**raining **P**rogram offers a training service so that you can plan over the time the growth of the device knowledge, from the frequency inverters to the soft starters for asynchronous motors up to the PLC "Programmable Logic Controller"; Touch screens and SCADA system. Courses are targeted to engineers, technicians, users and installers and to the service personal as well. MET suggests courses that are mainly oriented to the use of the drives and of the automation system.

Our highly trained engineers will guide you step by step through each training course, allowing you to perform each step by yourself from small examples to large applications to help you practice everything you learn during the training course.

This course covers most of the important topics you need to know if you are planning to use **Indusoft Webs Studio (IWS)** for **SCADA** applications**.** Upon completion of this course you will be able to:

- Create a new SCADA application from scratch
- Implement new screens with different objects and functions needed in your application such as (Switches, Lamps, Sliders, images...etc)
- Design a fully functional security system using high security logging options and alarms pages.
- View any time varying data using trend graphs.
- Add drivers to your application and learn how to write I/O memories inside them.
- Learn more about communication types and parameters.

NOTE:

This training guide was acquired from Indusoft. MET is just using the same guide as a professional training material.

2. Company Overview

Experience and high responsiveness

Year 2001 is not only a date for ALMAWARED ENGINEERING AND TRADING S.A.E (MET). It represents the starting of a company that is today specialized in the industrial automation, mechanical and electrical power transmission field, thanks to the entrepreneurial capabilities from foundation members Mr. Abdel Aziz Aboul Atta, Ms. Bahia Khairy, Eng. Mohamed Abdel Aziz, Eng. Khalid Abdel Aziz and Eng. Khalid Ateya, expertise and firm commitment of the promoting partners.

MET partners achieved the quality certificates, with the aim to grant a good quality system for the different market needs to satisfy most of their customers' needs by providing complete solutions or individual tailor-made solutions. Only a long experience allows a company to reach in a flexible and wide way to market demands, with a complete range of products and services.

Flexibility and rapidity of product range

Our structure is composed of real problem solvers who study the customer requirements and orient him towards simple and innovative solutions thanks to the structure of MET product range that gives our highly professional technical engineers a wide area of solutions. MET product range comprises a whole variety of automation, electrical and mechanical equipment such as PLCs, touch screens, SCADA systems, flow meters, density meters, BMS components, inverters, soft starters, AC/DC motors, Servo systems, gearboxes, clutches and brakes. Distributed Control System (DCS) is the latest product that MET provides to our customers.

A strong presence in the market

Our experienced sales and marketing team is highly technical and customer-oriented. They combine premium customer support with years of industry experience to develop and sustain long-term relationships.

Our team ensures each customer is treated with professionalism and integrity as we work hard to understand our customers' needs and to provide solutions to meet those needs, so we serve a wide variety of market fields such as soup and fats, sugar industries, petroleum and refineries, cement factories, iron and steel mills, pharmaceutics and cosmetics, textiles and dyeing, paints and chemicals, plastics and petrochemicals, food & beverages industries, paper and printing, packing and wrapping, pump & water treatment plants, sewage & water treatment plants and commercial HVAC.

Introduction to InduSoft

Web Studio Training Course

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. You will develop HMI and SCADA applications on a Windows NT/2000/XP operating system and run these applications on a Windows NT/2000/XP or Windows CE run-time workstation.

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**: Read the following publications for detailed information about 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. This publication is available in the *Documentation* folder on the IWS CD-ROM or from the Help menu on the main menu bar.
  - *InduSoft Web Studio Getting Started Guide*: Designed for first-time users, this publication contains information about the basic functions of InduSoft Web Studio. This publication is provided in the *Documentation* folder on the IWS CD-ROM or from the Help menu located on the main menu bar.
  - *Drivers User Guides*: Explain how to configure individual InduSoft drivers, according to their unique protocol characteristics. One customized user guide is included with each InduSoft driver. These publications are provided in the DRV subdirectory of the InduSoft Web Studio folder on the IWS CD-ROM or from the Help menu located on the main menu bar.
- **Customer/Technical Support**: Contact your InduSoft Technical Support Representative by email at info@indusoft.com or by telephone at 877-INDUSOF (877-463-8763).
- **InduSoft Web Site**: Visit *www.InduSoft.com*.

Revision History

Rev	Author	Date	Comments
A	Fabio Terezinho	May 09, 2002	Initial revision
B	Fabio Terezinho	June 17, 2002	Overall changes in the layout and content of the document
C	K. C. Francis	September 8,2002	Overall changes in the layout and content of the document
D	K. C. Francis	November 27, 2002	Minor changes to the content of the document
E	Suzanne Warnke	August 1, 2003	Overall changes in the layout and content of the document
F	Fabio Terezinho	April 20, 2004	Minor change to two screenshots
G	Julie Ward	August 31, 2004	Service Pack 3 enhancements and fixes, including updated screenshots
H	Julie Ward	May 31, 2005	Service Pack 4 & 5 modifications
I	Michael Hayden	March 7, 2006	Version 6.1 modifications, including updated screenshots; added content for Trend Control object and VBScript; deleted redundant / obsolete content; edited for language and usability.

Chapter 2. Installing and Starting InduSoft Web Studio

You can install InduSoft Web Studio (IWS) from the IWS CD-ROM.

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

> **Note:**
> For Windows CE applications, you use IWS to download CEView (the run-time software) to the Windows CE HMI using a serial or TCP/IP link.
>
> When you install InduSoft Web Studio on Windows NT/2000/XP computers, IWS stores the CEView run-time files in the following folder:
>
> ***<InduSoft Web Studio Folder>\Redist\CEView\<Processor Type>***
>
> Where:
>
> - ***<InduSoft Web Studio Folder>*** is the installation directory chosen during the installation (**C:\Program Files\InduSoft Web Studio** is the default installation directory).
> - ***<Processor Type>*** is the processor platform. InduSoft provides a CEView run-time for all processor platforms supported by the WinCE operating system (Arm, Mips, MipsFP, Pocket-Arm, Pocket-Mips, Pocket-SH3, PPC, SH3, SH4, Thumb, and x86).

Before You Begin

Before installing IWS, you must:

- Have Administrator privileges for the Windows NT/2000/XP workstation on which you are installing InduSoft Web Studio.
- Uninstall any older versions of IWS (or install the newer version to a different directory). Also, you cannot install the same version of IWS in two different paths on the same computer.
- Install the following hardware and software on your machine:
  - IBM-compatible computer with an Intel® Pentium IV-compatible processor or higher
  - Windows 2000/XP/2003 server operating system
  - Minimum of 256MB random-access memory (RAM) (512GB or higher recommended)
  - MS Internet Explorer 6.0 or higher
  - Minimum of 500MB free hard disk space to install the product and the application (the history files/databases will demand additional disk space)
  - Ethernet adapter
  - 100% IBM-compatible VGA or SVGA display adapter with 64MB Video RAM (VRAM) or higher
  - Microsoft-compatible pointing device (such as a mouse, trackball or touch-screen)
  - Standard keyboard with function keys F1 through F12
  - CD-ROM drive (optional)
  - 3.5-inch floppy drive (optional)
  - Parallel printer port (optional)
  - USB port (optional)
  - Serial COM ports and adapters (optional)

Notes:

- The requirements described above are based on typical applications. Depending on your specific application, the minimum requirements may vary.
- Applications developed with InduSoft Web Studio can run under devices powered with the Windows CE operating system (Windows CE v3.0 or Windows CE .NET) such as industrial HMIs or PDAs (PocketPC). Consult your vendor for the hardware requirements when running your application under the Windows CE operating system.
- In addition to the operating systems described above, the Web Thin Client station can be running the Windows 98 or Windows ME operating systems.
- When using a Hardkey instead of a Softkey to license the product, either the parallel or the USB port must be available in the local computer.
- Some of the items listed above as optional may be mandatory depending on your application. For instance, if you need to exchange data with a PLC via a serial interface, the computer must provide a serial COM port.

We are now ready to install and start InduSoft Web Studio—the procedure begins on the next page.

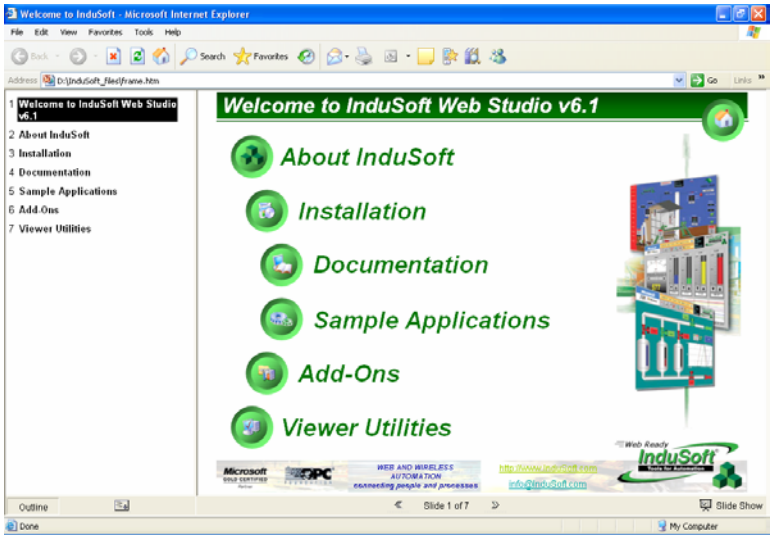
Exercise: Installing IWS

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

☑ Turn on the power to your development computer and be sure that no other programs are running.

☑ Insert the installation CD-ROM into the computer's CD-ROM drive.

A CDBrowser window should display automatically:



CDBrowser Window

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

The Browser window contains the following folders:

- **Read First**: Contains a document (in **.pdf** format) with important information you should read before using the current product.
- **Viewer Utilities**: Contains *Microsoft PowerPoint® Viewer* (needed to view the InduSoft presentations provided in **.ppt** format) and *Adobe® Acrobat Reader™* (needed to view the InduSoft documents provided in **.pdf** format).
- **About InduSoft**: Contains a short PowerPoint presentation about InduSoft.
- **Installation**: Contains an InduSoft Web Studio icon. Double-clicking this icon starts the installation program.
- **Documentation**: Contains all IWS documentation in **.pdf** format.
- **Sample Applications**: Contains sample applications to help you develop your own applications using InduSoft Web Studio.
- **Add-Ons**: Contains a demo version of the *Symbol Factory ActiveX* program—an extensive symbol library that simplifies application development.

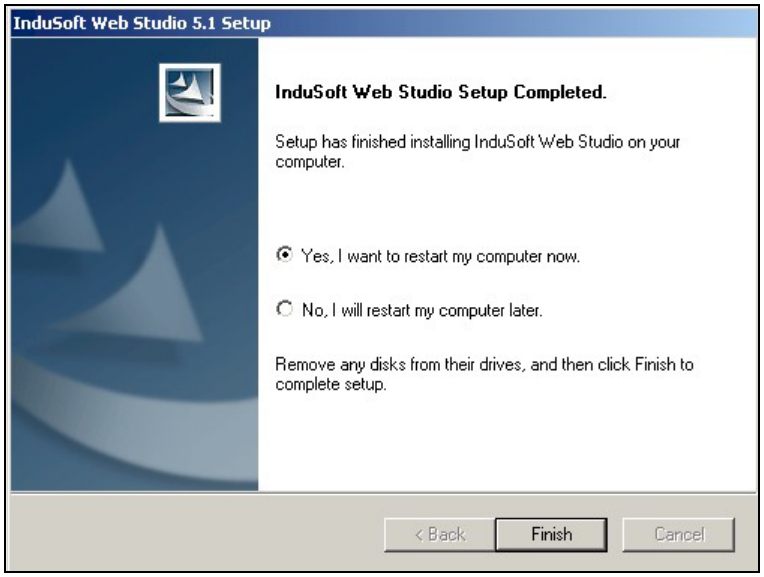
☑ In the CDBrowser window, double-click the *Installation* folder and then double-click the **InduSoft Web Studio** icon to start the *InduSoft Web Studio Installation Wizard*.

A *Setup* dialog displays to inform you that the *Wizard* is loading.

☑ Follow the instructions provided by the *Wizard* to proceed with the installation, which includes:

- Reading and accepting the License Agreement
- Entering a user name and your company name
- Choosing a destination location (*accept the default*)
- Selecting the components to install (*accept the default*)

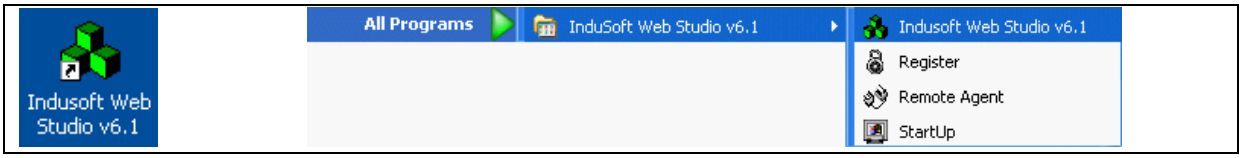
A *Setup Status* dialog displays while the program installs, and the *Setup Complete* dialog displays when the installation is finished:



Setup Complete Dialog

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

☑ After restarting your computer, double-click the **InduSoft Web Studio** icon on the desktop or select **Start → Programs → InduSoft Web Studio → InduSoft Web Studio** to start the IWS program.



From the Desktop **From the Start Menu**

Starting InduSoft Web Studio

➪ **Tip:**
You can run the development environment under any video setting; however, to run applications on a CE platform, we recommend configuring your *Display* video settings to a resolution of 800x600 (or higher) and using 256 colors (or higher) for a more pleasing environment. Application resolution (screen size) is independent of the operating system resolution.

> **Note:**
> Microsoft *.NET Framework 1.1* is automatically installed with IWS, since the database features provided by IWS can be used with ODBC drivers through the ADO.NET interface for ODBC.

Chapter 3. What Is InduSoft Web Studio?

This chapter provides a high-level overview of the InduSoft Web Studio (IWS) product, including the following information:

- Product Overview
- IWS Development Environment
- IWS Internal Structure
- Tags Database
- Development Modules

Product Overview

InduSoft Web Studio 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:

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

InduSoft Web Studio's process automation applications run on microcomputers connected in real-time to machines or processes through programmable controllers, remote I/O devices, or other data acquisition equipment.

IWS applications consist of animated operator-interface screens, drivers (configurable for PLCs or other I/O devices to be controlled), an application tags database, and optional modules such as alarm monitors, logic, trend charts, recipes, schedulers, and a security system. IWS applications interface with industrial I/O systems and other Windows applications in the run-time environment using ODBC, DDE, NetDDE, OPC, or TCP/IP protocols.

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

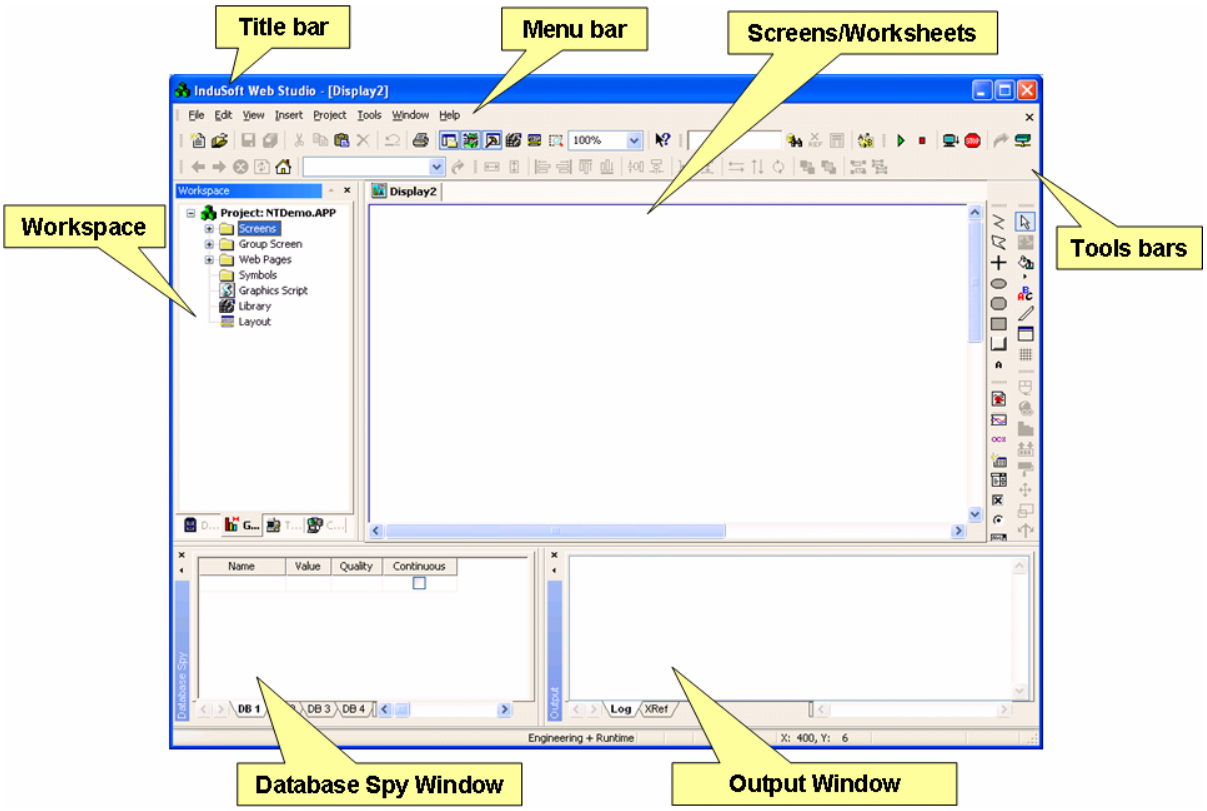
The InduSoft Web Studio product consists of two parts:

- The *development system* software runs on a desktop, laptop, or industrial PC running Windows NT/2000/XP.
- The *run-time system* software runs on an operator interface workstation running Windows NT/2000/XP or Windows CE.

> **Note:**
> The run-time system software (CEView) for the Windows CE operating system is usually pre-loaded on the HMI. If necessary, you can update the CEView version of the development system software by downloading the current version to the HMI.

IWS Development Environment

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



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 button**: Click to minimize the window.
- **Resize/Maximize button**: Click to toggle between the following two options:
  - **Resize** tiles the window
  - **Maximize** maximizes the window to fill your computer screen
- **Exit** (or **Close**) button: Click to automatically save the database, and then close Studio. 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 the development system does not close the run-time system.

The Status Bar

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

Example Status Bar

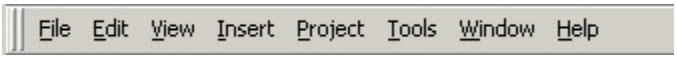
The fields are as follows (from left to right):

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

> **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.

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.

> **Notes:**
> The menu bar is dockable.

Toolbars

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

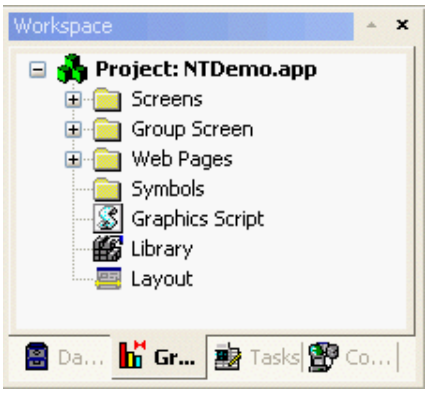
> **Note:**
> All toolbars are dockable screen objects. You can move a toolbar to a different screen location by clicking on its title bar and dragging it to a new location.

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

> ⇨ **Tip:**
> The status bar (located at the bottom of the InduSoft Web Studio interface) provides a brief description of the selected button.

Workspace

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



IWS Workspace

The *Workspace* window is divided into four tabs:

- The **Database** tab enables you to access any available tags from the application and security system components. This tab includes the following folders:
  - *Application Tags*
  - *Classes*
  - *Shared Database*
  - *Internal Tags*
  - *Security*
  - *Event Settings*
  - *Global Procedures*
- The **Graphics** tab enables you to access all screens and symbols in the application. This tab includes the following folders and icons:
  - *Screens*
  - *Group Screen*
  - *Web Pages*
  - *Symbols*
  - *Graphics Script*
  - *Library*
  - *Layout*

- The **Tasks** tab enables you to access all task worksheets in the application. This tab includes the following folders:
  - *Alarms*
  - *Trend*
  - *Recipes*
  - *Reports*
  - *ODBC*
  - *Math*
  - *Script*
  - *Scheduler*
- The **Comm** tab enables you to access all worksheets configured to establish communication with another device or software using available protocols. This tab includes the following folders:
  - Drivers
  - OPC
  - TCP/IP
  - DDE

You can right-click on all folders and components, which opens a menu for each. The rest of this section describes each tab, its folders, and component icons.

Screen/Worksheet Editor

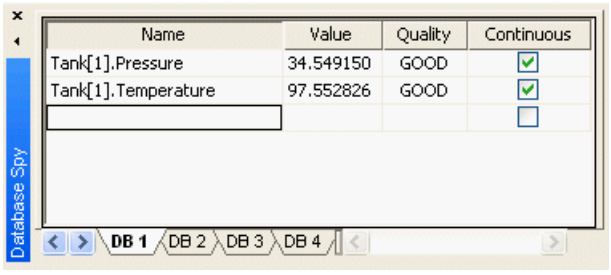
Use the powerful, object-oriented screen editor to create and edit a variety of screens and worksheets for your applications. You can input information using your mouse and keyboard, output control data to your processes, and automatically update screens based on data input from your processes.

Other screen editor features include:

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

Database Spy Window

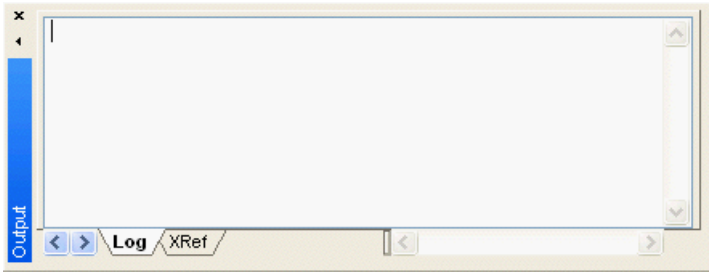
Use the *Database Spy* window as a debugging tool. You can monitor and force database tags and execute functions from this window.



Database Spy Window

Output Window (LogWin)

Use the *Output* or *LogWin* window to view debugging messages provided by IWS.



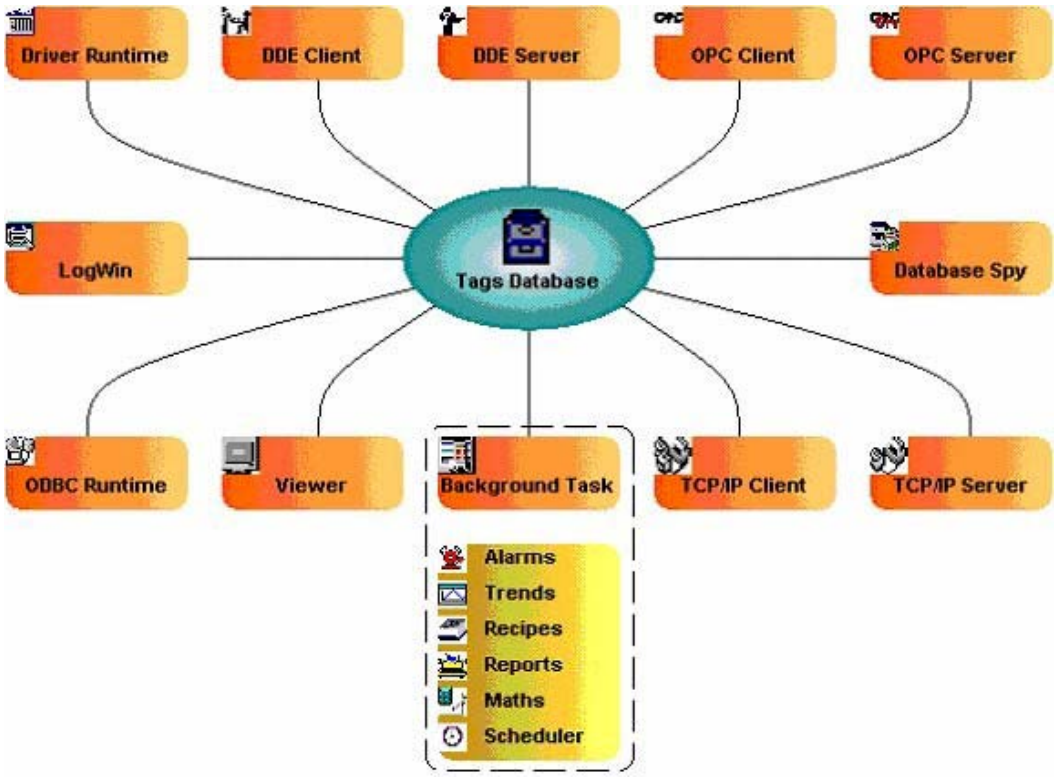
Output Window

Symbols Library

Symbols are reusable objects (or groups of objects) that you can store for reuse. IWS provides an extensive symbol library that enables you to add, edit, and reuse symbols quickly and easily. Click the **Open Library** ( ) button (in the *Workspace* or on the *Standard* toolbar) or select the **Library** option from the *View* menu.

IWS Internal Structure

This section describes the internal structure of the InduSoft Web Studio product. The following figure illustrates the basic architecture:



IWS Architecture

Tags Database

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

Configuring an application consists of defining which tags are 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.

Development Modules

This section discusses graphics and tasks development modules.

Graphics

The most basic function performed by InduSoft Web Studio is to provide a window into the process. The ability to display the status of the process by interacting with instrumentation or computers, is described as the Human-Machine Interface (*HMI*).

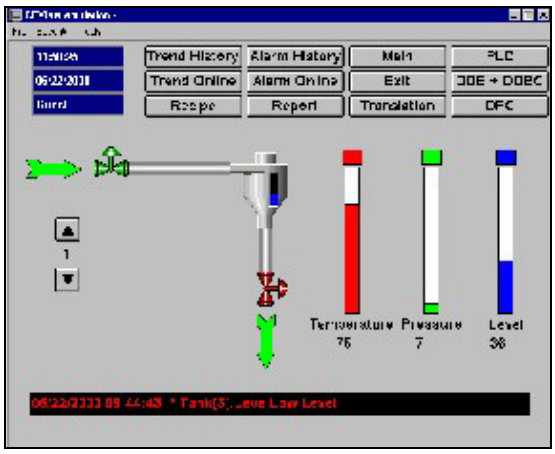
InduSoft Web Studio allows you to create applications that can monitor processes using high-resolution color screens.

The InduSoft Web Studio graphic tools consist of two modules:

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

You can use *animation links* to create dynamic graphic objects or symbols. Animation links cause objects and symbols to change appearance to reflect changes in the value of a tag or an expression. Each screen is an association of static and dynamic objects.

Screens can have an optional bitmap that acts as a background in the object window. On the following screen for example, the static images can be part of a bitmap in the background object and objects with animation in the dynamic object layer can reflect the changes in the plant, giving the illusion that the screen is three-dimensional.



Sample CEView Emulation Screen

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

Tasks

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

You can configure the following task-specific worksheets:

- **Alarm Groups** : Use to define an alarm group, its characteristics, and its messages that are reported in alarm conditions. The main purpose of the alarms is to inform the operators about any problem or change of state during the process so that corrective action can be taken.

  To show alarm messages on the screen, you must create the alarm object on the screen.

- **Trend Groups** : Use to define trend groups, which keep track of process variables behavior. You can store samples in a history file and show both history and online samples in a screen trend graph.
- **Recipes** : Use to read and write ASCII files from and to the hard disk, and to transfer values between files and real-time memory. Typically, you use this module to store process recipes, but these files can store any type of information such as operation logs, passwords, and so forth. You also can use this module to store data in XML format.
- **Reports** : Use to configure your own reports with system data, in either ASCII or RTF format. The main purpose of this module is to make report creation easier and more efficient.
- **Math Worksheets** : Use to implement additional routines to work with the basic functions of the InduSoft Web Studio modules. A *Math* worksheet is a group of programming lines that are executed as one of the application **Background Tasks**. You can configure the mathematics in blocks in different worksheets.

  This worksheet provides a free environment for logical routines and mathematical calculations that the project may need. For these purposes, the built-in IWS scripting language is simple and easy to use.

- **Script Groups** : *Script* groups serve the same purpose as *Math* worksheets, but they are written using Visual Basic Script (VBScript) rather than the built-in IWS scripting language.
- **Scheduler** : Use to generate the time bases used in an application. The Scheduler is capable of triggering events.
- **ODBC Configuration** : Use to enable InduSoft Web Studio applications to access any database that is compatible with the ODBC protocol; such as Access, Excel, Oracle, SQL Server and so on.

> **Note:**
> The ODBC interface is not available for WinCE applications.

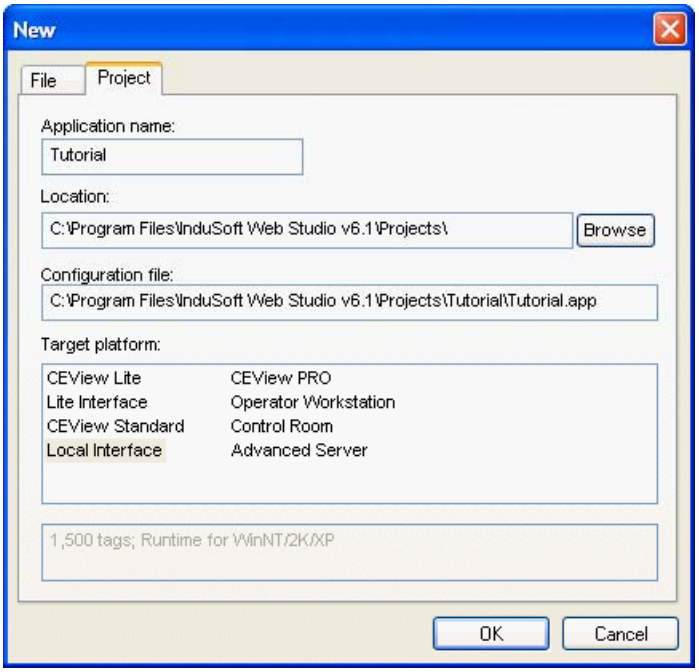
Chapter 4. Creating Projects

Next, you will create a new project and configure the necessary project settings and status.

Exercise: Creating Your Project

Use the following steps to create a new project:

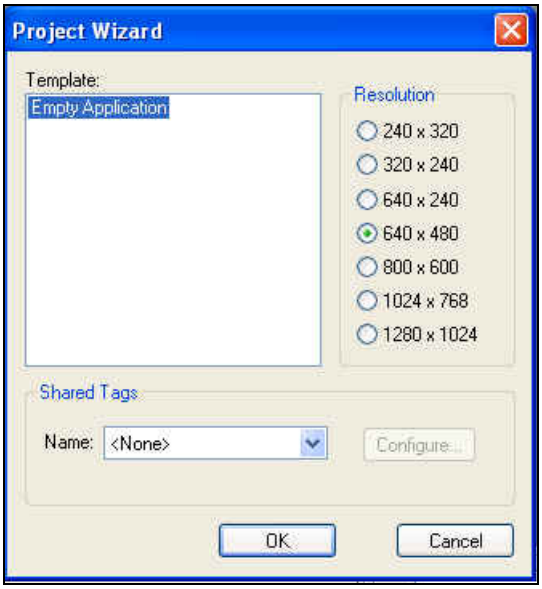
- ☑ From the development environment main menu bar, select **File** → **New** (or click the **New** button).
- ☑ When the *New* dialog opens, click the **Project** tab.



New Dialog – Project Tab

- ☑ Give your project a name by typing **Tutorial** into the **Application name** field.
- ☑ Verify that **Local Interface** (default platform) is highlighted in the *Target platform* pane, and then click the **OK** button.

☑ When the *Project Wizard* dialog displays, select the **Empty Application** option in the *Template* pane and click the **640 x 480** radio button in the *Resolution* group box. When you are done, click **OK**.

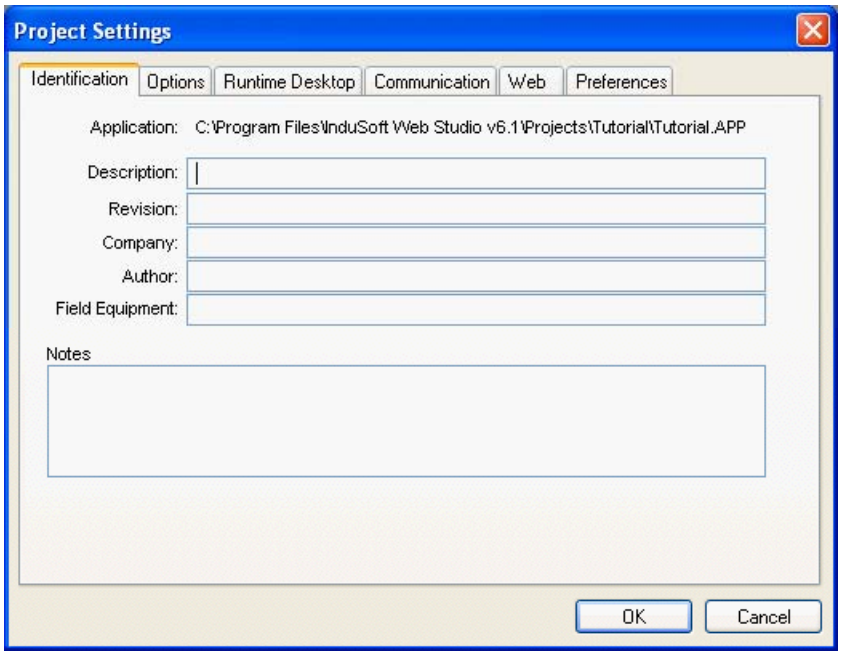


Project Wizard Dialog

Configuring Your Project Settings

Use the following steps to configure the project settings for your *Tutorial* application:

☑ From the main menu bar, select **Project → Settings** to open the *Project Settings* dialog.



Project Settings Dialog – Identification Tab

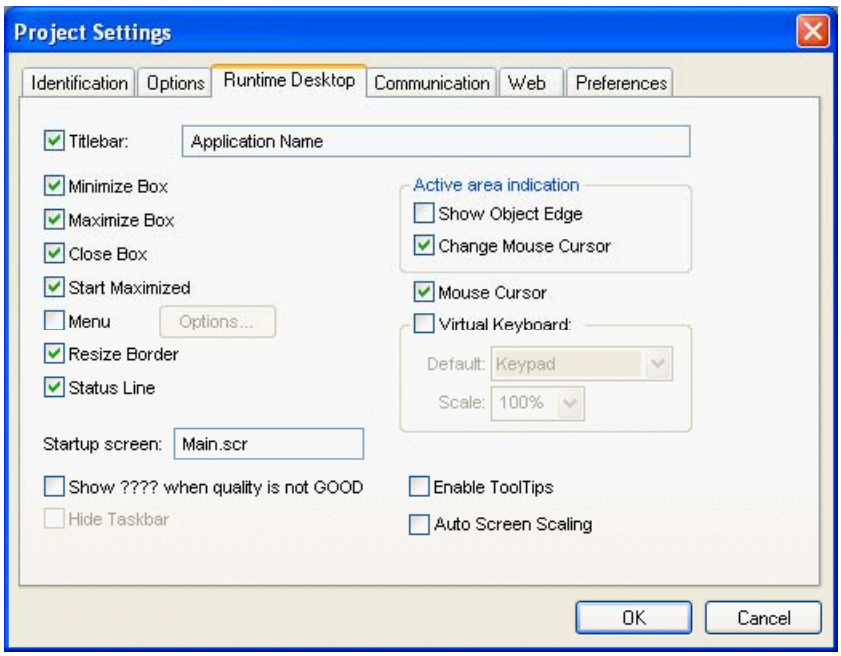
This dialog contains the following tabs.

- **Identification** tab: Use this tab to document information about your project. All of the fields on this tab are *optional*.
- **Options** tab: Use this tab to set parameters for language translation, target systems, PC-based controls, driver write-command buffering, and general information about the application.
- **Runtime Desktop** tab: Use this tab to specify parameters that determine how your application will run on a run-time workstation and which menu options will be available.
- **Communication** tab: Use this tab to specify communication parameters relating to the application in general.
- **Web** tab: Use this tab to specify parameters for remote thin clients that must access the application using a Web browser (such as *Internet Explorer*).

> **Note:**
> IWS provides a screen-grouping feature that allows you to open a linked set of screens all at once. This feature is not supported for CEView applications or for screens that will be exported to HTML format.

- **Preferences** tab: Use this tab to specify whether you want warning messages displayed before downloading applications to a target system.

☑ You always must specify a screen that will be the first screen to open when you run an application in emulation mode or on a run-time workstation. For this tutorial, click on the type **Main.scr** in the **Startup screen** field. (The file extension is *optional*.)



Project Settings Dialog – Runtime Desktop Tab

Notes:

- IWS provides a screen grouping feature that allows you to open a linked set of screens all at once. This feature is not supported for Local Interface applications or for screens that should be exported to HTML format.
- You can configure or change any of the parameters in the *Project Settings* dialog at any time during application development, but we recommend configuring these parameters at the beginning of your project development.

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

☑ You do not have to set any other *Project Settings* parameters at this time, click **OK** to exit the *Project Settings* dialog.

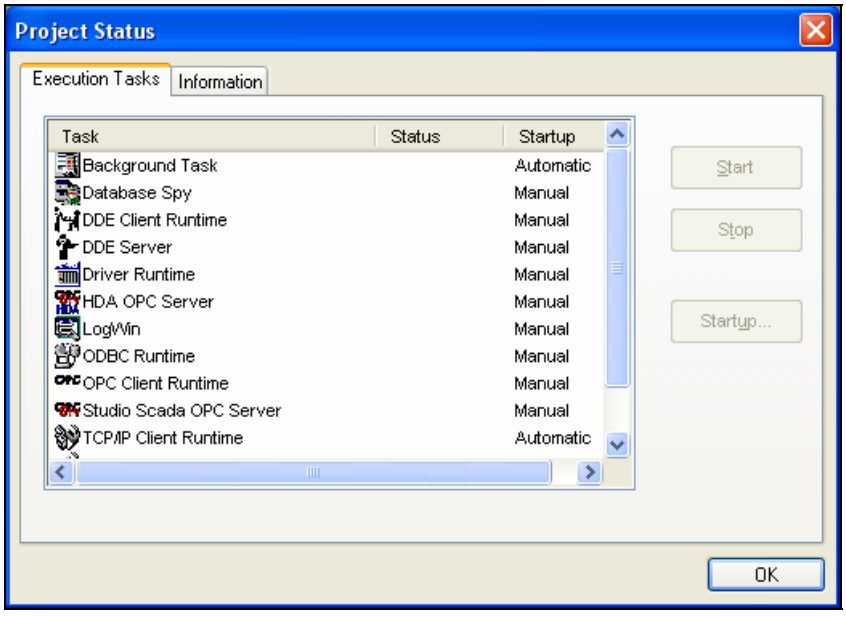
Configuring Project Status

At this point, you typically configure the run-time task for your project application. However, for Local Interface applications, all necessary runtime tasks will be started automatically in the target system—so for this class, no action is required. We will just review how to configure runtime task status.

☑ From the main menu bar, select **Project** → **Status** to open the *Project Status* dialog. This dialog has two tabs.

– You use the **Execution Tasks** tab to monitor and control the execution of each run-time task by starting and stopping the tasks using the **Start** and **Stop** buttons.

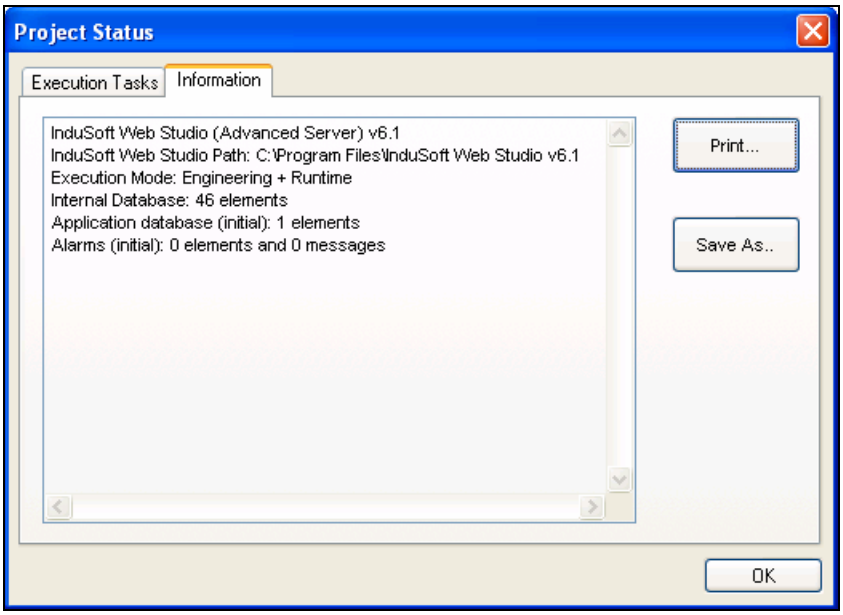
* The **Startup** button toggles between **Automatic** (run-time tasks start automatically) or **Manual** (run-time tasks must be started manually). The specified execution method displays in the *Status* column of the *Execution Task* pane.
* The **Stop** button stops the run-time task.



Project Settings Dialog – Execution Tasks Tab

Note:
The Execution Tasks tab is available only when the Target Station option from the Execution Environment dialog (Project → Execution Environment) is set to Local. With CEView applications, you cannot modify the Startup setting for each task (Automatic or Manual).

- You can use the **Information** tab to view some general information about the development system and your application. Use the buttons on this tab to print the information or save it to a file.



Project Status Dialog – Information Tab

☑ Click **OK** to close the *Project Status* dialog.

Chapter 5. Creating and Editing Tags

Before you can create or edit tags, you must understand some basic concepts about the types of tags, tag values, and tag parameters that you can use in InduSoft Web Studio.

Overview

Tags are variables (such as communication points in field equipment, calculation results, alarm points, and so forth) that are used in screens and worksheets.

You can use the following tag groups with any InduSoft Web Studio module:

- **Application tags** are tags you can create during application development.

  Example application tags 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*.
  For example, to change the time, you must use the proper math function rather than writing the information to the internal time tag.

  Example internal tags include

  – Date tags, which holds the current date in string format
  – Time tags, which holds the current time in string format.
- **Shared tags** are created in PC-based control software 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.

All tag values are stored in the IWS *Application* database and you declare all tags using the *Application Database* module on the **Database** tab.

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

- You can use letters, numbers, and the underscore character (_).
- You must begin the tag with a letter.
- You can use a maximum of 32 characters (for a tag name) or 16 characters (for a class member name).
- Your tag names must be different from internal tag and math function names.
- You can use upper- or lowercase, tag names are *not* case sensitive.

  Example tag names: *Temperature*, *pressure1*, *count*

  Because InduSoft Web Studio does not differentiate between uppercase and lowercase characters you can use both upper and lowercase characters to make tag names more readable (for example: *TankLevel* instead of *tanklevel*).

Defining Tag Value Types

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

- **Boolean** (*one bit*): Boolean or digital variables (0 or 1).
- **Integer** (*four bytes*): 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.
- **String** (*alphanumeric data, 256 chars*): Character string up to 255 characters (from 0 to 254) that holds letters, numbers, or special characters. Supports both ASCII and UNICODE characters.

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

Defining Array, Class, and Pointer Tags

You can define the following types of *Application* tags:

- **Array tags**: A set of tags that have the same name, but use unique *indexes* (a matrix of *n* lines and one column).
- **Class tags**: Tags that allow a high-degree of encapsulation in the *Application* database.
- **Pointer tags**: Tags that provide pointers (*indirect access*) to any other tag type, including class tags. Any string-type tag can be a pointer tag.

Defining Array Tags

InduSoft Web Studio tags can hold a single value or an array of values. An *array tag* consists of a set of tags all using the same name. Array tags can simplify configurations and enable you to use multiplexing in screens, recipes, and communication interfaces. Array tags also can save development time during tag declaration.

You identify array tags using indexes (a matrix of *n* lines and *one* column), and the maximum array size is set by the product type.

You define *array* tags in the *Application* database by specifying **Tag[number]** or **Tag[another tag]** wherever you use a variable name.

For example: **tank[0]**, **tank[1]**, **tank[2]**, **tank[500]**

Because array tags greatly simplify your configuration tasks, we suggest using them whenever possible. Suppose, for example, that you want to create a display to monitor multiple tanks. You can use array tags to configure a single display containing tags linked to any tank, as follows:

pressure[tk], temperature[tk], temperature[tk +1]

Where the **tk** tag is the index containing the number of the desired tank. Array indexes can be tags, numeric values, or expressions with a value plus a tag.

To refer to an array with an arithmetic operation (**+**) index, you must use the following syntax (where ***N*** is a numerical constant):

<tag name>[<tag>+N]

For example: **temperature[tk+2]**, **temperature[tk+6]**

Using array tags also can save you a lot of application development time. Suppose that you need tag points related to the temperature of four tanks. The conventional configuration method is:

temperature1 high temperature on tank 1
temperature2 high temperature on tank 2
temperature3 high temperature on tank 3
temperature4 high temperature on tank 4

Using array tags simplifies this task as follows:

temperature[j] high temperature on tank {j}

When you create a four-position array tag, the system actually creates five positions (from 0 to 4). Therefore, **TagExample[15]** array has 16 elements.

Defining Classes

In addition to defining the standard tag types we just discussed, you can define a new tag type known as a *class*. *Class* tags are structures that permit a high-degree of encapsulation within the Application Database. Class-type tags contain a group of values to be associated with the class.

So, if you create a new tag called **Tank** and its type is **CTank**, you are actually creating a tag with all the properties of the **CTank** class. You use a period (.) separator to access the members of a class tag, as follows:

Tank.Level or **Tank.Temperature**

If **Tank** is an array tag, the syntax would be:

Tank[1].Level or **Tank[n].Temperature**

Class: CTank

	Name	Type	Description
1	Temperature	Real	Tank temperature
2	Pressure	Real	Tank pressure
3	Level	Real	Tank level
*		Integer	

CTank Class Example

Defining Pointer Tags

InduSoft Web Studio allows you to access tags in the database indirectly — where the tag can provide a pointer to any other tag type, including a class type. To specify a pointer tag, you must use the following syntax:

@<name of indirect tag>

For example, assume that a tag ***X*** holds the **Temp** string. Reading from or writing to **@X** provides access to the **Temp** tag value.

	Name	Size	Type	Description
1	@Z	0	Integer	
2	@X	0	Boolean	

Indirect Tag Example

Defining Tag Parameters

A *field* is a set of parameters inherent to each tag in the database. Applications use these parameters at run-time as tag fields. You can set most of the parameters using the *Tag Properties* dialog, accessed by clicking the **Tag Properties** button on the *Tag Properties* toolbar. To access a tag field use the syntax: **TagName->Field**.

You can access the following fields 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	
Description	Description of tag, 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). Field updates every time 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 current 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	✓	✓	✓	×	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

Field Name	Description of Value Associated w/ Each Field	Tag Type Associated with Field				R=Read Only RW=Read+Write
		Boolean	Integer	Real	String	
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
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

Caution:
You cannot use tag fields (such as Bit fields) to configure *Alarm* or *Trend* worksheets.

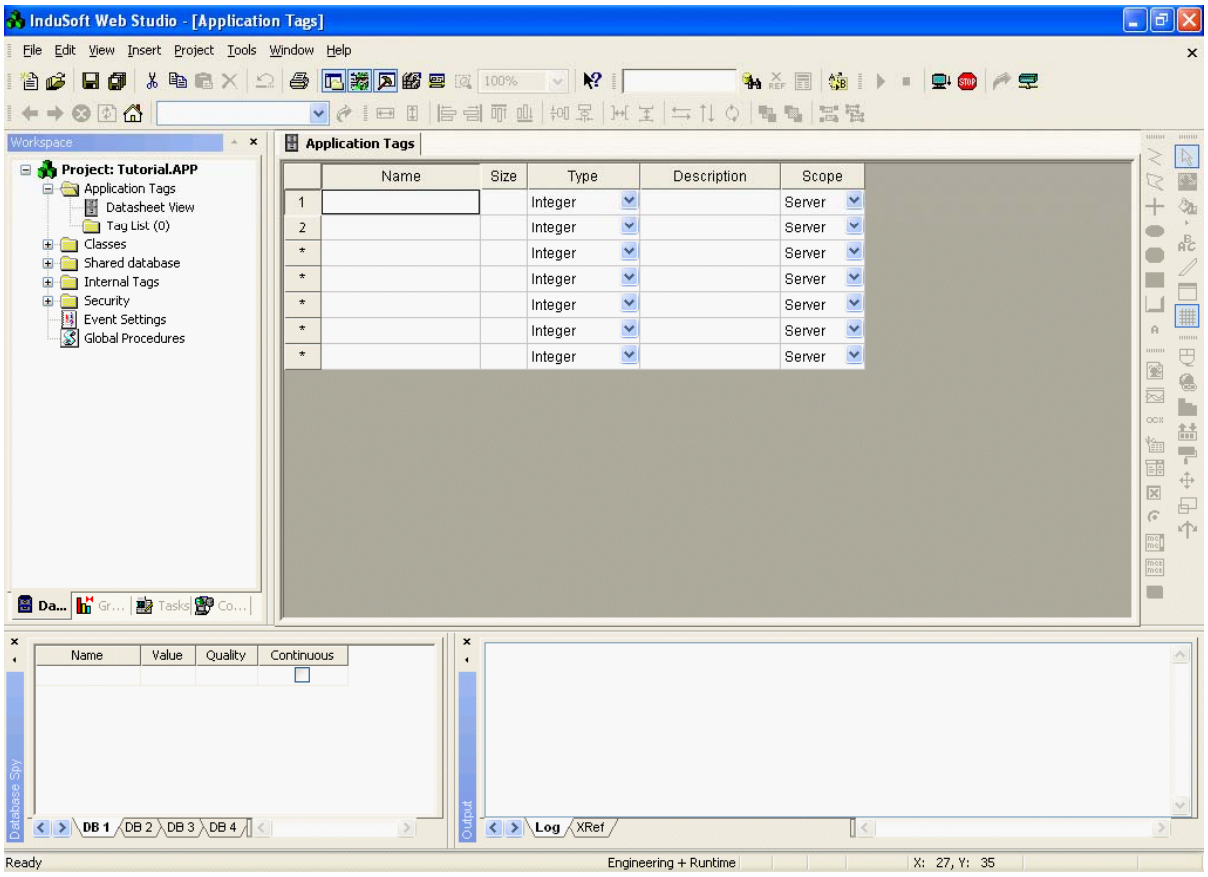
Note:
If the application tries writing a value outside the range specified in the **Min** and **Max** fields, the *Tags* database will not accept the new value and writes a warning message in the *LogWin*. If you configure both **Min** and **Max** properties with the value 0 (zero), any value applied to the tag type can be written to the tag.

Database Exercise

For this exercise, we will define tags for your application, which will have three tanks (with similar configurations) with two valve types: one to fill and one to empty each tank. Your tag definitions will contain values that control the state of the valves (**Valve_Fill_State** and **Valve_Empty_State**). You also will use arrays to quickly configure the tags associated with all three tanks.

Configure the tags as follows:

- ☑ In the *Workspace* window, select the **Database** tab.
- ☑ Open the *Application Tags* folder and then double-click **Datasheet View** to open the *Application Tags* worksheet.



Application Tags Worksheet

- ☑ Click in the first *Name* field (line 1) and type **Valve_Fill_State**.
- ☑ Tab through each of the columns and type the following information for the tag:
  - **Size**: 3
  - **Type**: Boolean
  - **Description**: Fill valve state (openclosed)
  - **Scope**: Server

☑ Tab to line 2 and enter the following information for the next tag:
- **Name**: Valve_Empty_State
- **Size**: 3
- **Type**: Boolean
- **Description**: Empty valve state (openclosed)
- **Scope**: Server

> **Note:**
> In this exercise, we are using only two lines in the database to configure six tags. Arrays reduce the time spent configuring the database and allow you to configure functions and scripts to optimize the overall application.

After configuring tags to receive the valve states, you must configure tags that can send commands to the host controller. These tags will have the same number of states and characteristics as the valve state tags.

☑ Tab to line 3 and enter the following information:
- **Name**: Valve_Fill_Command
- **Size**: 3
- **Type**: Boolean
- **Description**: Fill valve command (open/close)
- **Scope**: Server

☑ Then, tab to line 4 and enter the following:
- **Name**: Valve_Empty_Command
- **Size**: 3
- **Type**: Boolean
- **Description**: Empty valve command (open/close)
- **Scope**: Server

Application Tags

	Name	Size	Type	Description	Scope
1	Valve_Fill_State	3	Boolean	Fill valve state (openclosed)	Server
2	Valve_Empty_State	3	Boolean	Empty valve state (openclosed)	Server
3	Valve_Fill_Comm...	3	Boolean	Fill valve command (open/close)	Server
4	Valve_Empty_Co...	3	Boolean	Empty valve command (open/close)	Server
*			Integer		Server
*			Integer		Server
*			Integer		Server
*			Integer		Server
*			Integer		Server

Completed Lines 3 and 4

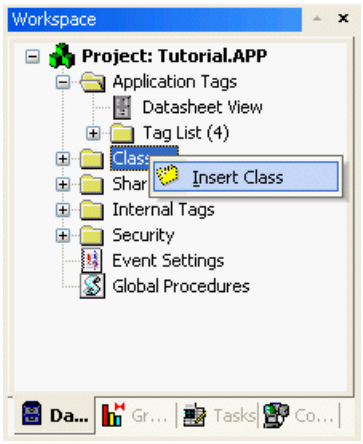
☑ Select **File** → **Save** from the main menu bar to save the information on the *Applications Tags* worksheet.

Next you will create a class tag to store the following properties (or attributes), which are typically associated with tanks:

- temperature
- level
- pressure

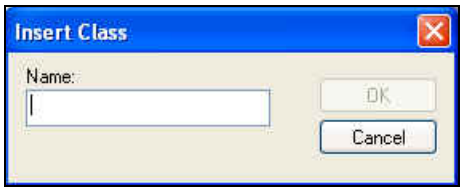
To create the new class tag:

☑ Return to the *Workspace*, right-click the *Classes* folder, and select **Insert Class** from the resulting pop-up menu.



Inserting a Class Tag

☑ When the *Insert Class* dialog displays, type **CTank** into the *Name* field and then click **OK** to close the dialog.

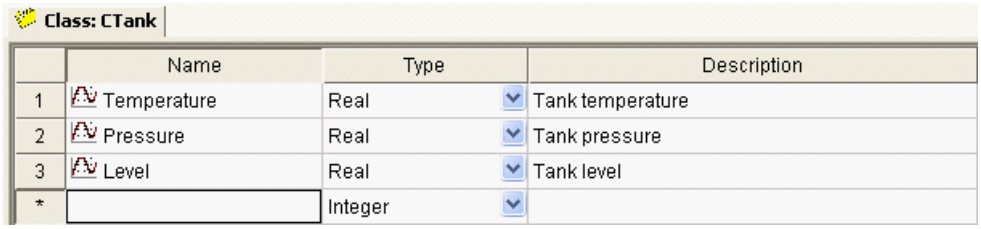


Insert Class Dialog

☑ When the *Class: CTank* worksheet displays, type or select the following information in the columns for lines 1–3:

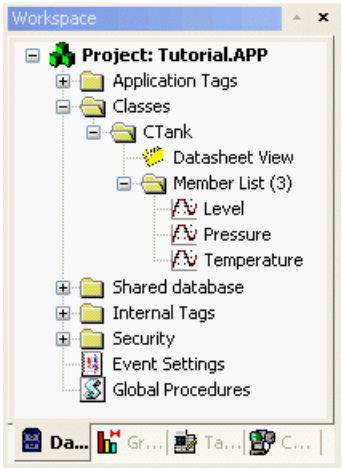
> **Note:**
> All of the tank property class members are analog, so you will declare them to be **Type**: Real.

	Name	Type	Description
Line 1	Temperture	Real	Tank temperature
Line2	Pressure	Real	Tank pressure
Line 3	Level	Real	Tank level



Completed Class: CTank Worksheet

You can expand the *Classes* folder and its subfolders to see the data structure.



Expanded Classes Folder

☑ Now, you have defined each tank property as a member of the *CTank class* tag. Close the *Class: CTank* worksheet.

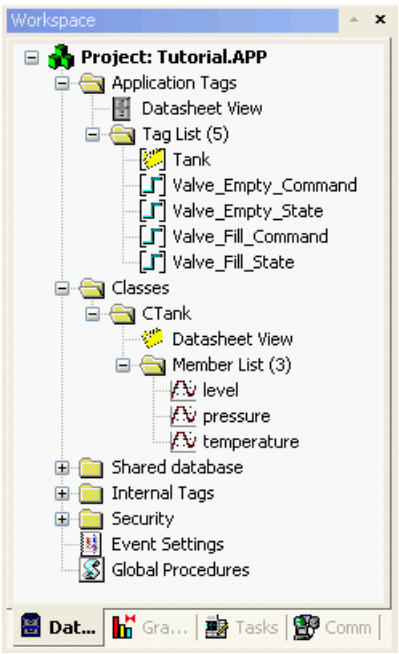
☑ Return to the *Application Tags* worksheet to create/associate a Tag with the new class. Type the following information on line 5:

- **Name**: Tank
- **Size**: 3 (because there are three tanks)
- **Type**: Class: CTank
- **Description**: Tank Data
- **Scope**: Server

Application Tags

	Name	Size	Type	Description	Scope
1	Valve_Fill_State	3	Boolean	Fill valve state (openclosed)	Server
2	Valve_Empty_State	3	Boolean	Empty valve state (openclosed)	Server
3	Valve_Fill_Comm...	3	Boolean	Fill valve command (open/close)	Server
4	Valve_Empty_Co...	3	Boolean	Empty valve command (open/close)	Server
5	Tank	3	CTank	Tank Data	Server
*			Integer		Server
*			Integer		Server
*			Integer		Server
*			Integer		Server
*			Integer		Server

Creating the Tank Class Tag



Expanded Applications Tag and Classes Folders

Shortcuts:

- Create new application tags and classes by right-clicking on the appropriate folders and choosing the **Insert** option from the pop-up menu.
- Modify a tag property by right-clicking on the tag's icon and choosing the **Properties** option from the pop-up menu.
- During application development, if you type a nonexistent tag, InduSoft Web Studio displays a message to ask if you want to create a new tag. If you accept, IWS opens a dialog that you can use to create the tag.

You have finished creating the initial *Application Tags* database for your application. Proceed to the next chapter to create screens for your project.

Chapter 6. Creating Application Screens

Before creating an application screen, you should consider the structure of the screen. IWS applications permit you to open more than one screen at a time. SO, you should create a Header and Footer screens, and a **default** screen with the dimensions (Height, Width, Position and Style), and then you insert objects on the screen. You then save the template screen under different names to create the different screens.

Typically, an application screen consists of three, basic areas (or screen types):

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

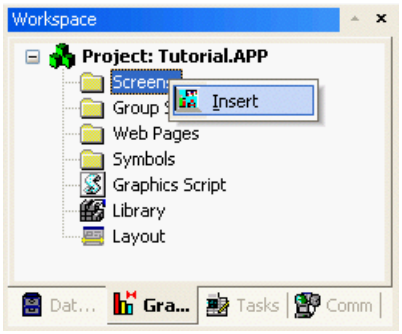
Using this structure to develop screens provides the following advantages:

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

Using this recommended structure, begin the next exercise to create screens for your application.

Exercise: Creating a Standard Screen

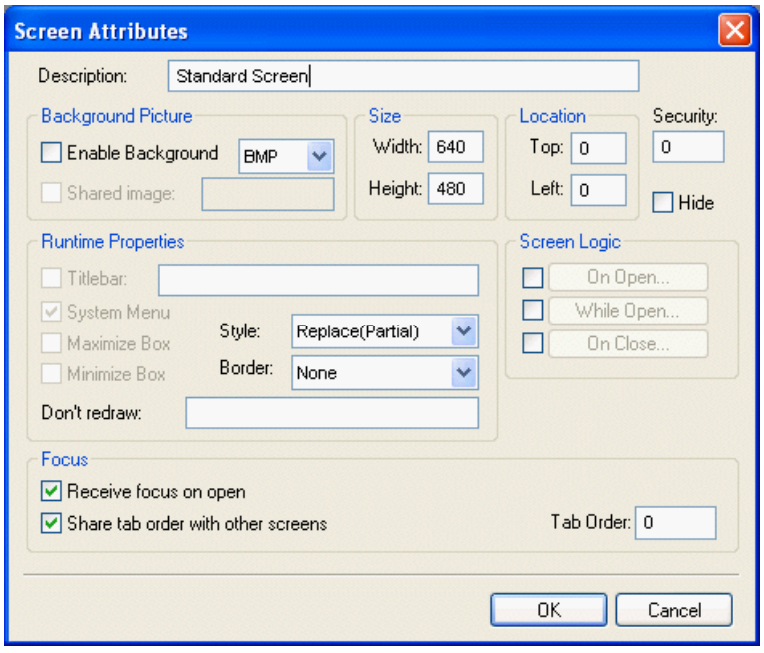
In this exercise, you will create a *Standard* screen as a template for all other screens. By saving the template screen under a new and unique name, you can create application screens that all have the same attributes and basic objects.



Creating a New Screen

Use the following procedure to create your first project screen:

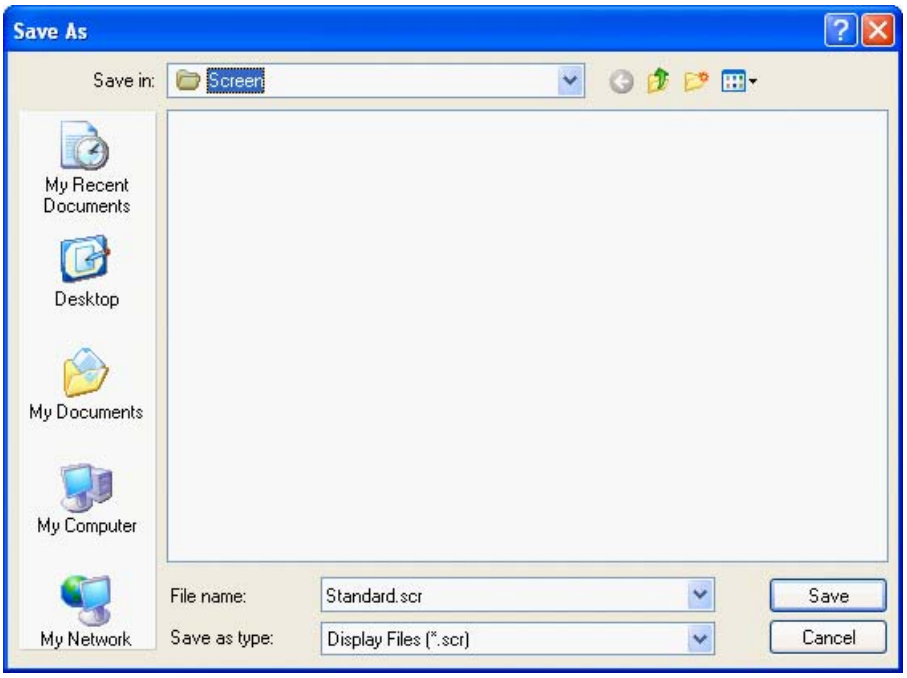
- ☑ In the *Workspace*, select the **Graphics** tab.
- ☑ Right-click the *Screens* folder and select **Insert** or select **Insert →Screen** from the main menu bar.
- ☑ When the *Screen Attributes* dialog displays, type **Standard Screen** into the **Description** field and then click **OK**.



Screen Attributes Dialog

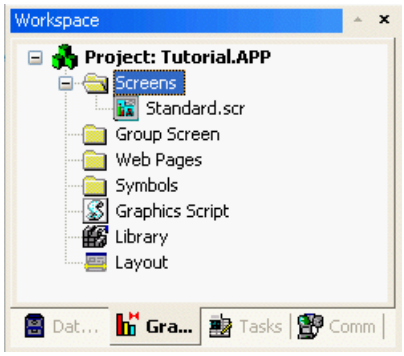
An empty screen displays.

☑ Click the **Save** icon on the *Main* toolbar or select **File → Save** (or **File →Save As**) on the main menu bar to save the new screen. When the *Save* (or *Save As*) dialog displays, type **Standard.scr** (or **Standard**) into the **File name** field.



Specifying the File Name

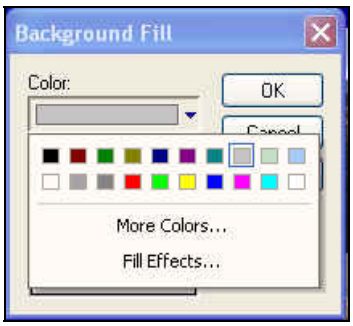
☑ Expand the folders in the **Graphics** tab to see the saved screen.



Checking for the New Screen

☑ Next, change the background color to gray. Click the **Background Color** button located on the *Tools* toolbar or right-click on the blank screen and choose **Background Color** from the pop-up menu.

☑ When the *Colors* dialog displays, click the light gray color, then click **OK**.



Colors Dialog

> **Tip:**
> Double-click on a color to select that color and automatically close the *Colors* dialog.

DRAWING THE BUTTONS AND CREATING DYNAMIC LINKS

The first button you will create will be used to navigate between application screens.

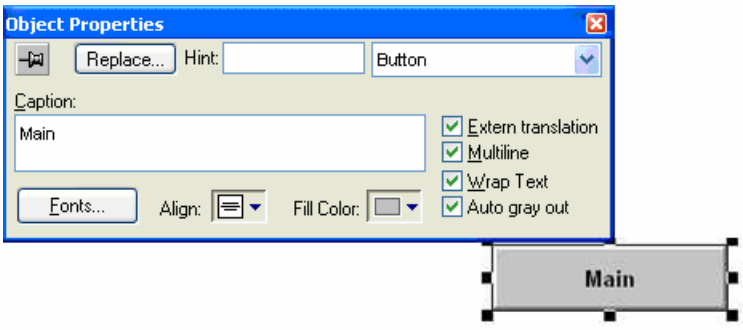
☑ Click the **Button** icon on the *Static Objects* toolbar and draw a button at the top of your display screen. (Suggestion: Make the button about 120 pixels wide and 30 pixels tall. Watch the status bar as you draw the button to monitor the size).

New Button

☑ Right-click the new button and select **Properties** from the pop-up menu, press **Alt**+**Enter**, or double-click on the button to open the *Object Properties* dialog.

By default, **Button** should be displayed in the upper right corner drop-down box. (If not, click the combo-box and select **Button** from the pop-up menu.)

☑ In the *Object Properties* dialog, type **Main** into the **Caption** field to change the caption text on the button.



Changing the Button Caption

☑ Close the *Object Properties* dialog.

Most buttons (except **Log On** and **Exit** button) open screens using the **Open("<*screen name*>")** function. Although you have not yet created all of your screens, you can still configure the buttons.

Continue to the next step to add a dynamic property to your button (*Note*: you use this same procedure to add dynamic properties to any object):

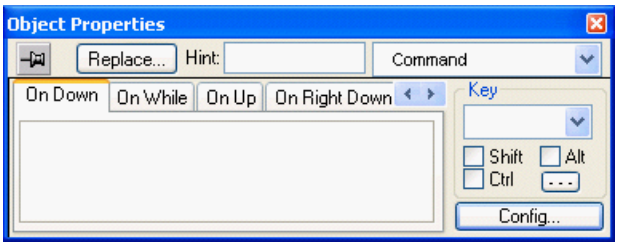
☑ Select the **Main** button object again and click the **Command** button on the *Tools* toolbar. Clicking this button adds the **Command** dynamic property to the **Main** button object.

Command Property off

Command Property on

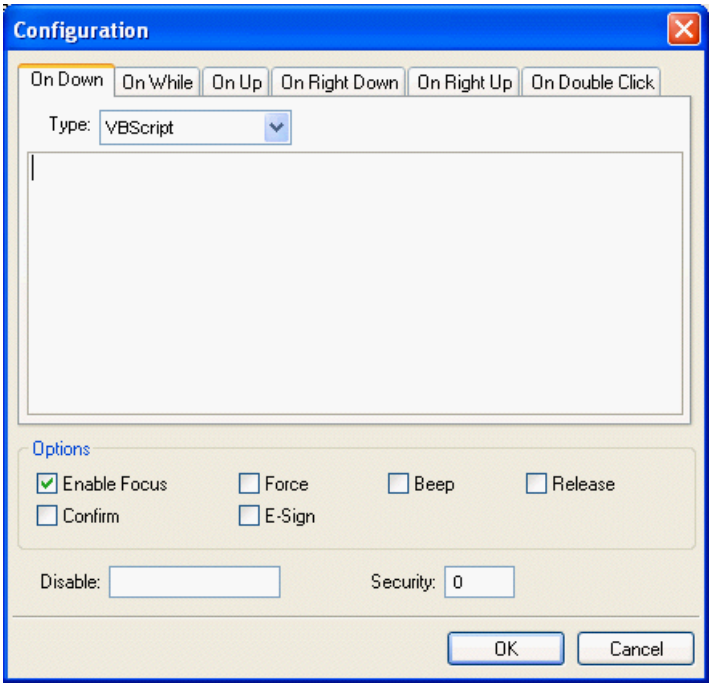
☑ With the **Command** property added, open the *Object Properties* dialog again for the **Main** button object.

In the *Object Properties* dialog, **Command** should now be displayed by default in the upper right corner drop-down box. (If not, click the combo-box and select **Command** from the pop-up menu.)



Command Property in the Object Properties Dialog

☑ Click the **Config…** button to open the *Configuration* dialog.



Configuration dialog

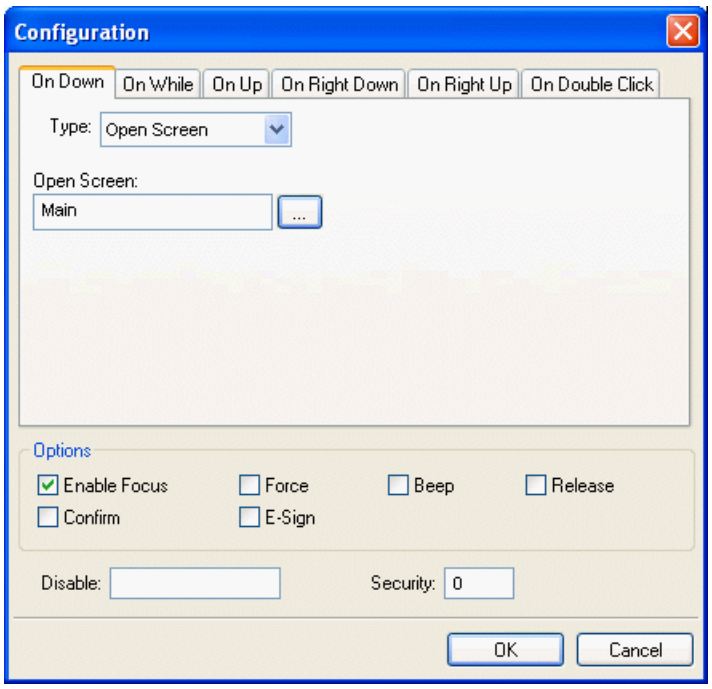
☑ Select the **On Down** tab, if it is not already selected.

Different functions can be defined for each position of the button: when it's pressed down, while it's being held, when it's released up, and so on. These button positions are also known as Events.

☑ Click the *Type* combo-box and select **Open Screen** from the pop-up menu.

In addition to certain pre-configured function types — opening and closing screens, or setting/resetting tag values — you can also implement more complex functions using either VBScript or the built-in IWS scripting language. For this exercise, we're using the pre-configured function **Open Screen**.

☑ Type **Main** in the *Open Screen* field:



Configuring the Main Button to Open Screen "Main"

You can also click on the **…** button to open a standard Windows file browser and select the screen file (`*.scr`) to be opened. However, because we have not yet created the *Main* screen, the screen file is not available to be selected that way. Instead, you must manually enter the screen name.

After you create the *Main* screen in the next exercise, this function will correctly open it.

☑ Click **OK** to close the *Configuration* dialog, and then close the *Object Properties* dialog.

Using this same procedure, create eleven more buttons and label them as follows:

- **Trend Online**
- **Trend History**
- **Recipe**
- **Report**
- **Alarm Online**
- **Alarm History**
- **LogOn**
- **Translation**
- **PLC**
- **OPC**
- **Exit**

Notes:

- To save time and create buttons that are all the same size, you can use the **Copy** (Ctrl+C) and **Paste** (Ctrl+V) commands or press the **Ctrl** key and use your mouse to click-and-drag copies of the original button with the mouse.
- Use the **Alignment** and **Spacing** buttons on the *Tools* toolbar to align and space the buttons symmetrically.
- To change the caption font, click the **Fonts** button and select the new settings. (We suggest using the **Arial Bold** font and font size=10.)

When you are done, the top of your screen should look something like this:

All New Buttons

The rest of the buttons that will open screens can now be configured with the appropriate functions, using the same procedure as the *Main* button.

☑ Select each button, add the **Command** property, and configure a function in the *Object Properties* dialog as follows:

- *Trend Online* button — Open Screen **TrendOnline**
- *Trend History* button — Open Screen **TrendHistory**
- *Recipe* button — Open Screen **Recipe**
- *Report* button — Open Screen **Report**
- *Alarm Online* button — Open Screen **AlarmOnline**
- *Alarm History* button — Open Screen **AlarmHistory**
- *Translation* button — Open Screen **Translation**
- *PLC* button — Open Screen **PLC**
- *OPC* button — Open Screen **OPC**

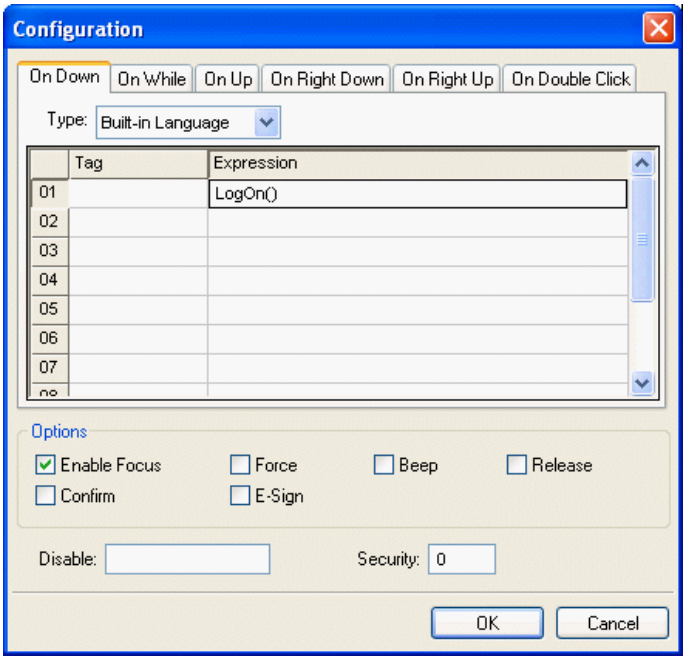
For the *Logon* and *Exit* buttons, there are no pre-configured functions that do what we need. Instead, you must implement the functions using the scripting language.

☑ Select the *Logon* button, add the **Command** property, and go to the button's *Configuration* dialog.

☑ Select the **On Down** tab, if it is not already selected.

☑ Click the *Type* combo-box and select **Built-in Language** from the pop-up menu.

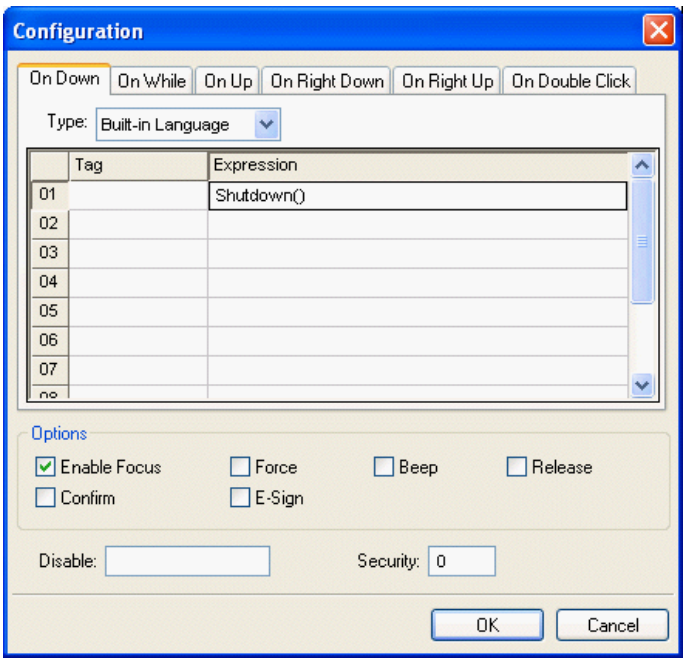
☑ Type **LogOn()** in the *Expression* field:



Configuring the Logon Button with the Scripting Language

☑ Click **OK** to close the *Configuration* dialog, and then close the *Object Properties* dialog.

☑ Use the same procedure to configure the *Exit* button with the expression **Shutdown()**:



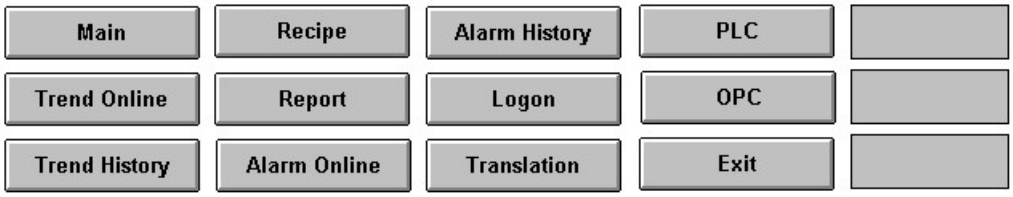
Configuring the Exit Button

When you are finished creating and configuring expression for all twelve buttons, save your work and then continue to the next section to create some Legend objects for your screen.

CREATING LEGEND OBJECTS

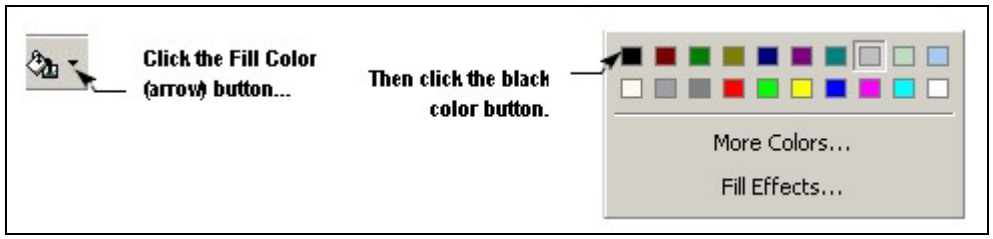
To create a Legend object, you will draw and label some rectangles, and then add dynamic properties to these objects.

☑ Click the **Rectangle** button and draw three rectangles next to your buttons on the screen (similar to the following):



Adding Three Rectangles

☑ Select all of the rectangles (press the **Shift** key and click all three rectangles) and click the **Fill Color** button located on the *Tools* toolbar. When the color palette box displays, select the black fill color.



Selecting a Color

Alternatively, you could use the **Fill Color** option in the *Object Properties* dialog.

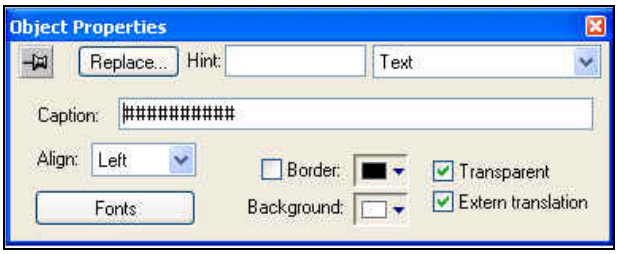
> **Note:**
> Depending on the palette configured for your system, the colors of graphic objects imported into the environment may have color distortion. If this happens, change the palette configured for your system.

☑ Next, you are going to add the following text to these rectangles:

- Ten number signs (#) to the topmost rectangle
- Eight number signs to the middle rectangle
- At least ten number signs to the bottom rectangle

☑ Use the **Text** button on the *Tools* toolbar and click on a rectangle. Type the specified text object to that rectangle and repeat the procedure for the remaining two rectangles.

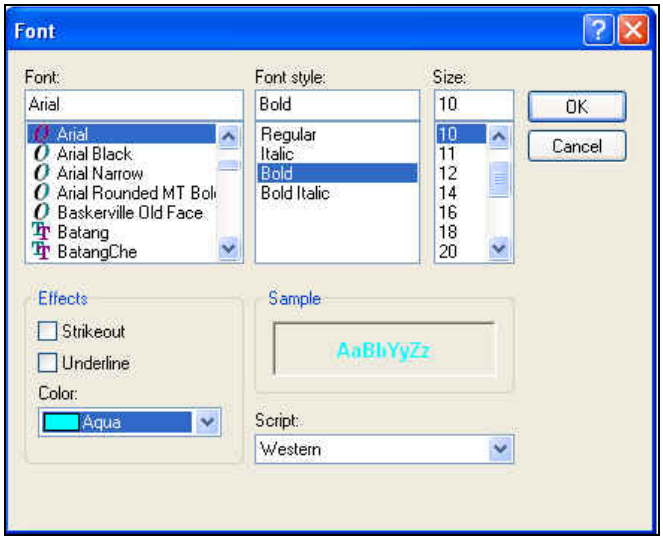
☑ Open the *Objects Properties* dialog for each text box on top of the rectangles, and click (*enable*) the **Transparent** check-box.



Making the Text Transparent

☑ Click the **Fonts** button and set the following text parameters. (Alternatively, you could change font properties using the **Fonts** button on the *Tools* toolbar.)

- **Font**: Arial
- **Font style**: Bold
- **Size**: 10
- **Color**: Aqua



Setting the Font Properties

Your rectangles should now look something like this:

Completed Rectangle Objects

Continue to the next section to apply dynamic properties to these objects.

ADDING DYNAMIC PROPERTIES

You are going to associate the following dynamics to your legend objects:

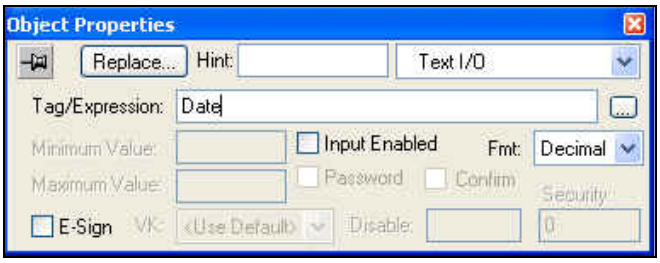
- Top rectangle: System date
- Middle rectangle: Time
- Bottom rectangle: User name

Use this procedure to add these dynamics:

☑ In the display window, select the number sign text in the topmost rectangle.

☑ Click the **Text I/O** button on the *Tools* toolbar.

☑ Open the *Object Properties* dialog for the text. Check the upper-right corner to verify that Text I/O property is displayed in that field. (If not, click the drop-down arrow and select **Text I/O** from the menu.)



Object Properties Dialog–Text I/O

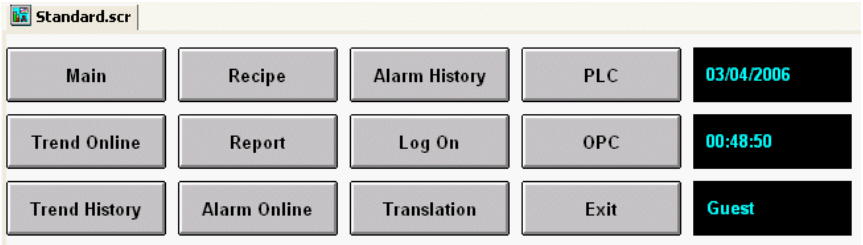
☑ Type **Date** (internal tag that holds the system date) into the *Tag/Expression* field.

Now, when you run the application, the system date will display in place of these number signs.

☑ Repeat this process for the next two text objects.

- **Middle rectangle**: Type **Time** (internal tag that holds the system time) into the *Tag/Expression* field.
- **Bottom rectangle**: Type **UserName** (internal tag that holds a string identifying the current user of the application) into the *Tag/Expression* field.

☑ Now, you can quickly test these functions by clicking the **Test Display** button on the *Execution Control* toolbar. You should see something similar to the following:



Testing the Dynamics

☑ When you are done, click the **Stop Test Display** button .

In the next exercise, you will use the *Standard* screen as a template, and create the *Main* screen for your application. Continue to the next page.

Exercise: Creating the Main Screen

The *Main* screen will always be the first screen to open when you start the application.

In this lab, you will create the *Main* screen and add the following:

- Three tanks with temperature, pressure, and level properties (applied graphically and numerically)
- Valves with open and closed state conditions
- Commands to open or close each valve

> **Notes:**
> Because the three tanks have the exact same characteristics, we will build just one screen that is generic for any tank in our application. To switch to a different tank, we will simply change the index of the tags in the array used in our configuration.

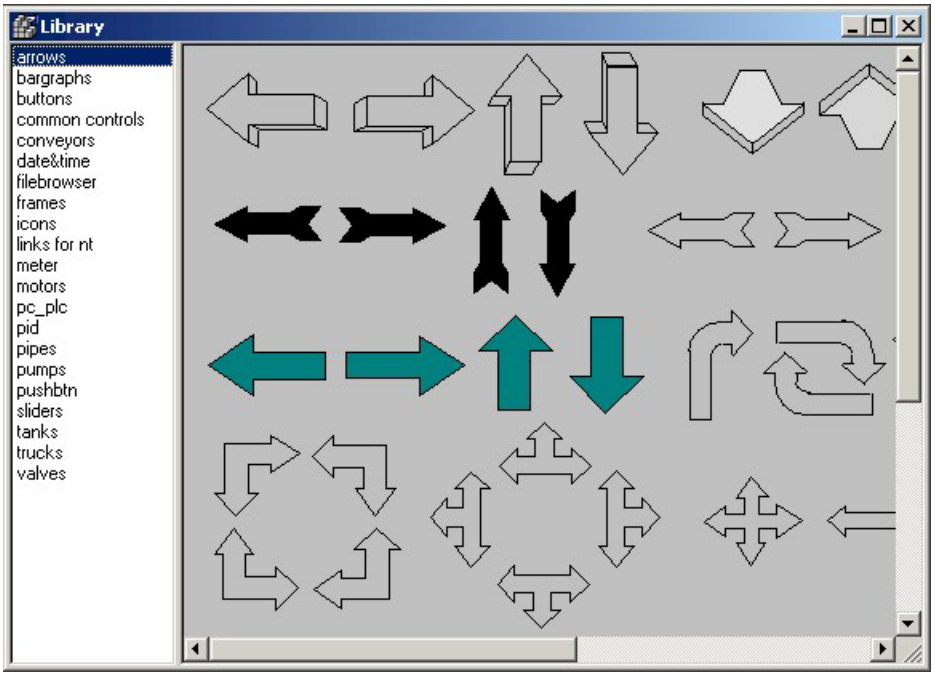
Use the following steps to create the *Main* screen:

☑ Select **File** → **Save As** from the main menu bar.

☑ *When the Save As* dialog displays, save the *Standard.scr* screen as **Main.scr**.

> **Note:**
> The *Standard.scr* screen still exists to be used as a template for other screens.

☑ Click the **Open Library** button on the *Standard* toolbar to open the **Symbol Library**, which contains all the preconfigured tank and plumbing graphics you will need to complete this lab (including arrows, valves, pipes, and tanks).



Symbol Library Window

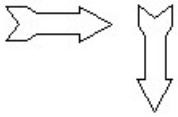
Alternatively, you could right-click the *Library* folder on the **Graphics** tab in the *Workspace* and select **Open** from the pop-up menu, press **Ctrl**+**A**, or select **View** → **Library** from the main menu bar.

☑ In the left pane, select **tanks** from the list.

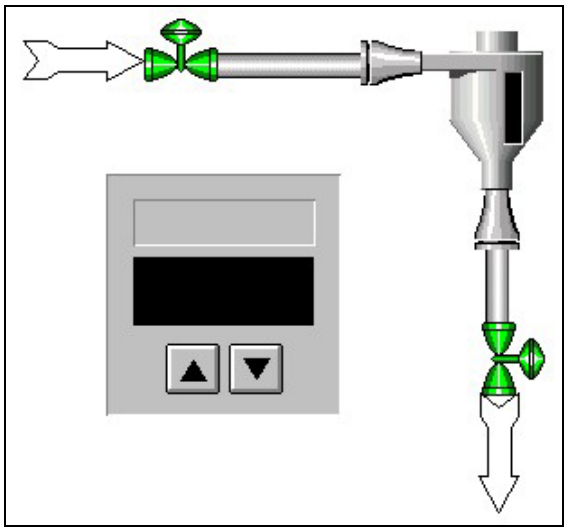
☑ In the right pane, locate the following tank object, and then use a click-and-drag movement to place the object on your screen.

Tank Object

☑ Repeat the procedure to locate and place the following objects onto the *Main* screen:

Select from Right Pane	Locate these Objects
arrows	
arrow buttons	
frames	
pipes (*vertical pipe not shown*)	
valves	

☑ Arrange these objects on your screen to resemble the following arrangement (you may have to resize and rotate some of the objects):



Tank Configuration

> **Note:**
> You could use drawing tools available on the *Tools* toolbar to draw similar objects, but using these preconfigured objects greatly simplifies the screen creation process.

☑ Create a new Tag in the *Application* database (review the procedure on page 5–7 if necessary) with the following specifications:

- **Name**: TankID
- **Size**: 0
- **Type**: Integer
- **Description** (*optional*)
- **Scope**: Local

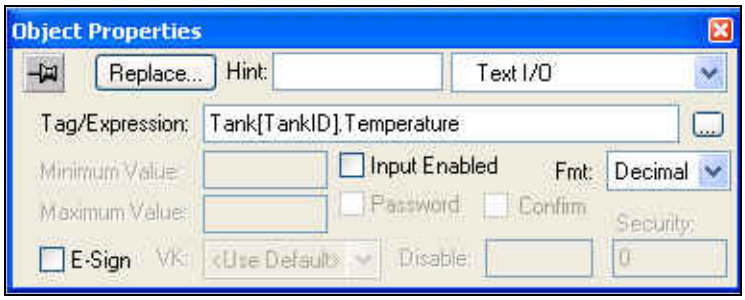
Now, continue to the next step to configure the links on your screen.

☑ Create the following text objects:

- **Temperature**
- **###F**
- **Pressure**
- **### mca**
- **Level**
- **### m**

☑ Select the **### F** text object and click **Text I/O** button on the *Tools* toolbar.

☑ Open the *Object Properties* dialog and type **Tank[TankID].Temperature** into the **Tag/Expression** field to associate this text object with the *TankID* tag.



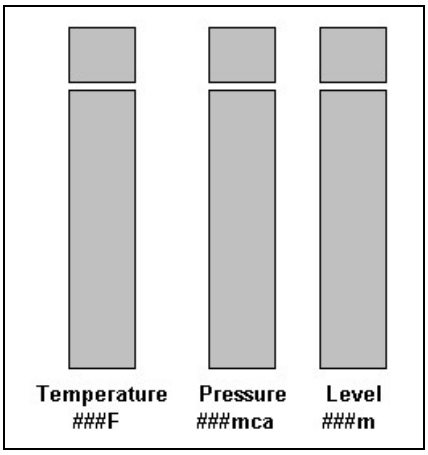
Associating the Text with a Tag

☑ Repeat the process and type the following text into the appropriate Tag/Expression field.

- **### mca**: Tank[TankID].Pressure
- **### m**: Tank[TankID].Level

☑ To show the values of temperature, pressure, and level in a graphical format, click the **Rectangle** button on the *Tools* toolbar and draw three rectangles on your screen.

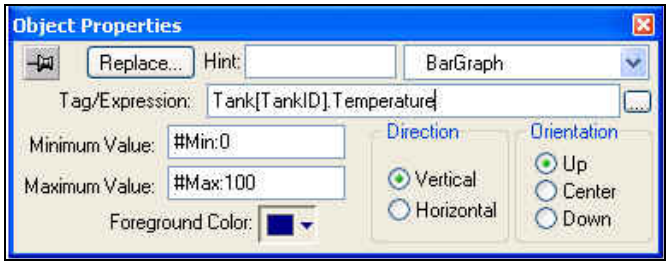
Arrange these rectangles and the text objects as shown in the following figure:



Rectangle Arrangement

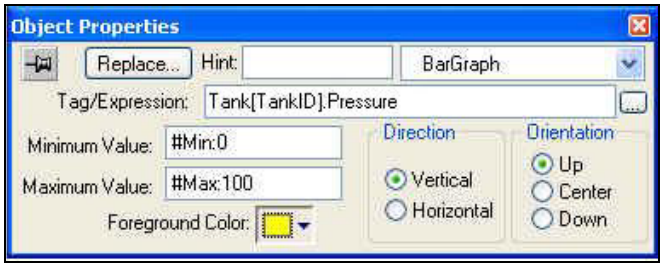
☑ To associate a previously configured tag to these rectangles, select each rectangle and then click the **Bargraph** button.

– Configure the *Temperature ### F* rectangle as follows (**Foreground Color** is dark blue):



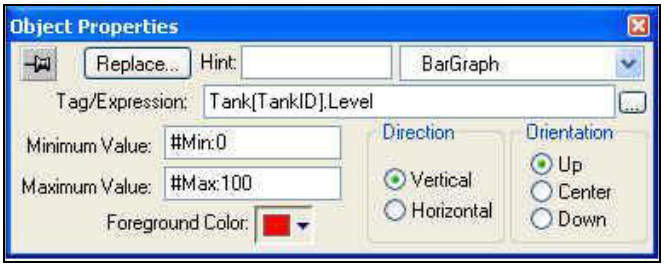
Temperature Rectangle Properties

– Configure the *Pressure ### mca* rectangle as follows (**Foreground Color** is yellow):



Pressure Rectangle Properties

– Configure the *Level ### m* rectangle as follows (**Foreground Color** is red):



Level Rectangle Properties

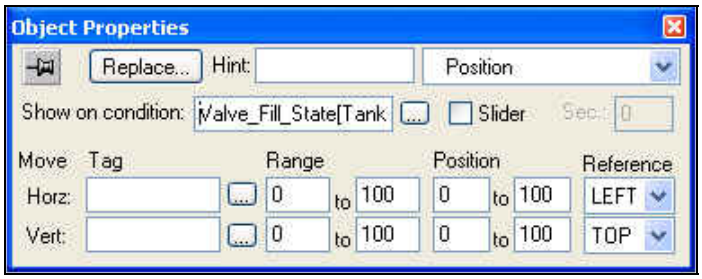
☑ To remind yourself of the colors you selected for the indicators, click each of the smaller rectangles above the indicators and use the *Object Properties* dialogs to specify the appropriate colors.

☑ For the *Valve_Fill* and *Valve_Empty* valves, configure the following links:

– **Color** (to show the valve's status)

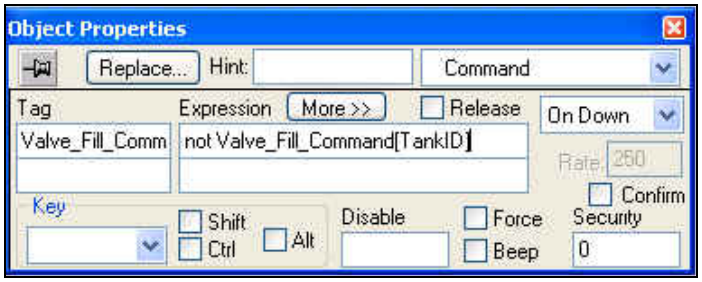
– **Command** (to allow authorized users to send commands to the valves)

☑ Double-click on the Fill valve and when the *Object Properties* dialog displays, select the **Position** property. Configure it as follows:



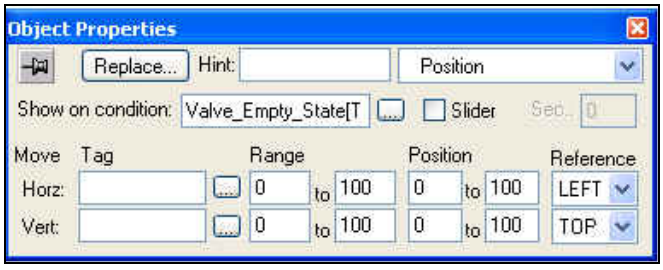
Valve_Fill_State[TankID]

☑ Click on the drop-down arrow in the upper right corner of the *Object Properties* dialog and select **Command** from the menu. Configure as follows:



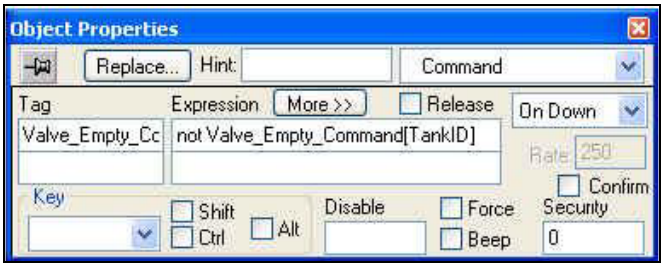
Valve_Fill_Command[TankID]

☑ Now, double-click on the Empty valve and when the *Object Properties* dialog displays, select the **Position** property. Configure it as follows:



Valve_Empty_State[TankID]

☑ Click on the drop-down arrow in the upper right corner of the *Object Properties* dialog and select **Command** from the menu. Configure as follows:



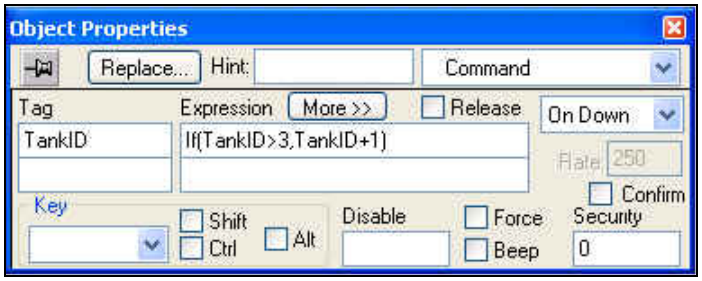
Valve_Empty_Command[TankID]

> **Note:**
> The *security* associated with these valves will block the action (or command) from unauthorized users.

The last task in this Lab is to configure the buttons. To do this, you must change the index of the array tags used on the screen (for example, switch the tank number).

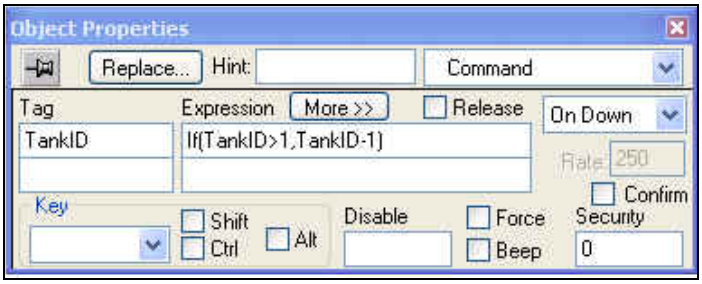
☑ Go to your screen, select the **Increase** button, and click the **Command** button.

☑ When the *Object Properties* dialog displays, configure it as follows:



Configuring the Increase Button

☑ Then go back to the screen, select the **Decrease** button, and click the **Command** button. Configure as follows:



Configuring the Decrease Button

To see how many tanks are being supervised, configure your counter with the following two text strings:

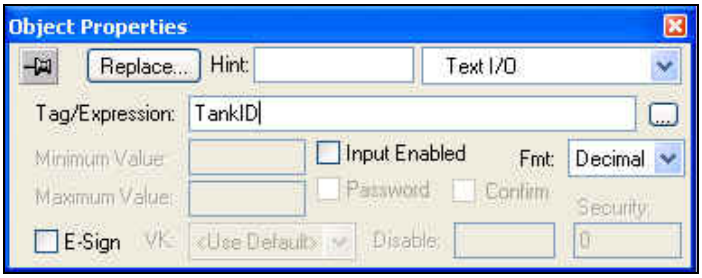
- Tank Number
- #

Your counter should look similar to the following figure.

Finished Counter

☑ Double-click on the # string, and when the *Object Properties* dialog displays, select **Text I/O** from the drop-down menu.

☑ Type **TankID** into the **Tag/Expression** field.



Tag/Expression: TankID

☑ You must configure one link (Bargraph) for the main tank to show the level animation in the tank. Double-click the tank object to open the *Object Properties* dialog, and select **BarGraph** from the drop-down menu.

☑ Configure this dynamic as follows (**Foreground color** is red):

Tag/Expression: Tank[TankID].Level

☑ Save your work and verify the application (**Tools → Verify Application**).

☑ Select **Project → Run Application** from the main menu bar or click the **Run Application** icon from the toolbar. (Use the *DatabaseSpy* and *Output* windows to debug your application.)

Please note that as you run your application now, the screen does not display any activity in the process — that is, the tank valves and measurements don't seem to change. This is because the relevant tags do not yet contain any field process data. Normally, such data are communicated to the application by field equipment (e.g. a PLC or Soft Control) or an external database, but for the purposes of this tutorial, a field process simulator running as a background task will provide the data.

Before we proceed with creating the other application screens, we will learn how to work with scripting languages and create the process simulator.

Chapter 7. Working with Scripting Languages

Expressions, operations, conditional statements, and programming functions are all created using integrated scripting languages. You've already used one of these languages in the previous exercise, to create simple If/Then expressions and Not operations.

Indusoft Web Studio offers two integrated scripting languages: the built-in IWS scripting language and the industry standard Visual Basic Script (or VBScript). Complete descriptions of both languages are provided in both the *IWS Users Guide* and the online help (**Help → Technical Reference**).

Overview of Built-In IWS Scripting Language

At this point we are ready to discuss the built-in IWS scripting language syntax and functions. You can use the built-in language in many places, for example:

- Dynamic object properties in the Application Builder
- Screen logic in the Application Builder
- Scheduler Worksheets
- Math Worksheets

  Math Worksheets have two columns:

  – **Tag Name**: Names of tags to receive results from expressions specified in the Expression column on the same line.
  – **Expression**: Mathematical expressions defined by InduSoft Web Studio.
  – For example, **Tag Name a**, will receive the result of **Expression (10c)-5**.

	Tag Name	Expression
1	a	10*c-5

Example Math Worksheet

IMPORTANT!
- No attributions are done on the **Expression** column. If you write A=2 in this column, IWS will compare A with the number 2. The integer result of this expression (Boolean value *0* if false or *1* if true) will be written to the tag in the **Tag Name** column.
- The system is not case-sensitive.
- To add comments to an expression line, use the // characters.

The following data types are acceptable:

- **Integer numbers** (32 bits): 1 23 45 -123
- **Floating point** (8 bytes): 1.234 -775.344
- **Hexadecimal integer numbers** (32 bits): 0x5 0xA0 0xBC4
- **Strings** (255 characters): “demo” “new demo”

Accessing the Application Database (Examples)

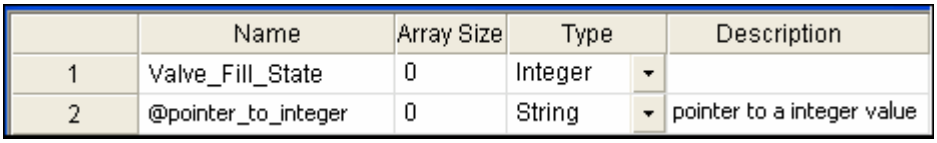
To read a value in the database, use the tag name directly, for example:

- **Example 1**: In the following script line, the X tag will receive the sum of two tags, level and temp.

	Tag Name	Expression
1	X	Level+temp

Example 1

- **Example 2**: IWS allows you to read and write tags using references or pointers. You can declare a tag being used as pointer to another tag in two ways:
  – As a *string* (a pointer to an undefined type)
  – As a *pointer* of a specific kind (*recommended*)



	Name	Array Size	Type	Description
1	Valve_Fill_State	0	Integer	
2	@pointer_to_integer	0	String	pointer to a integer value

Example 2

In the prececding figure Valve_Fill_State is a variable of a string type that is a pointer. The @pointer_to_integer variable is a pointer to integer values.

Notes:
The syntax @tag allows a tag to access another tag by reference. You can use any tag declared as a string as an indirect tag (*pointer*).

Using Operators

InduSoft Web Studio supports all the following operators:

Arithmetic Operators		**Logic Operators**	
+	addition	**AND**	AND, logic
-	subtraction	**NOT**	NOT, logic
*	multiplication	**OR**	OR, logic
/	division	**XOR**	exclusive or, logic
>	greater than	**&**	AND, bit
<	less than	**|**	OR, bit
=	equal	**~**	NOT, bit
>=	greater than or equal to	**^**	XOR, bit
<	less than or equal to	**>>*n***	Rotate *n* bits to right.
<>	different than (unequal to)	**<<*n***	Rotate *n* bits to left.

Using Functions

The function tag names used in IWS must conform to the following syntax:

- **num[Name]** Numerical tag or value
- **str[Name]** String tag or value
- **tag[Name]** Tag Name
- **optNum[Name]** Optional Numerical tag or value
- **optStr[Name]** Optional String tag or value
- **optTag[Name]** Optional Tag Name

This syntax identifies the argument types required for each parameter of the IWS function.

IMPORTANT!
You can use the *Database Spy* window to execute any math expression by writing the expression in the **Tag Name** field and clicking the **Toggle** button. The return value of the expression will display in the **Value** field.

Overview of VBScript

The Microsoft Visual Script Language (VBScript), is a simple, standard and flexible scripting language that allows you to implement logics and algorithms within the IWS application.

IWS implements the Microsoft Visual Basic Scripting Edition 5.5 or higher. Because IWS hosts VBScript, the user can take advantage of all features provided by this language, such as:

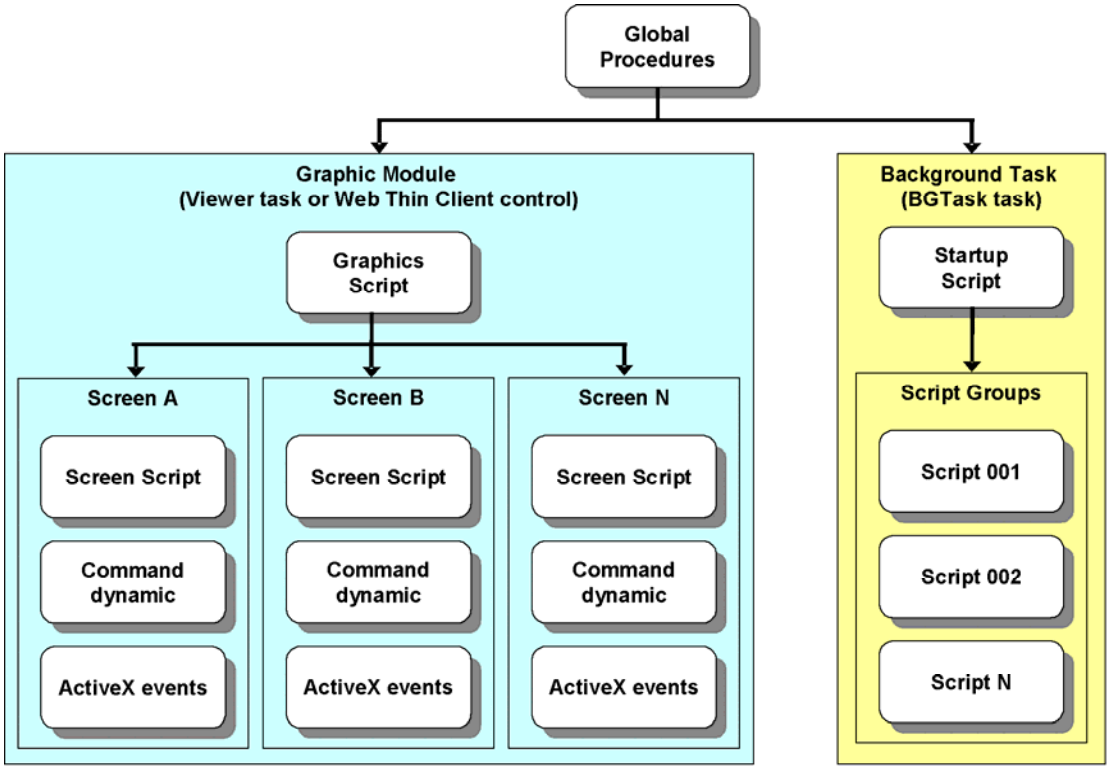
- Use syntax, operators and functions available in the language
- Create new variables and Procedures (Functions and/or Sub-routines)
- Access properties, methods and/or events from COM objects, including ActiveX controls
- Execute the logics in any platform that supports VBScript, including Microsoft Windows NT/2K/XP (IWS Server station), Microsoft Windows CE (CEView) and Microsoft Internet Explorer (Web Thin Client).

VBScript in IWS

The following table provides a summary of the VBScript interfaces supported by IWS:

–Interface	–Scope for Procedures and Variables	–Execution	–Functionality
–Global Procedures	–Graphics and Tasks	–-	–Declaration of Procedures
–Graphics Script	–Graphics Script interface only	–Server (Viewer) + Web Thin Clients	▪ Declaration of Variables ▪ Declaration of Procedures ▪ Execution
–Screen Script	–Screen where the script is configured	–Server (Viewer) + Web Thin Clients	▪ Declaration of Variables ▪ Declaration of Procedures ▪ Execution
–Command Dynamic	–Object where the script is configured	–Server (Viewer) + Web Thin Clients	▪ Declaration of Variables ▪ Execution
–ActiveX Events	–Object where the script is configured	–Server (Viewer) + Web Thin Clients	▪ Declaration of Variables ▪ Execution
–Startup Script	–All Script Sheets from Tasks	–Server (BGTask)	▪ Declaration of Variables ▪ Declaration of Procedures ▪ Execution
–Script Groups	–Script Group only	–Server (BGTask)	▪ Declaration of Variables ▪ Execution

The following picture illustrates the scope of each VBScript interface and the order that they are scanned by IWS:



The illustration shows that the Global Procedures are shared by the Graphic Module and the Background Task. However, the other VBScript interfaces are either from the Graphic Module or from the Background Task, and they do not share variables or procedures between them. They are independent of each other.

> **Note:**
> Although the Graphics Script is scanned by IWS before the Screen Scripts, the procedures and variables declared in the Graphics Script interface are NOT available for any script interface configured on the screens. You must use the Global Procedures interface to implement procedures that must be available for all screens.

When writing your code in a VBScript interface, you can access any tag from the IWS tags database or any function from the IWS built-in language by applying the “$” prefix to the tag/function name, as in the examples below:

- **$Time** 'Returns the value of the tag Time from the tags database
- **$MyTag** 'Returns the value of the tag MyTag from the tags database
- **$Open("Main")** 'Executes the Open() built-in function to open the "Main" screen

Therefore, you can create scripts using built-in functions from IWS, tags from the IWS tags database, VBScript functions, VBScript variables, ActiveX properties, methods or events, and any other interface available. The IWS tags are shared by all modules from IWS, including the Graphic Module and the Background Task.

The following VBScript interfaces are available in IWS:

- Global Procedures
- Graphic Module - Graphics Script
- Graphic Module - Screen Script
- Graphic Module - Command Dynamic
- Graphic Module - ActiveX Events
- Background Task - Startup Script
- Background Task - Script Groups

Chapter 8. Configuring Math and Script Worksheets

Before continuing our application development, we must create a field process simulator to generate simulated data. Because these data must be updated constantly, we will create this simulator in a background task. In a real world application, these data would be coming from field equipment (e.g. a PLC or Soft Control) or an external database.

Remember that IWS incorporates both its own built-in scripting language and the industry standard VBScript. This process simulator may be created using either language, and exercises describing both are provided in this chapter. To use the built-in IWS scripting language, you must configure a Math worksheet. To use VBScript, you must configure a Script worksheet.

Complete descriptions of both languages are provided in both the *IWS Users Guide* and the online help (**Help → Technical Reference**).

> **IMPORTANT!**
> For the purposes of this tutorial, you may choose to configure the Math and Script worksheets together in the same application project. However, only one of the two should actually be enabled for execution when you run your application.

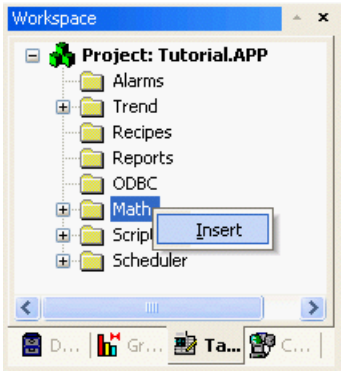
Exercise: Configuring a Math Worksheet

(Simulating the Field Process)

Use the following procedure to configure a Math worksheet:

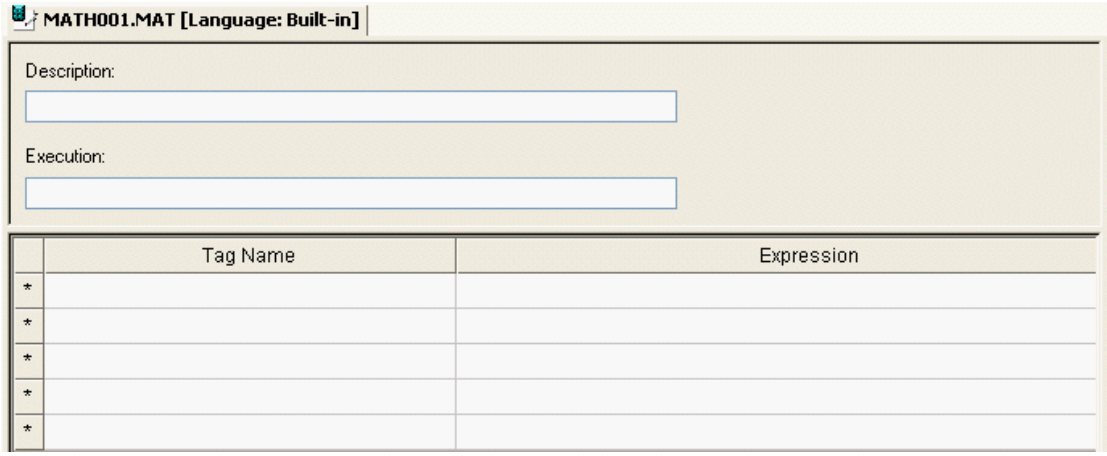
☑ In the *Workspace*, select the **Tasks** tab and then right-click on the *Math* folder.

☑ 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. Whatever it is, 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 create 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:

MATH001.MAT [Language: Built-in]

Description: Field Process Simulator

Execution: 1

	Tag Name	Expression
1	Valve_Empty_State[1]	Valve_Empty_Command[1]
2	Valve_Empty_State[2]	Valve_Empty_Command[2]
3	Valve_Empty_State[3]	Valve_Empty_Command[3]
4	Valve_Fill_State[1]	Valve_Fill_Command[1]
5	Valve_Fill_State[2]	Valve_Fill_Command[2]
6	Valve_Fill_State[3]	Valve_Fill_Command[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(Valve_Empty_State>Valve_Fill_State AND Tank[J].Level>0,Tank[J].Level-1)
15	Tank[J].Level	If(Valve_Empty_State<Valve_Fill_State AND Tank[J].Level<100,Tank[J].Level+1)
16	Next	
*		

Completed Math Worksheet

Caution:
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.

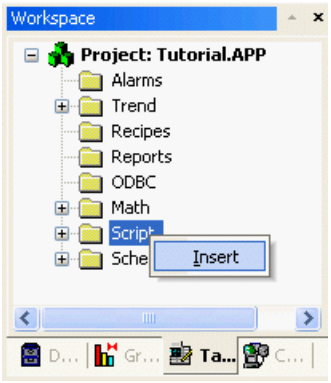
Exercise: Configuring a Script Worksheet

(Simulating the Field Process)

Use the following procedure to configure a Script worksheet:

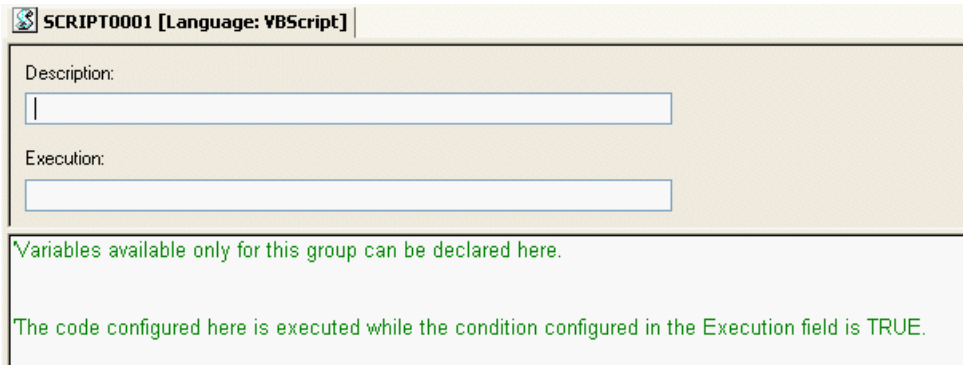
☑ In the *Workspace*, select the **Tasks** tab and then right-click on the *Script* folder.

☑ When the pop-up menu displays, select the **Insert** option:



Opening a Script Worksheet

A blank Script worksheet displays:



Blank Script 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**.

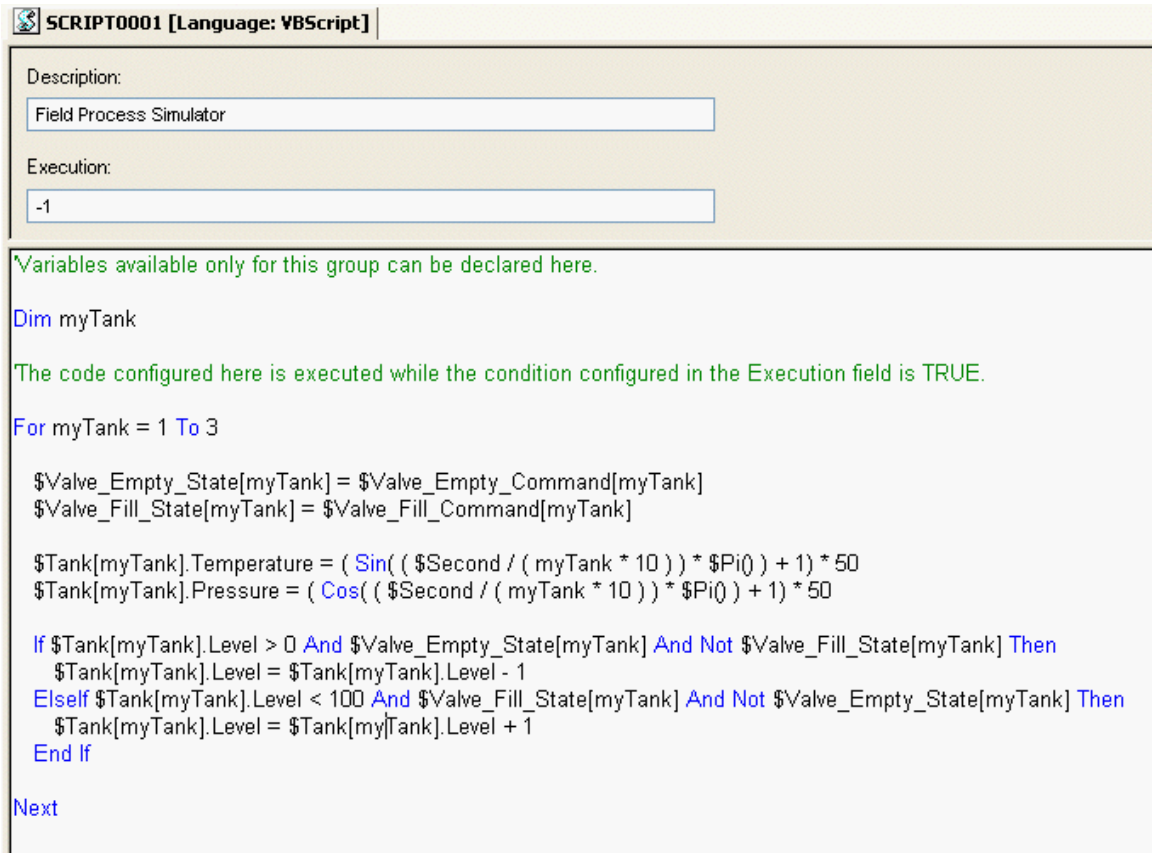
☑ In the **Execution** field, type **-1**. This enables continuous execution of this Math worksheet; the value -1 is always a TRUE condition.

> **Note:**
> That's not a mistake — in VBScript, TRUE is equivalent to -1.

Next, to generate simulated process data that can be used by the various screens, we must create 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

Caution:
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.

Chapter 9. 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

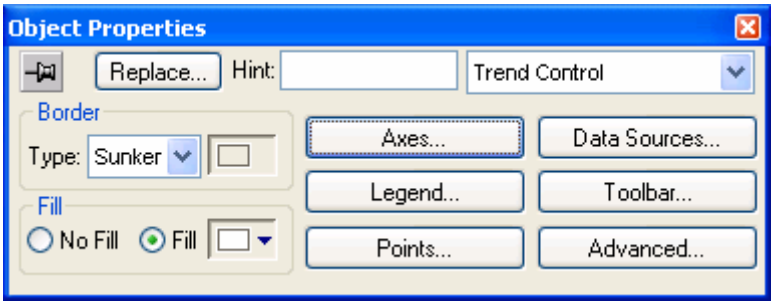
The Trend Control object displays data points (values) from different data sources in a graphic format. The main features provided by the Trend Control object are:

- 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, which provides interfaces for the user to interact with the Trend Control object during the runtime
- Built-in legend, which displays the main information associated to each pen linked to the object
- Zooming and auto-scaling tools
- Horizontal and vertical orientation

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

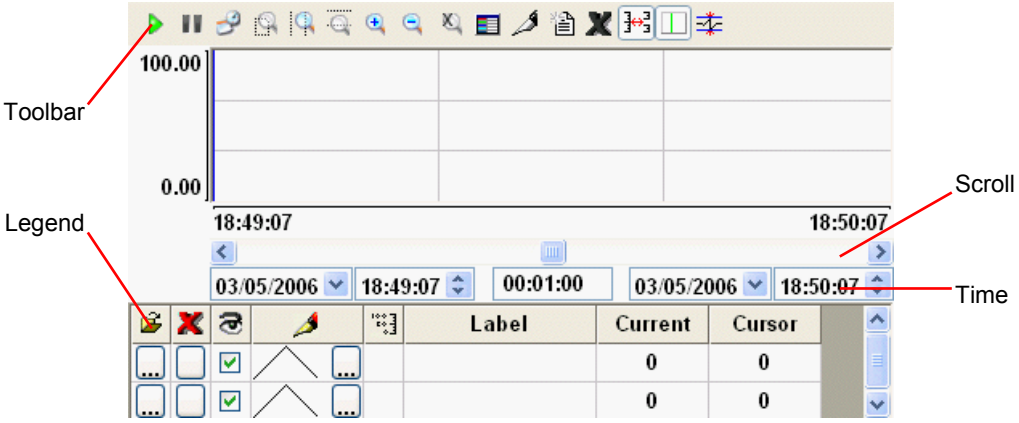
– **Border** box: Specify a border line Type (style) by clicking on None, Solid, Dashed, Etched, Raised or Sunken. You can also select the color of the border line with the color box to the right of the Type field.

- **Fill** box: If you click Fill, you can choose a background color for the Trend Control object by selecting it from the color box to the right of this radio button. If you select No Fill, the background of the Trend Control object will remain transparent.
  The rest of the buttons on this dialog launch other dialogs for configuring specific settings for the Trend Control object:
- The **Axes** dialog allows you to configure the settings for the X and Y axis.
- The **Data Sources** 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.
- The **Legend** option lets you choose whether to show an embedded legend during the runtime, and if so, which fields to display in it.
- The **Toolbar** option lets you choose whether to show an embedded toolbar during the runtime, and if so, which buttons that trigger actions to display in it.
- The value of each data **Point** can be represented in the Trend Control object as a pen, during the runtime. You can select which data Points will be visible during the runtime (add/remove pens to the chart), regardless of how many data Points you associate with the Trend Control object.
- The **Advanced** dialog 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:

- The **Toolbar** option lets you choose whether to show an embedded toolbar during the runtime, and if so, which buttons that trigger actions to display in it.
- The **Legend** option lets you choose whether to show an embedded legend during the runtime, and if so, which fields to display in it.
- Using the **Scroll** bar, you can slide through the X-axis values, according to the period configured for this scale.
- Using the **Time** bar, you can modify the Duration, as well as the Start Date/Time and/or the End Date/Time, for the data displayed on the object. Changing these values will affect the tags associated with the X-axis scale (if any).



Embedded Interfaces in the Trend Control Object

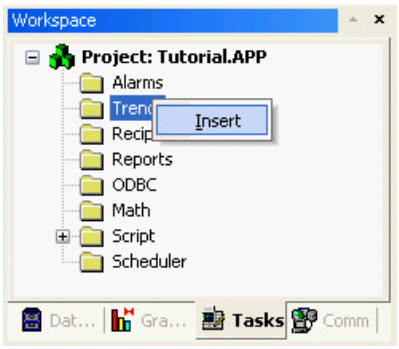
Exercise: Creating an Online Trend

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 THE 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:

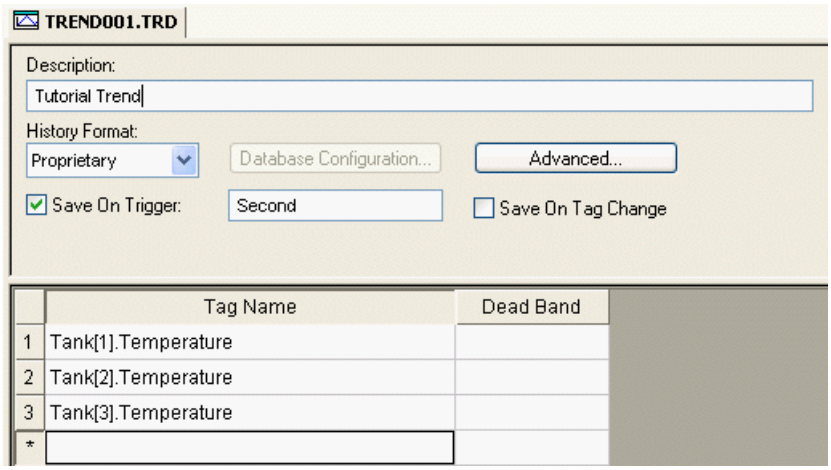
☑ Create a new Trend worksheet by right-clicking on the *Trend* folder (in the *Tasks* tab of the *Workspace*) and selecting **Insert**:



Inserting a Trend Worksheet

> ➪ **Tip:**
> 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.

☑ Configure the Trend worksheet as shown:



Configuring the TREND001.TRD Worksheet

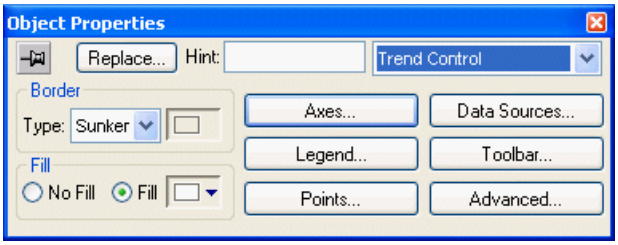
☑ Save and close the worksheet.

CREATING THE TREND ONLINE SCREEN

Next, you will create the Trend Online screen, draw a Trend Control object on the screen, and configure the object:

☑ Open the *Standard.scr* screen and save it as **TrendOnline.scr**.

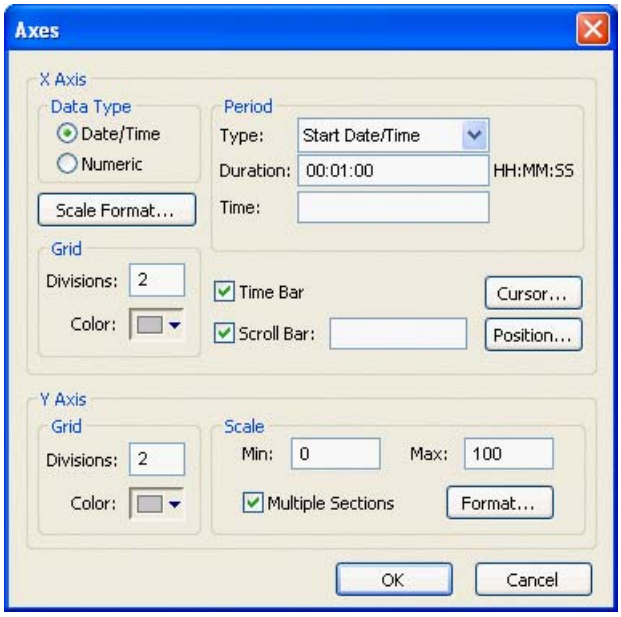
☑ Click on the **Trend Control** button, draw a *Trend Control* object on the screen (below the standard buttons), and open its *Object Properties* dialog:



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.

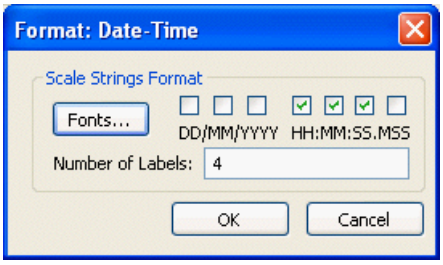
☑ Click the **Axes** button to open the *Axes* dialog:



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:

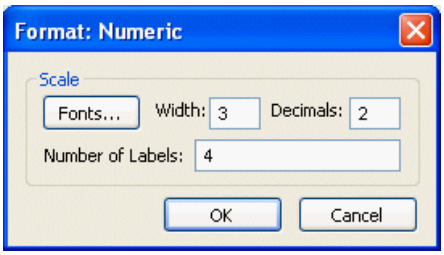


X Axis — Format: Date-Time

☑ 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.

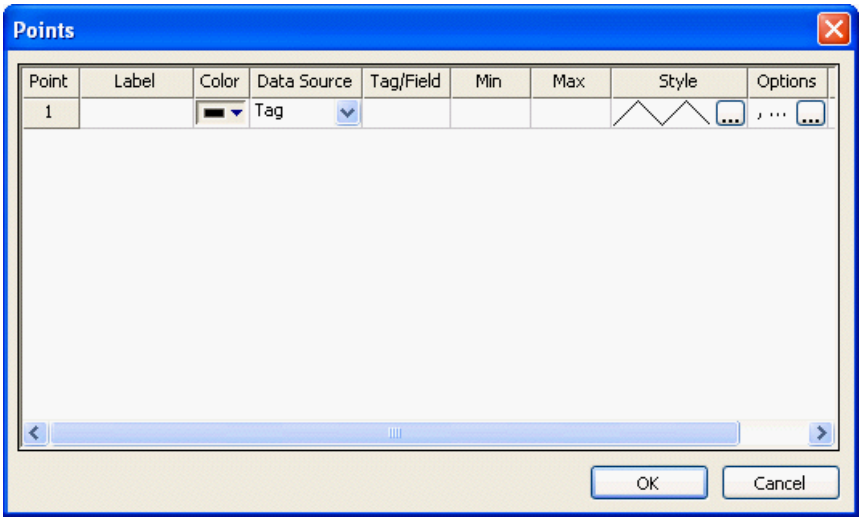
By default, multiple trend data points are each displayed in their own sections of the trend graph. However, in this exercise the data points should be displayed together in the same section, so that they can be compared as they progress.

☑ In the *Y Axis* area, click the **Multiple Sections** checkbox to deactivate it.

☑ 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

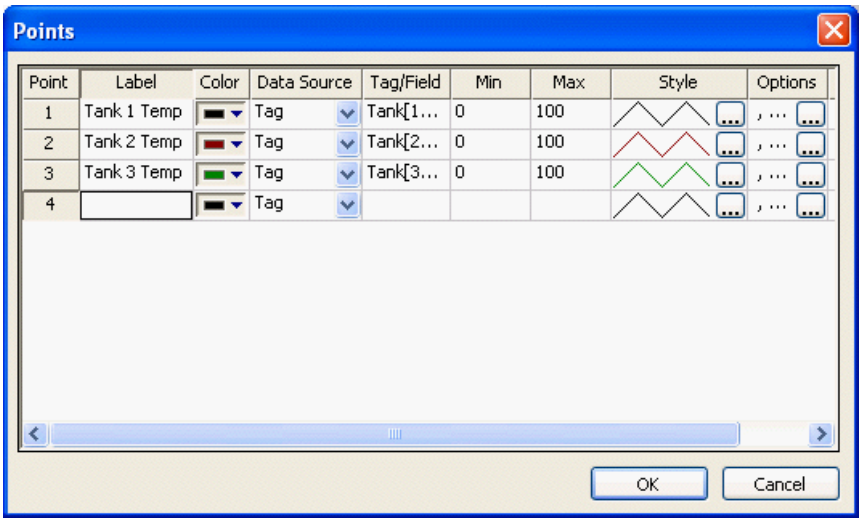
Point 2:

- **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:

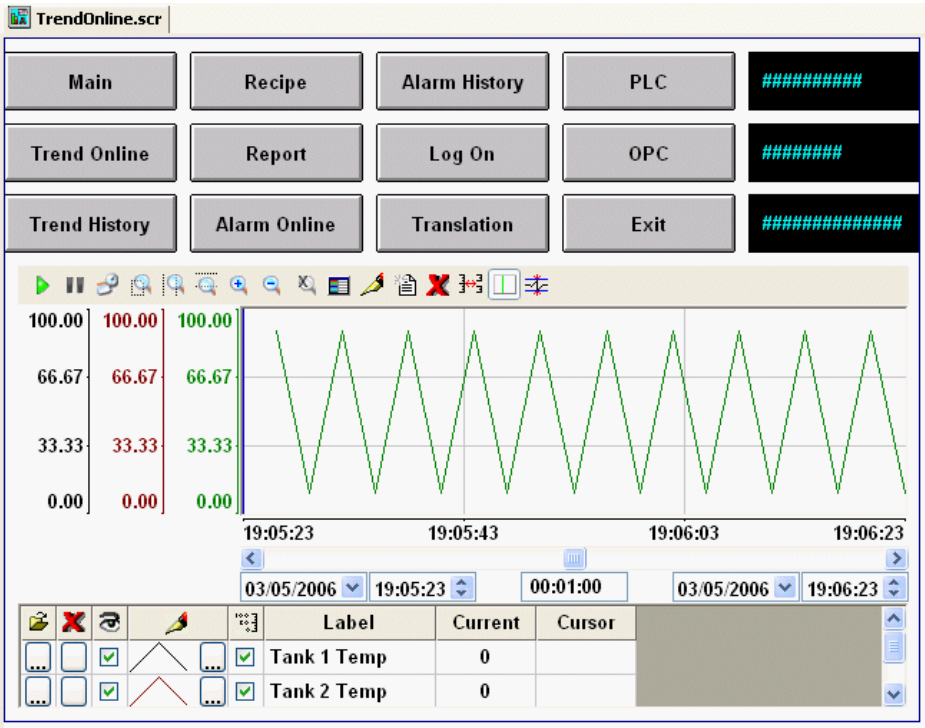


Points Dialog with Three Points Configured

☑ Click **OK** to close the *Points* dialog, and then close the *Object Properties* dialog.

☑ Save your work and verify the application (**Tools → Verify Application**).

The finished *Trend Online* screen, including the buttons and text fields created earlier, should look something like this:



Trend Online 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

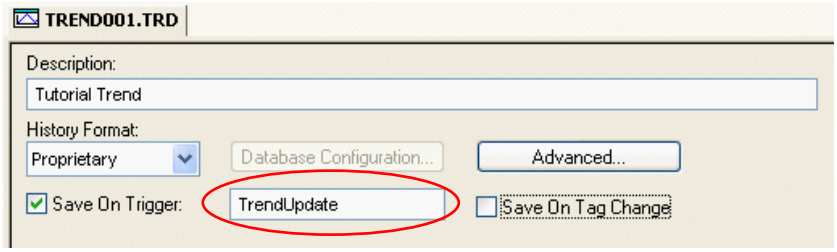
As it is now configured, the Trend worksheet saves to the history file and updates the trend graph every second. While this makes for a smooth graph, it also decreases system performance and generates extremely large history files.

You can decrease the save frequency by reconfiguring the *Save On Trigger* field (on 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, of course — the save is triggered.

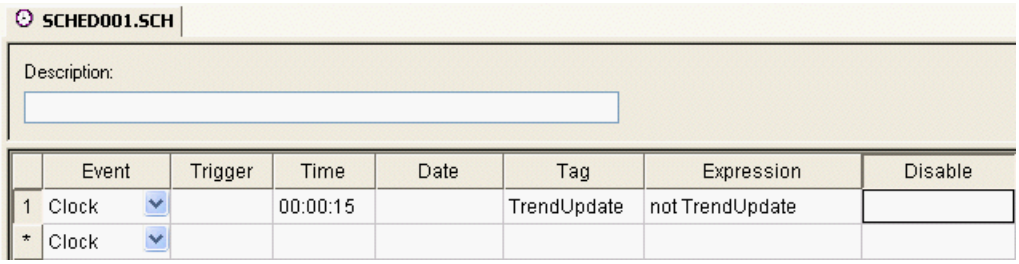
You could easily reconfigure the *Save On Trigger* field with another Internal Tag (e.g. **Minute**, **Hour**, **Month**, and so on), but this may trigger saves too infrequently to be useful.

To trigger saves at a nonstandard frequency, use the Scheduler:

- ☑ Create a new Tag in the *Application* database (review the procedure on page 5–7 if necessary) with the following specifications:
  - **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.

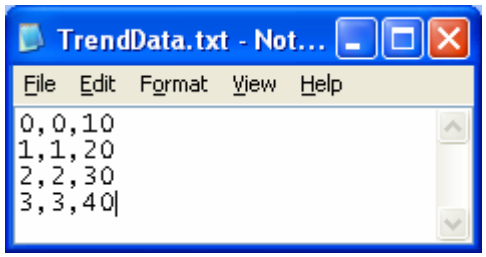
Supplemental: Using an External Text File

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.

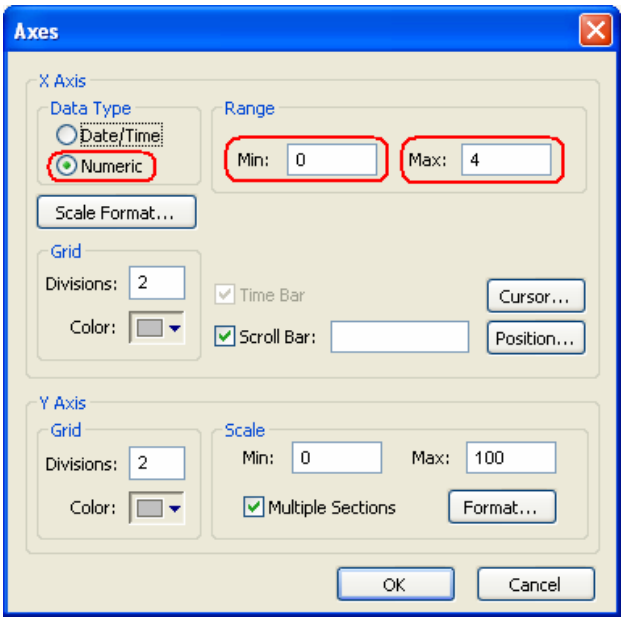
☑ The first step is to convert the data into a text file. If we adopt the comma as our separator the file will be as shown below:



Creating a Text File Using Notepad.exe

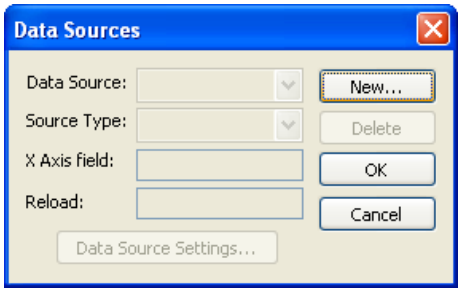
☑ We strongly recommend that you save the file in the same folder where the application is. By doing so, you do not have to specify the entire path and your application will still work, even if it is copied to a different computer.

☑ Once you have added the **Trend Control** to your screen, double click on the object to open then **Object Properties** and click on **Axis**. Change the **Data Type** of the X Axis to numeric, and set the ranges as shown in the picture below:



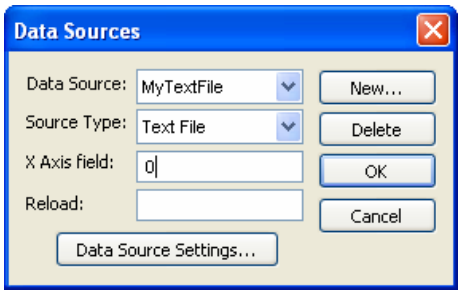
Configuring the X Axis for Numeric Data

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



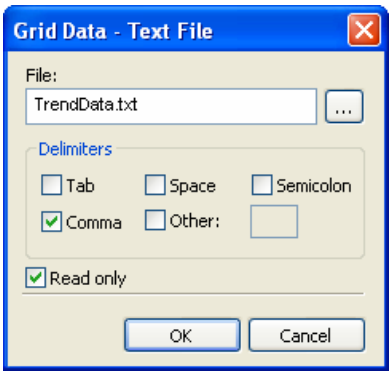
Trend Control Object Properties — Data Sources

☑ 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)

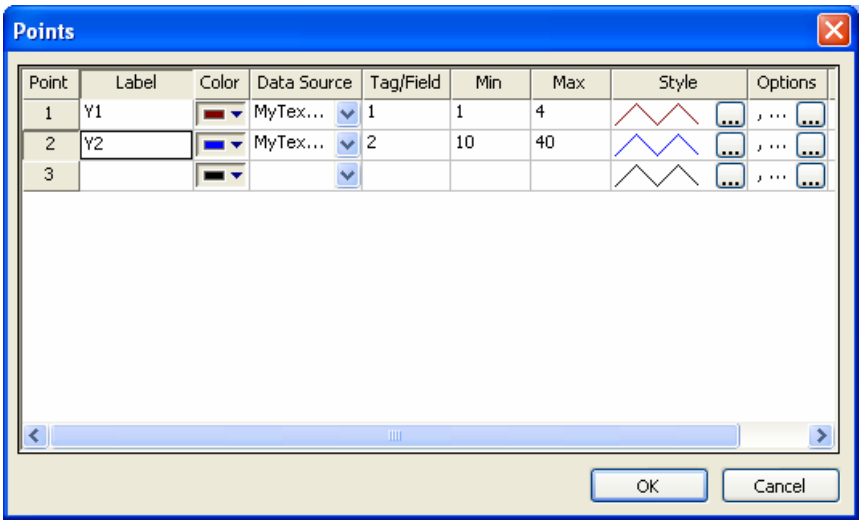
☑ On the **X Axis field** we need to indicate which column in our text file represents the X Axis. In our example we are using column zero, so enter with zero for this field, then click on the button **Data Source Settings**, the following Window will display:



Specifying the Text File Name and Field Delimiter

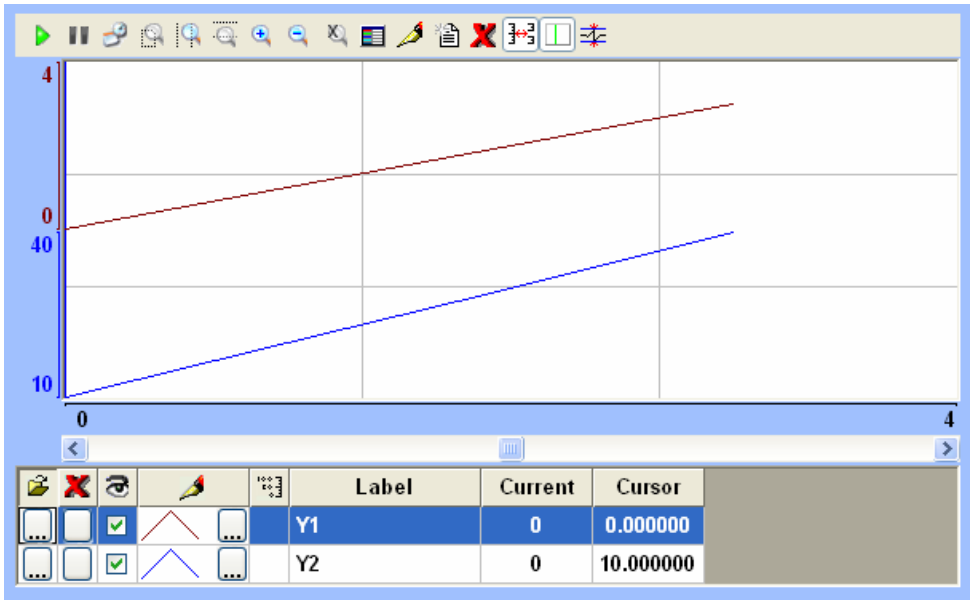
☑ If you have copied the text file to the application folder, you only have to specify the file name, otherwise, enter with the complete path where the file is located (use the browse button ... as needed). Click **Ok** on this window and **Ok** again to finish the data source configuration and close 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:



Supplemental: Using an External Database

The **Trend Control** can generate trend charts from any **Relational Database** that can be accessed through the ADO.Net technology. This Appendix illustrates how to access a Microsoft Access Database; if you are using another type of database, almost all the definitions will apply, however you will need to configure your connection on a different way. For information on how to configure other databases, please refer to the Appendixes in the Database Interface section of this manual.

Suppose that you have an access database at your C drive named “mydata.mdb” and that you want to generate a chart based on the information in the following table:

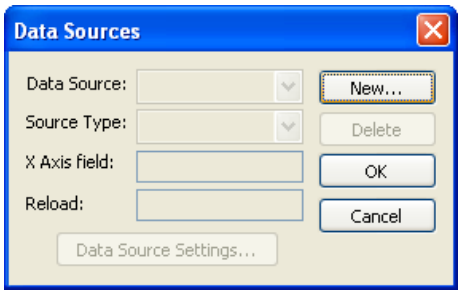
TrendData : Table

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

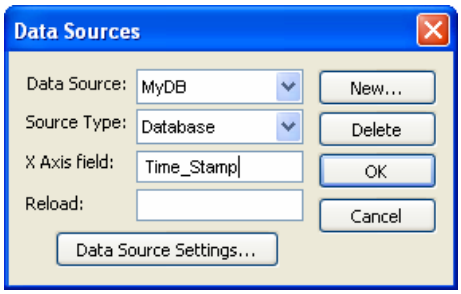
Table in a Relational Database

☑ The first step is to add the Trend Control to your screen. Now double click on the object to open then **Object Properties** and click on **Data Sources**. The following window will display:



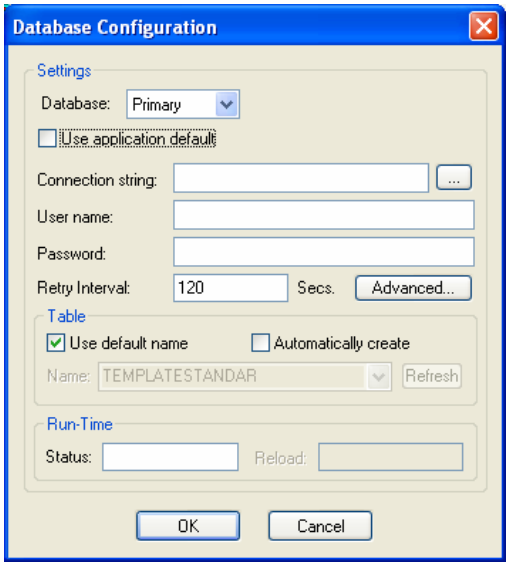
Trend Control Object Properties — Data Sources

☑ 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:



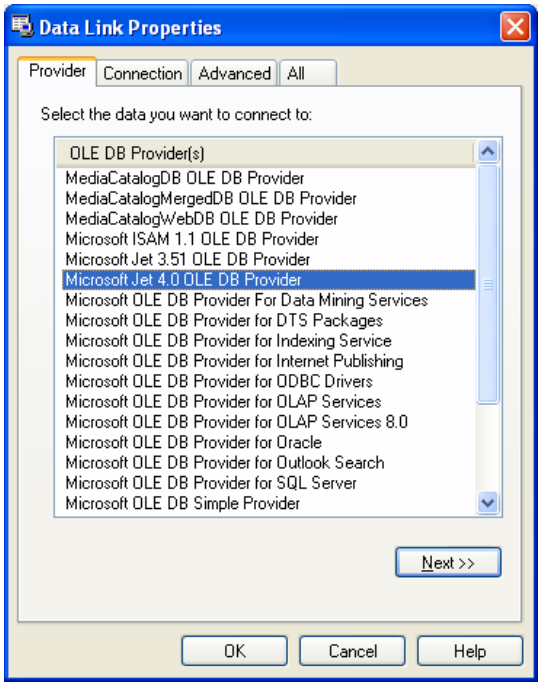
Creating a New Data Source (Database)

☑ 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:



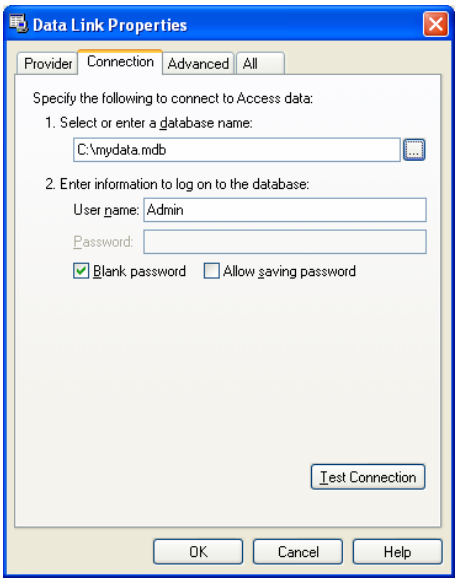
Database Configuration Window

☑ 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:



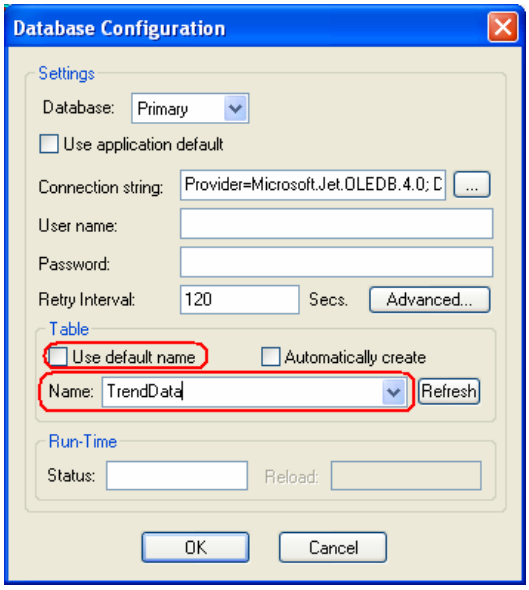
Data Link Properties — Selecting a DB Provider

☑ 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

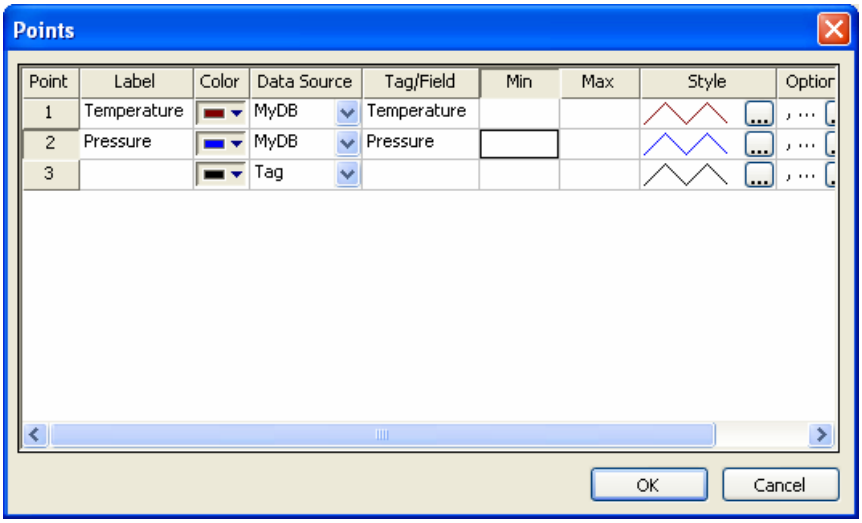
☑ Click **Ok** to finish the Connection String configuration. Now uncheck the option Use default name and select the table from your database as shown below:



Selecting the Database Table

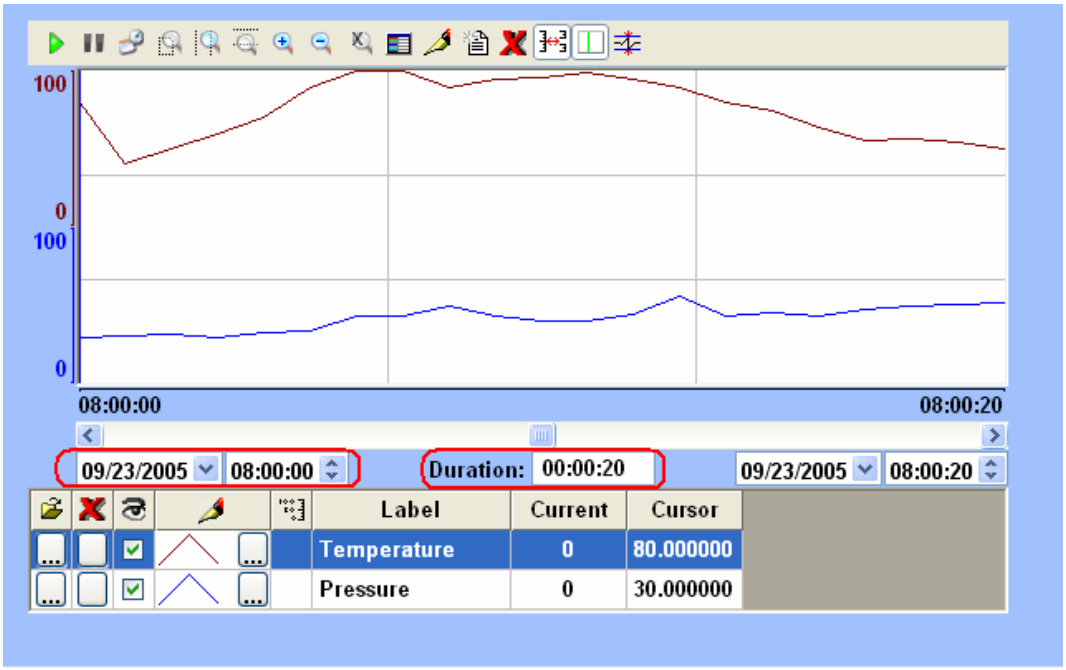
☑ Click **Ok** on this window and **Ok** again to finish the data source configuration and close the **Data Source** configuration window.

☑ 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:



Chapter 10. Configuring a Scheduler Worksheet

The *Scheduler* folder generates events with defined mathematical expressions to be executed according to the time, date, or any monitored event.

> **Note:**
> IWS sequentially increments the number that identifies the *Scheduler* worksheet for each one you create. The purpose of using different scheduler groups is to organize them.

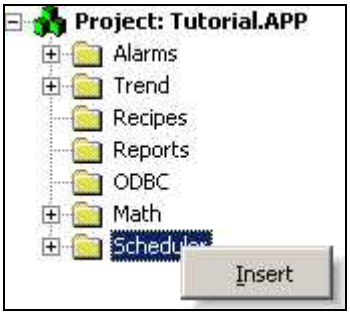
Exercise: Configuring a Scheduler Worksheet

Next, you will configure a *Scheduler* worksheet 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.

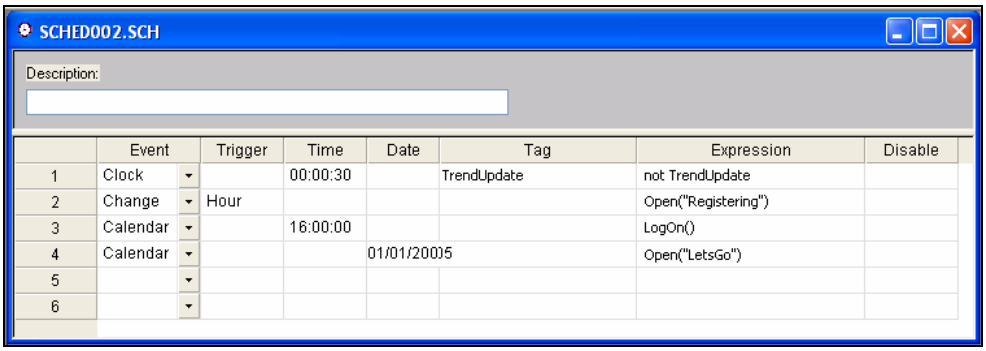
Use these steps to create a new *Scheduler* worksheet:

☑ Right-click on the *Schedule* folder:



Opening a Scheduler Worksheet

☑ Configure the *Scheduler* worksheet as follows:



	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 11. Creating Recipes

InduSoft Web Studio allows you to publish real-time, dynamic, and animated recipes and reports. You can then import or export these recipes and reports in a variety of formats, including *XML* (Extensible Mark-up Language).

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.

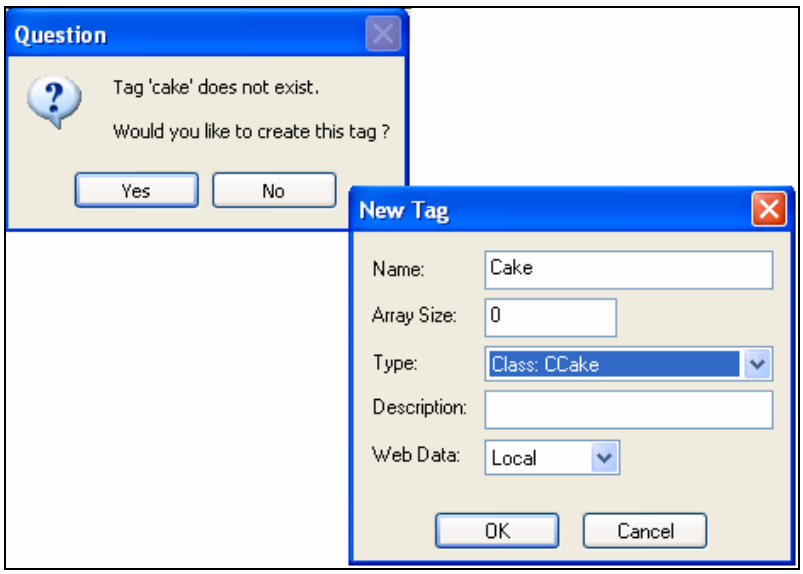
☑ Create a new Class (in the *Database* tab of the *Workspace*) named **CCake** and configure it as shown:

Class: CCake

	Name	Type	Description
1	Sugar	Integer	
2	Fruit	Integer	
3	Milk	Integer	
4	Flour	Integer	
5	Yeast	Integer	
*		Integer	

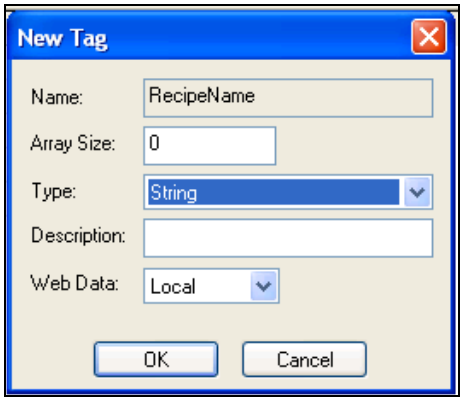
Creating the CCake Class

☑ Next, create a **Cake** tag using the **CCake** type.



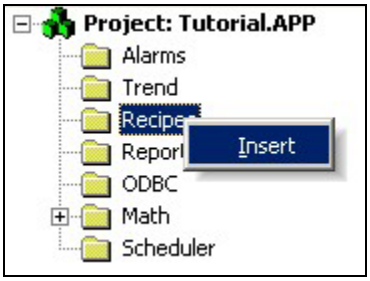
Creating a Cake Tag

☑ Create a RecipeName tag using type STRING (this tag is not a CLASS type) to store the input file name used in the **Recipe** task:



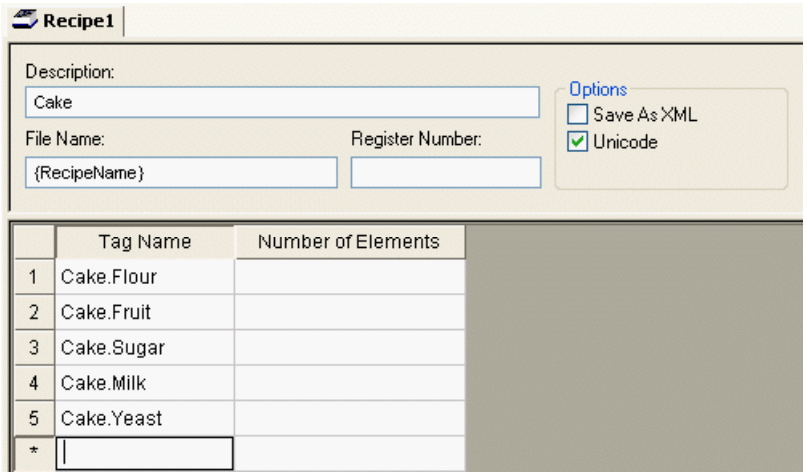
Creating RecipeName Tag

☑ 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.



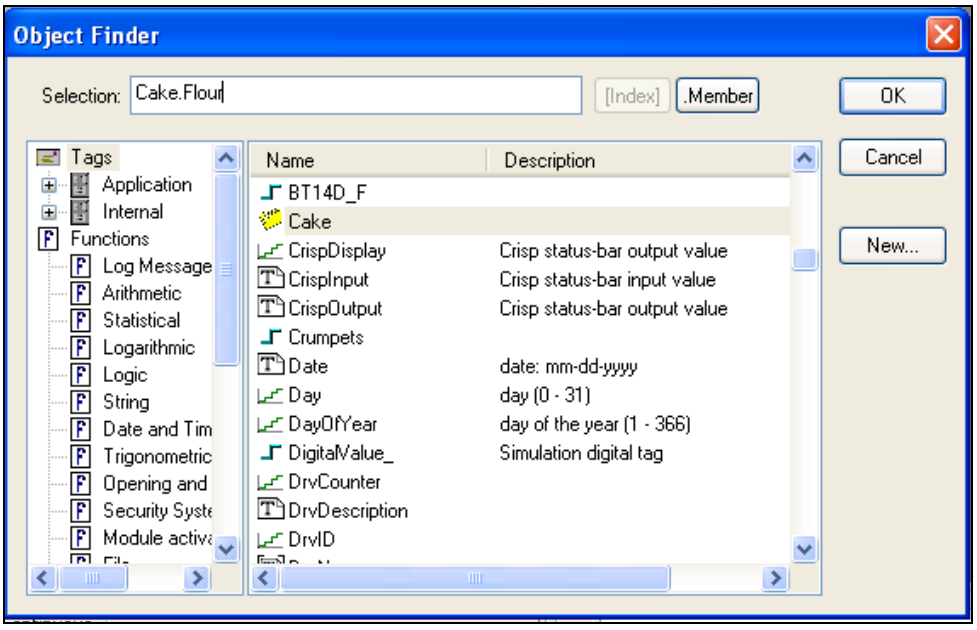
Creating a Recipe Worksheet

☑ 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

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**.



Note:
Remember that the syntax used to access a value from a class tag is: ***<tag_name>.<member_name>*** (for example, **Cake.Sugar**, **Cake.Fruit**, etc.).

– **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. (*No longer used*)
- **Number of elements** column: Specifies how many array tag positions are in use.

  So, 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 (for example, **Tag**).
- **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.

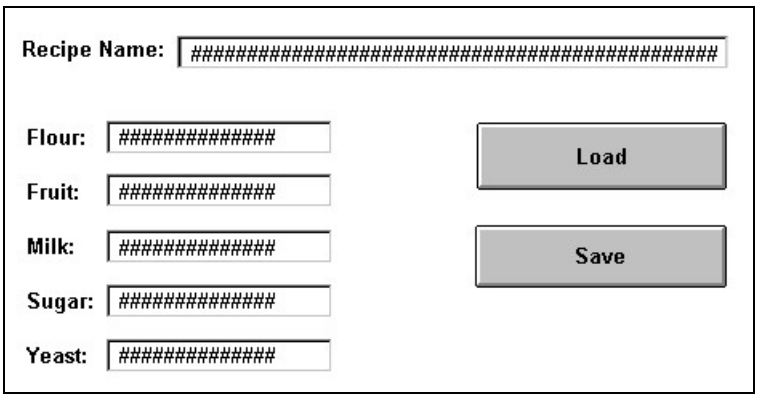
☑ Proceed to creating a *Recipe* screen.

Creating a Recipe Screen

Use the following steps to create a *Recipe* screen.

☑ Open the *Standard.scr* screen you created as your template and save it as **Recipe.scr**.

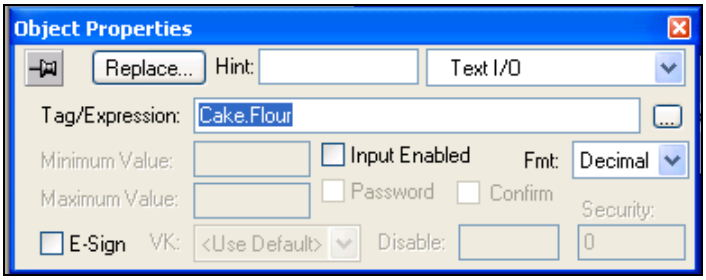
☑ Open the **Symbol Library** (or use your toolbars) and use it to draw the objects shown in the following figure. Also, arrange these objects as shown.



Creating Objects for Your Screen

☑ Double-click on each of the **############** texts to open their *Object Properties* dialogs. Select **Text I/O** from the drop-down menu and then type the tag names and *Cake* tag member names into the **Tag/Expression** fields.

For example: For the **Flour: ############** text, type **Cake.Flour** into the **Tag/Expression** field as shown.



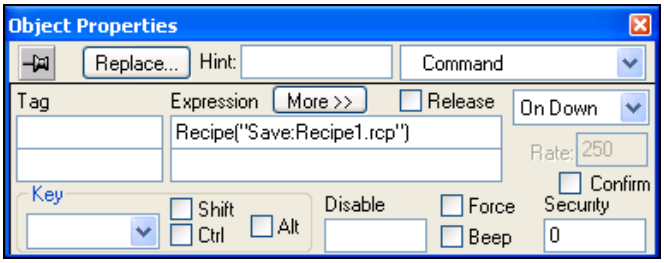
Cake.Flour Tag/Expression

☑ Allow the **Input Enabled** check-box to remain checked (*enabled*).

☑ Associate the RecipeName string tag to the **Recipe Name** field.

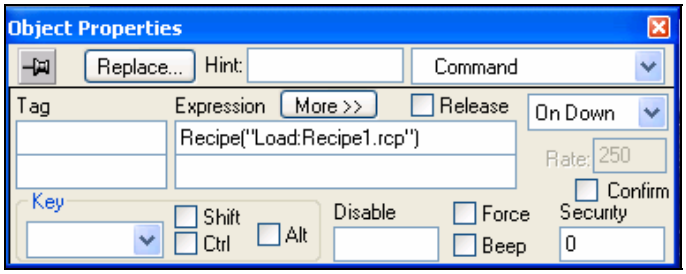
☑ Configure the **Save** and **Load** buttons:

Double click on the **Save** button. When the *Object Properties* dialog displays, select **Command** from the drop-down menu and configure the following:



Configuring Save Button

☑ Double click on the **Load** button. When the *Object Properties* dialog displays, select **Command** from the drop-down menu and configure the following:



Configuring Load Button

Testing the Recipe Laboratory

Now, to test your new *Recipe* screen:

- ☑ Type a name into the **RecipeName** field and add some values to its ingredients. Save it.
- ☑ Type a different Recipe name into the field and add some ingredient values. Save this Recipe also.
- ☑ Now, type the name of the first recipe and click the **Load** button to load it.

Continue playing with this *Recipe* screen to test it. When you are finished, continue to the next section to create a Report.

Chapter 12. Creating Reports

Creating reports with the Report tool is very easy — no other programming tools (such as Visual Basic) 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 print a report, use an InduSoft Scripting Language function anywhere an expression is allowed.

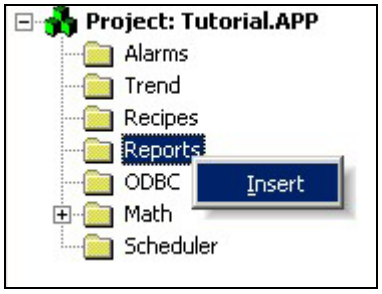
Exercise: Creating Reports

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:

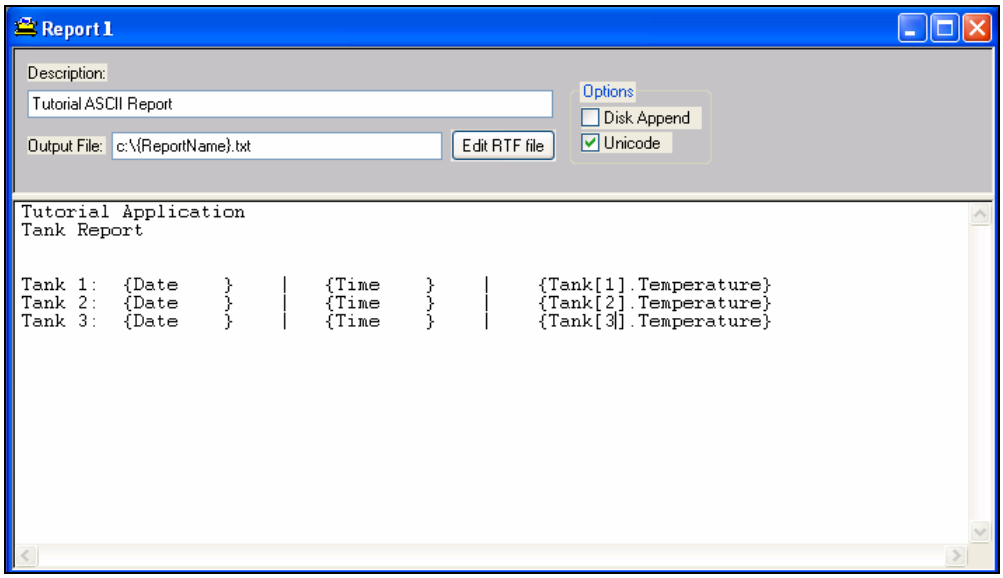
☑ In the *Workspace*, right-click on the *Reports* folder to create a *Report* worksheet. You will use this worksheet to create the body of your report file.



Creating a Reports Worksheet

☑ Configure the *Report* worksheet as follows:

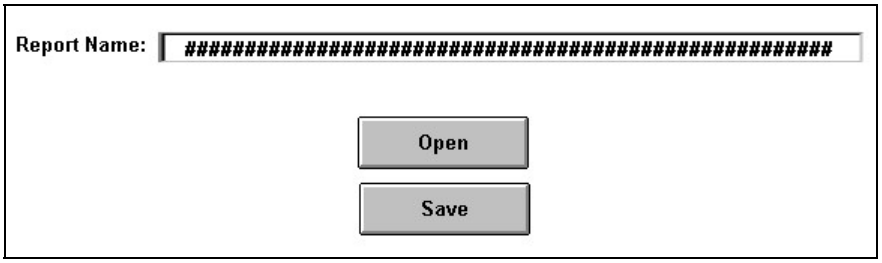
- As in the Recipe exercise, the report (including the report name) will be created in an output file. To name your report, type **ReportName.txt** inside curly brackets as shown in the following figure.
- Ignore the **Disk Append** check-box for now (this option will be explained in the next example).



Report1.rep Worksheet

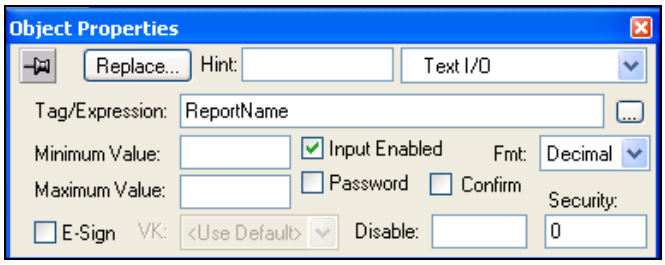
☑ Save the *Report* worksheet as **Report1.rep** (*default name*).

☑ Use the **Symbol Library** and your toolbars to create the following *Report Screen*.



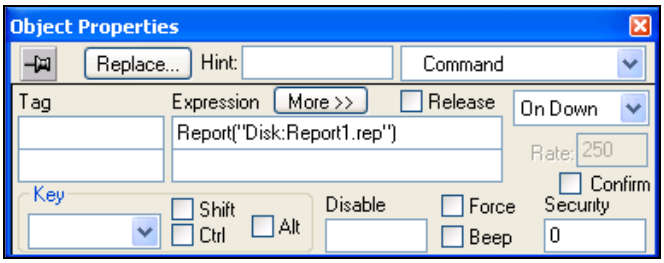
Creating the Report Screen Objects

☑ Double-click on the **##########** text, and when the *Object Properties* dialog displays, configure the **Text I/O** property as follows:



Configuring the Text I/O Properties

☑ Double-click the **Save** button and configure the **Command** property as follows:

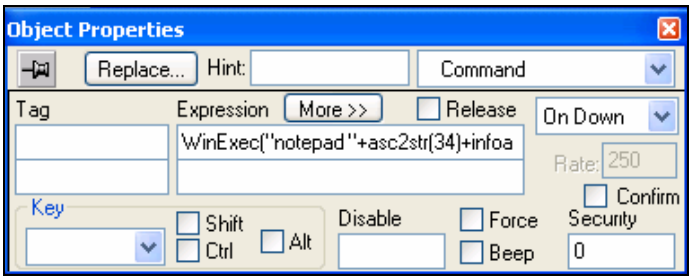


Configuring the Save Button

☑ Double-click the **Open** button and configure the **Command** property as follows:

WinExec("notepad "+asc2str(34)+infoappdir()+"\"+reportname+".txt"+asc2str(34))

> **Note:**
> There is a space between the **d** in **notepad** and the quotation mark that follows.



Configuring the Open Button

This command uses the Microsoft Windows *NotePad* program to display the ASCII Report you just created. To test your new *Report* screen, type a name into the **Report Name** field and try using the **Save** and **Open** buttons.

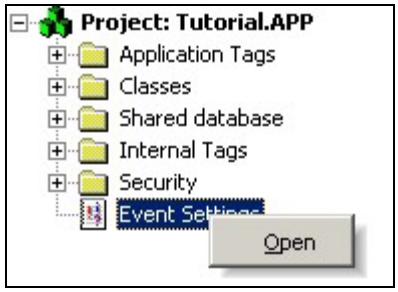
34	= ASCII code for " (quotes)
asc2str	= return a string based on the ASCII code
infoappdir	= current folder where the application is

Chapter 13. Events

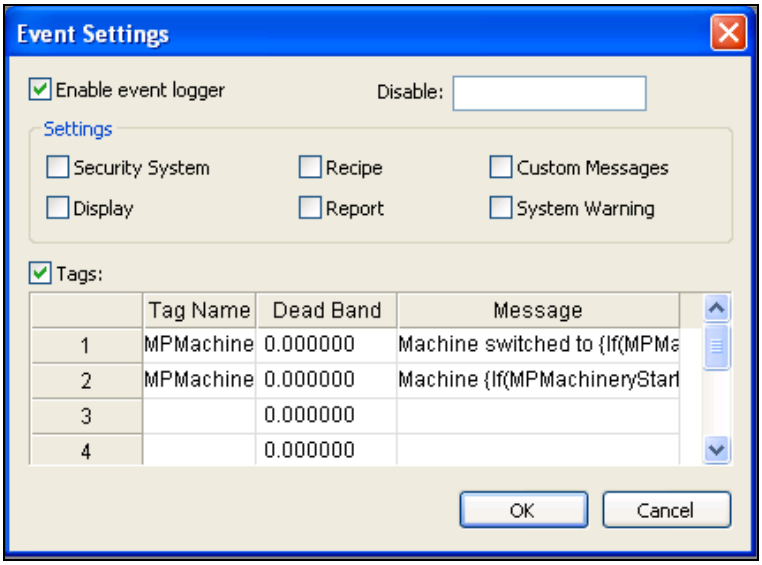
Use the steps to configure the event-retrieval feature.

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** text box: Type a tag into this field. If the tag value is other than 0 (*false*), InduSoft Web Studio automatically disables the *Event Logger*.
- **Security System** check-box: Enable (*check*) this box to include security system events in the historic event file. IWS logs the following security system events:

  - Log On / Log Off users
  - User created/removed using the **CreateUser()** or **RemoveUser()** functions
  - User blocked/unblocked using the **BlockUser()** or **UnblockUser()** functions

- User blocked by the security system after several attempts to enter an invalid password
- Password expired
- Password modified
- Invalid Log On attempt

☑ **Display** check-box: Enable (*check*) this box to include screen Open and Close events in the historical event file.

☑ **Recipe** check-box: Enable (*check*) this box to include recipe load, save, init, and delete events in the historical event file.

☑ **Report** check-box: Enable (*check*) this box to include *reports saved to disk* or *send to printer* events in the historical event file.

☑ **Custom Messages** check-box: Enable (*check*) this box to include events generated by the **SendEvent(strEvent)** function in the historical event file.

☑ **System Warning** check-box: Enable (*check*) this box to include general system warnings (such as **Division by zero, Attempted to access invalid array index**, and so forth) in the historical event file. IWS logs the following system warning events:

- Errors that occur when sending alarms by email
- Tag was blocked/unblocked
- Division by zero
- Connection/Disconnection of the remote security system

☑ **Tags** check-box: Enable (*check*) this box to enable and log tag changes in the historical event file. Configure the tags you want to log in the Tags table as follows:

- **Tag Name** column: Type the name of the tag you want to log in the event file.
- **Dead Band** column: Type a value for comparing and filtering acceptable changes.
- For example, if you specify a **Dead Band** value = 5 for a tag value = 50 and the tag value changes to 52, the system will not register this variation in the event log file, because the variation is less than 5. However, if the tag value change is equal to or greater than 5, the system will save the new value to the history file.
- **Message** column: Type a string (message) related to this tag change. You can specify tags in messages using the **{tag name}** syntax.

The **Tags** parameter can be useful if you want to generate a log file of events that are not necessarily alarm conditions (for example, **Motor On**, **Motor Off**, and so forth).

Chapter 14. Using the Translation Tool

You can use the *Translation Tool* to change translate texts on buttons or in any other text field from one language to another. All you have to do is create *Translation* worksheets and apply the translation functions.

The translation function searches your application for all the texts typed in the **Original** column of a *Translation* worksheet and than changes that text into the text typed in the **Translation** column of that worksheet. Also, the *Translation Tool* ignores any text that has not been entered into the *Translation* worksheet.

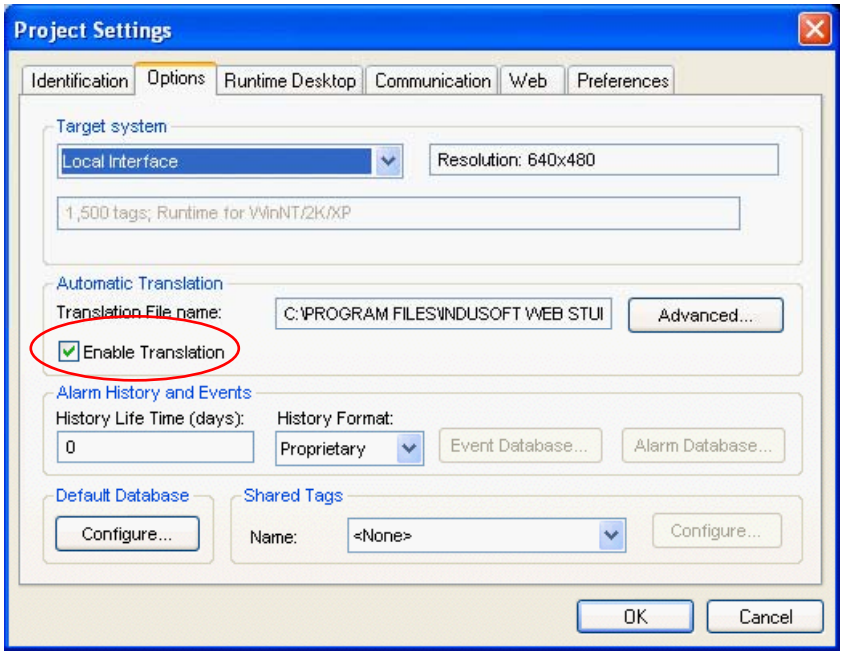
In the next exercise you will set up a *Translation* worksheet and use the translation function to translate some text.

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.

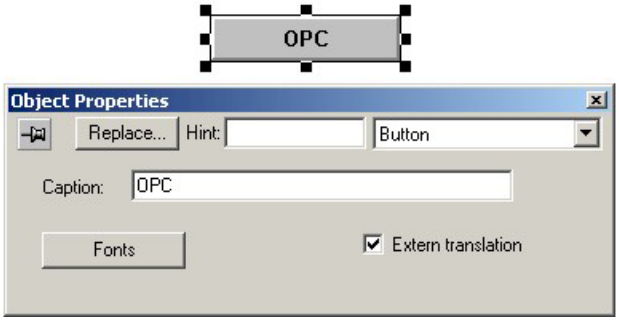
Configuring Object Properties for Screen Objects

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:

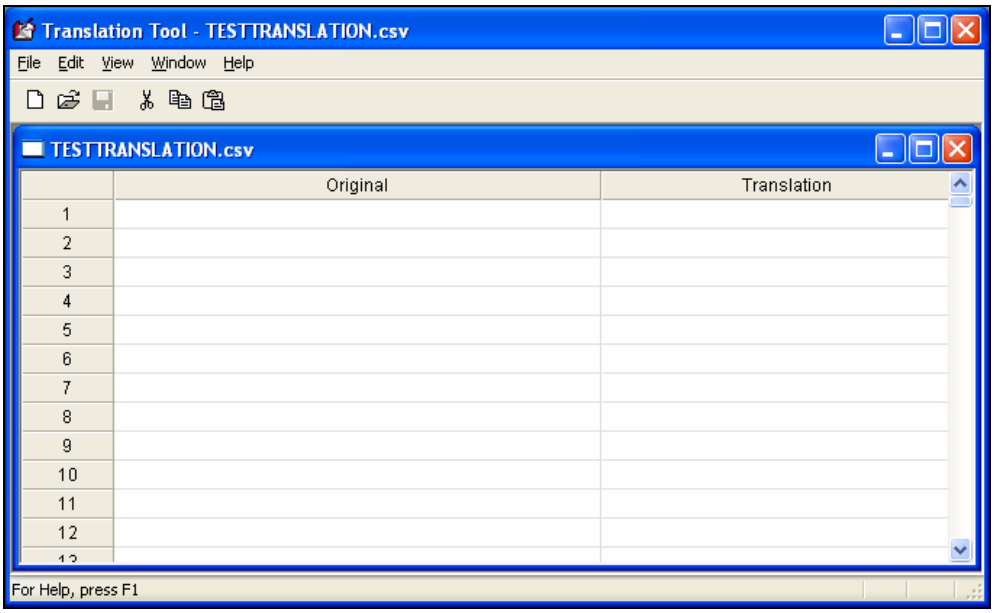


Translation Enabled for the OPC Button

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).

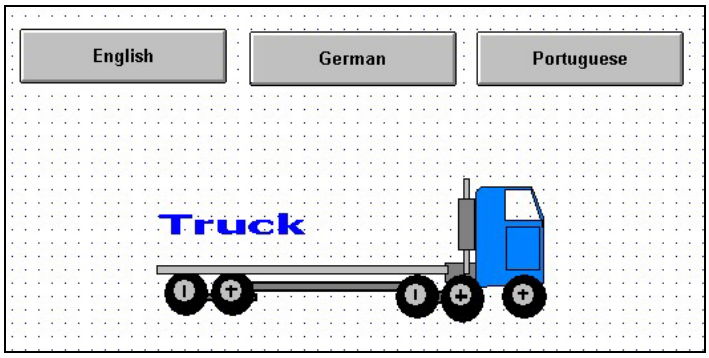
> **Note:**
> For legacy reasons, the *Translation Tool* supports translation files with the extension TRA. The translation files saved with this extension use the pipe character ("|") instead of the comma character (",") to separate the text between columns.

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**):

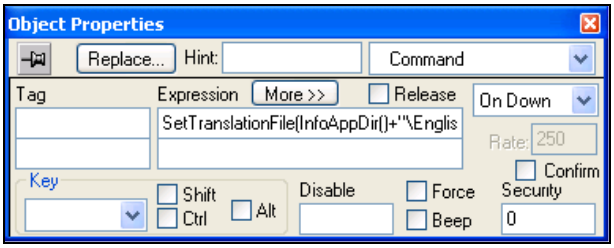


New Translation Screen

☑ 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 15. 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.

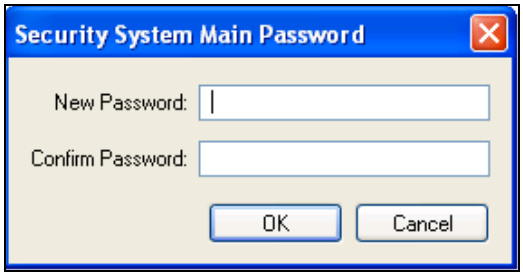
Security System Dialog

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:

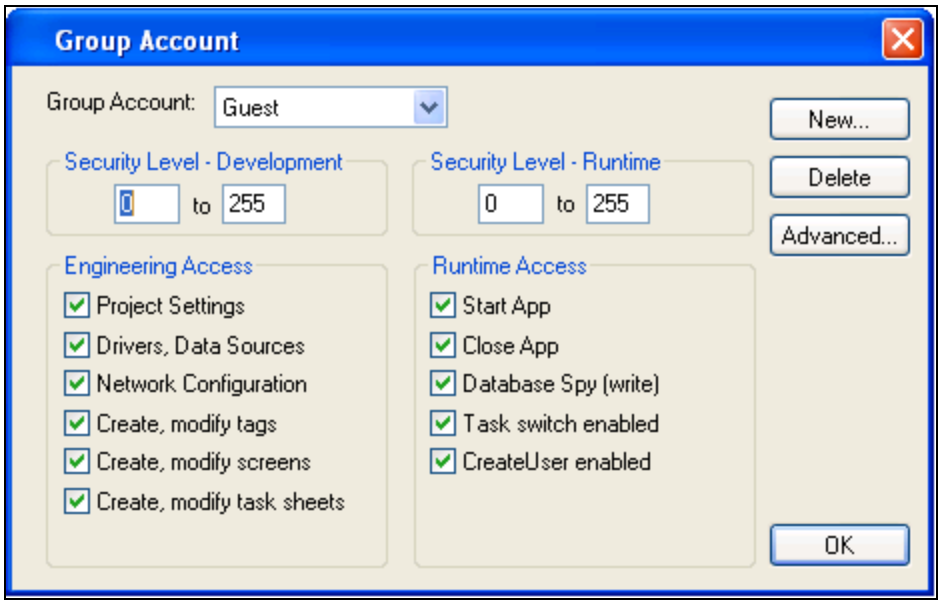
- **New Password** field: Type a password.
- **Confirm Password** field: Retype the same password to confirm it.
  If the passwords are different, you will be prompted to type it again.

> **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.

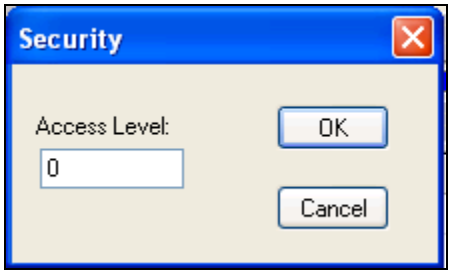
> ➲ **IMPORTANT!**
> You also can set the security level for documents (such as worksheets and screens) to protect them in the development environment.

  – **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*

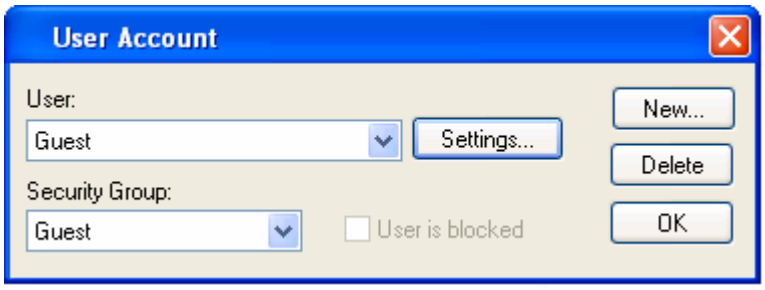
Consequently,

- 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.)

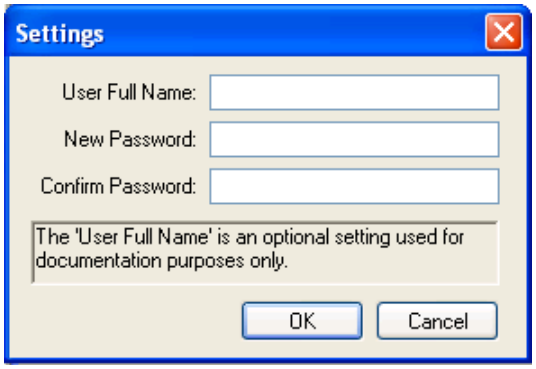
> **Note:**
> Alternatively, you can access the *User Account* dialog from the *Users* folder located in the *Security* folder on the **Database** tab, or by selecting **Insert →User** from the main menu bar.



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

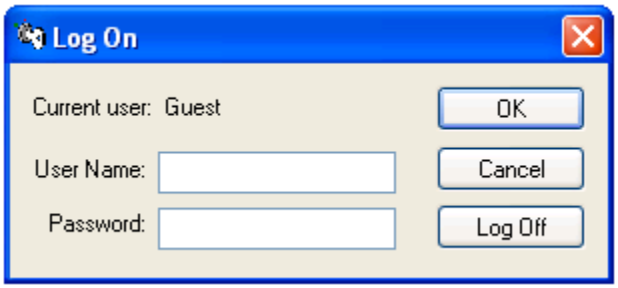
After initializing, a user is logged on as a *Guest* user (by *default*). If no other user logs on or the current user logs off, the *Guest* user is automatically logged on.

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

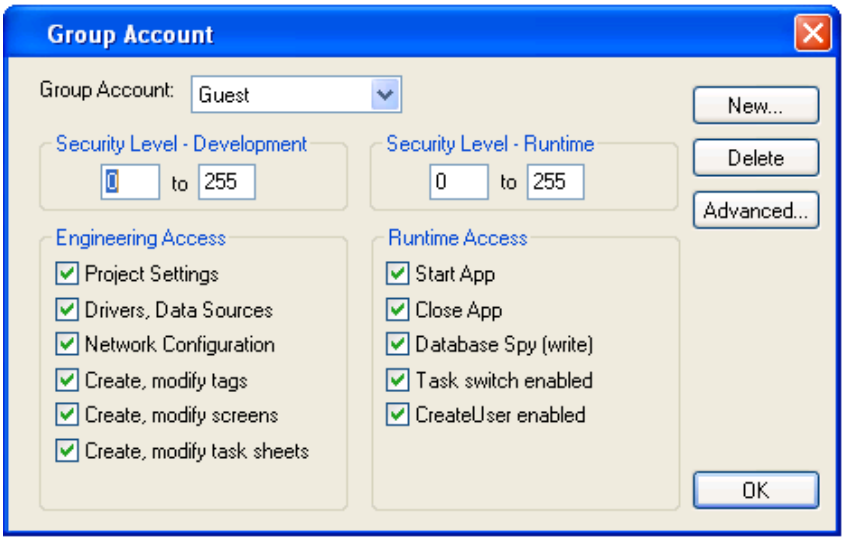
Use the following procedure to create these groups:

☑ 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

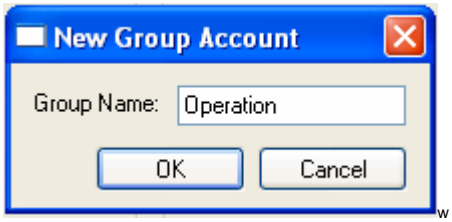
The *Group Account* dialog displays:



Group Account Dialog

Remember, you cannot delete the default group called *Guest*, so you must create a new group, as follows.

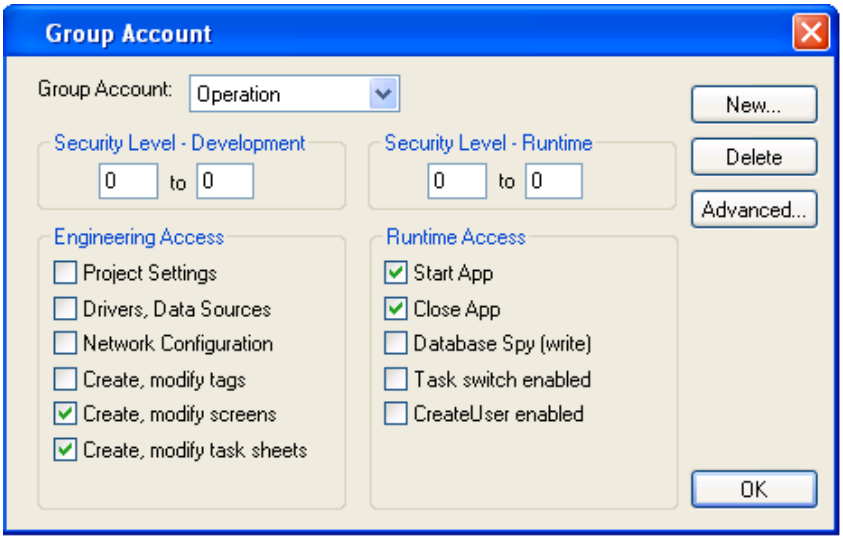
☑ Click the **New** button, and when the *New Group Account* dialog displays, type a group name into the field provided. (For this class, type **Operation**.) Click **OK** to close the dialog.



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.

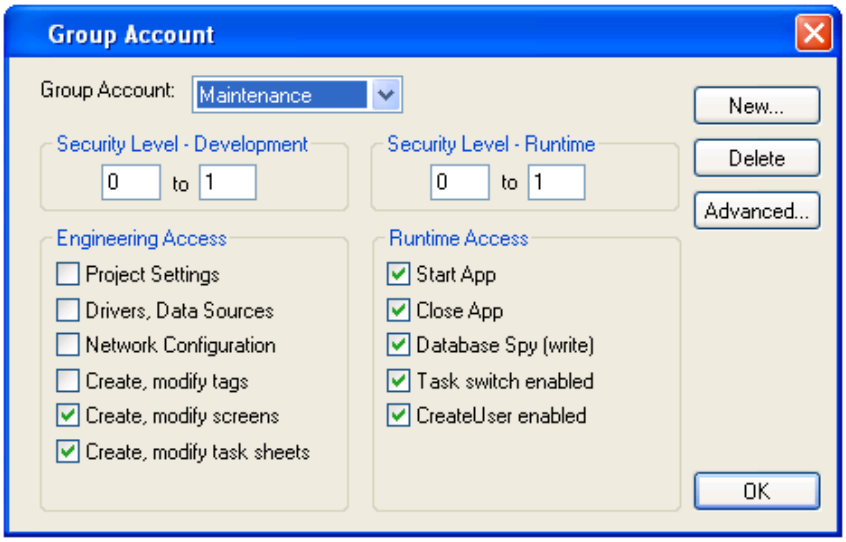
☑ Configure the access rights for this group as shown:



Operation Access Rights

☑ Click the **New** button again and create the Maintenance group. Click **OK** to close the *New Group Account* dialog.

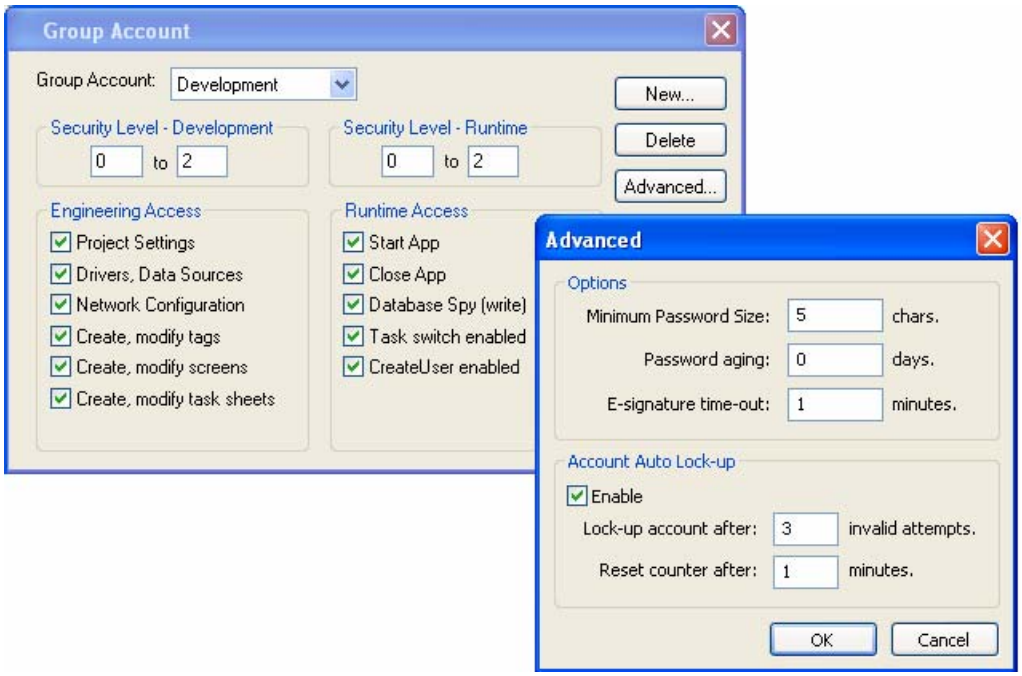
☑ Select **Maintenance** from the **Group Account** combo-box and configure the access rights for the group as follows:



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.

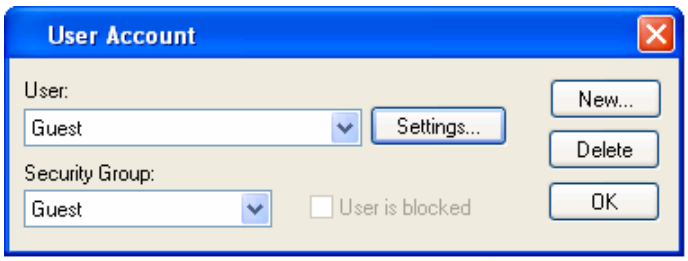
Next, you must create new users and associate these users to the group accounts you just created. Use the following steps:

☑ 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

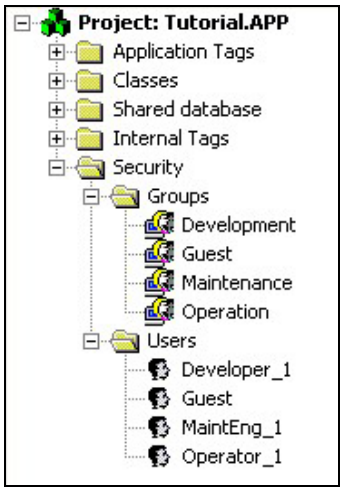
☑ Reopen the *User Account* dialog once more and add the last user as follows:

- 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

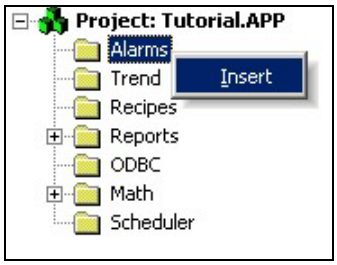
In this exercise, you will create alarm groups using the following procedure:

☑ Create the following tags:

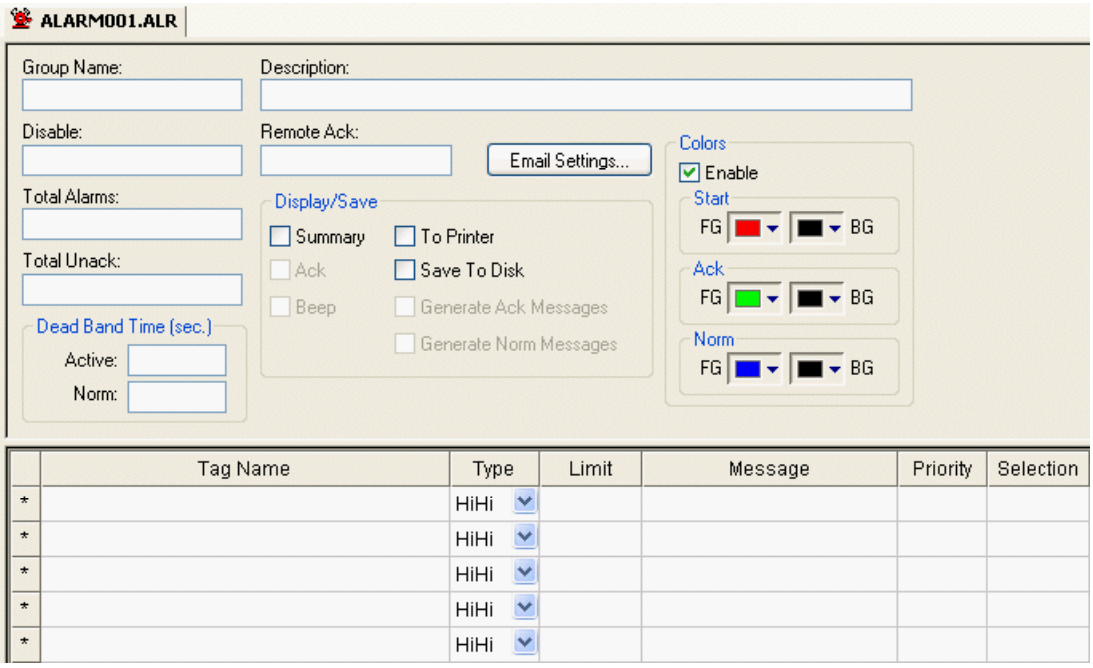
Tag Name	Size	Type
Alarm_Sel	2	String
View	2	Integer
Filters	2	Boolean

☑ 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.

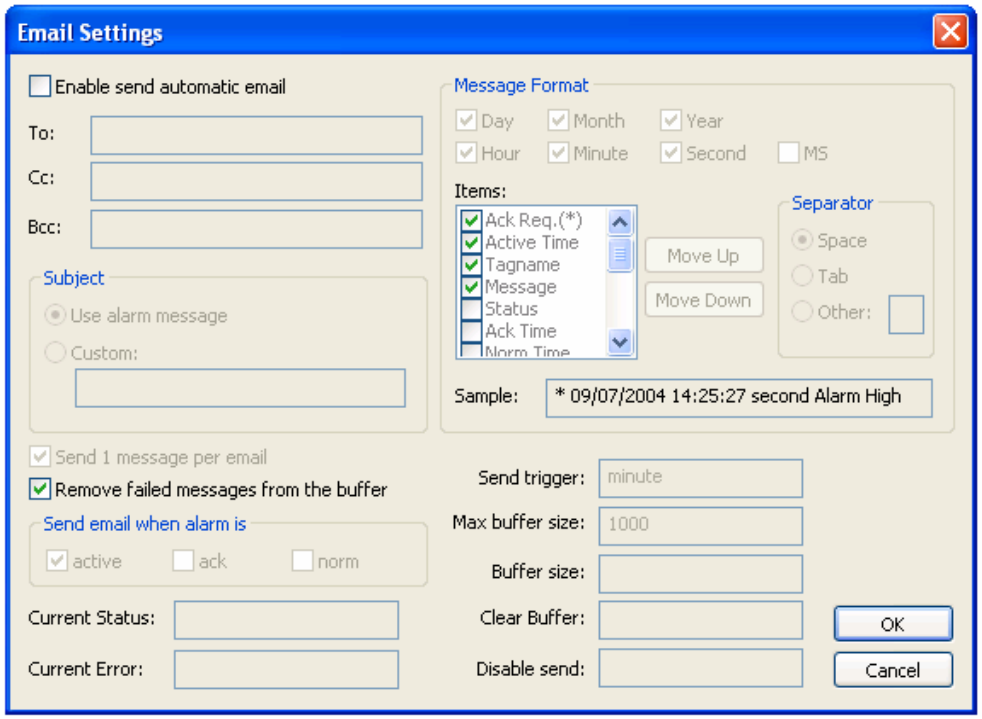
  You must add a tag to this field.

  – 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).

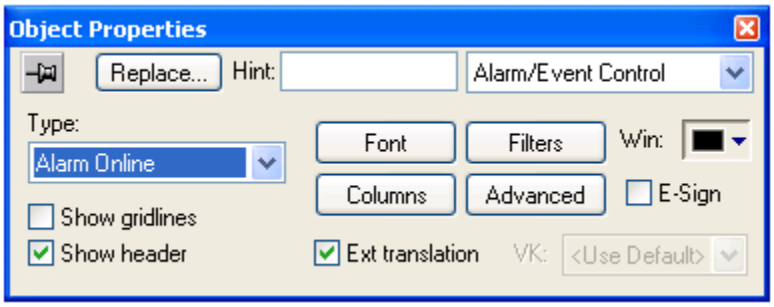
- **Alarm summary**: When you enable the alarm history file for a group, IWS saves the file as **AL*yymmdd*.ALH** in the application's **\app\ALARM** directory.

  Where ***yymmdd*** refers to the year, month, and day the file was created.

Exercise: Creating On-Line Alarm Screens

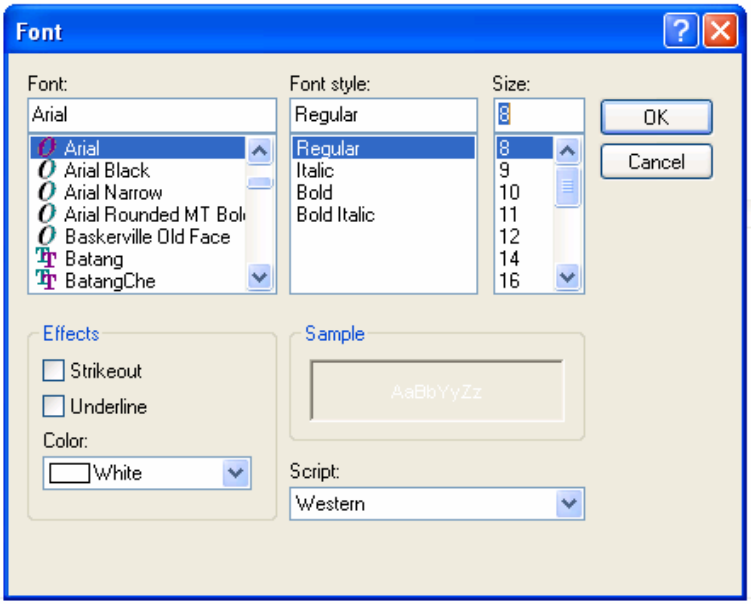
To begin this exercise:

- ☑ 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:



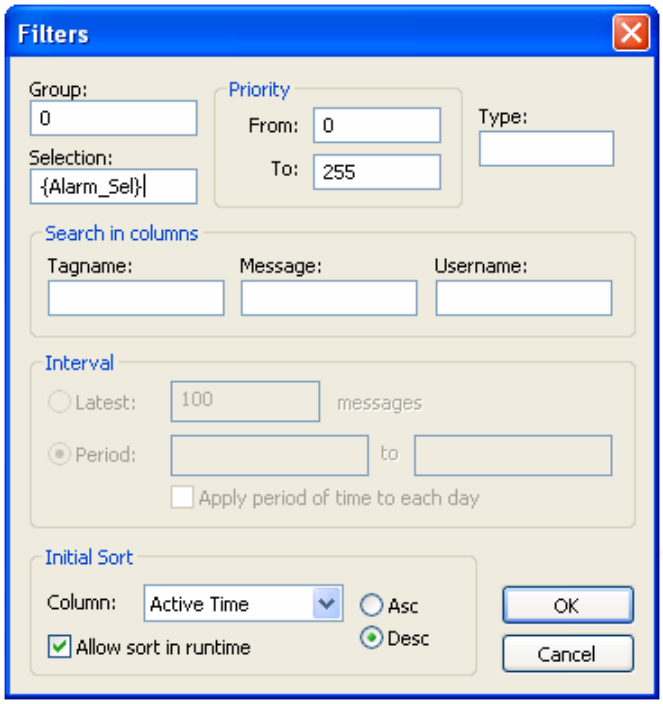
Object Properties: Alarm

- ☑ Configure the *Alarm* object. (Being sure to select the **Alarm Online** type and set the font parameters to Arial, 8 point, and White.)



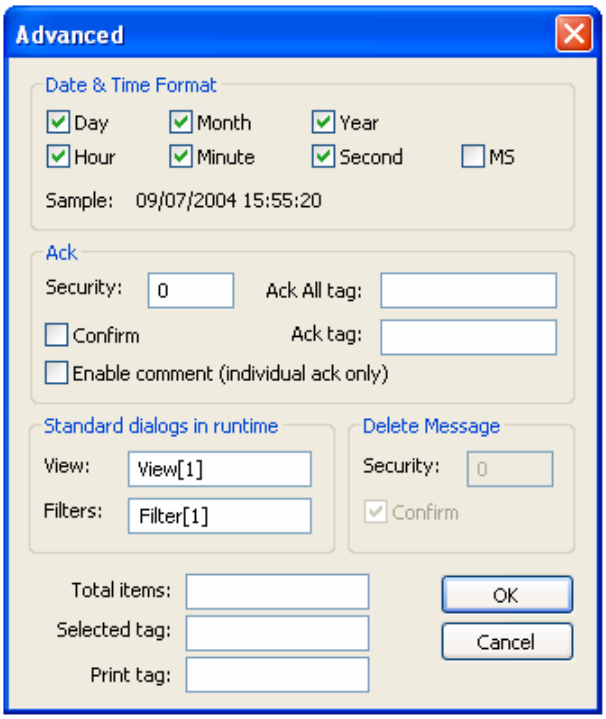
Configuring the Alarm Object

☑ Click the **Filters** button on the *Object Properties* dialog to configure the alarm filter as follows:



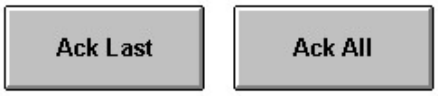
Configuring the Alarm Filter

☑ Click the **Advanced** button on the *Object Properties* dialog to configure the alarm Advanced Settings as follows:



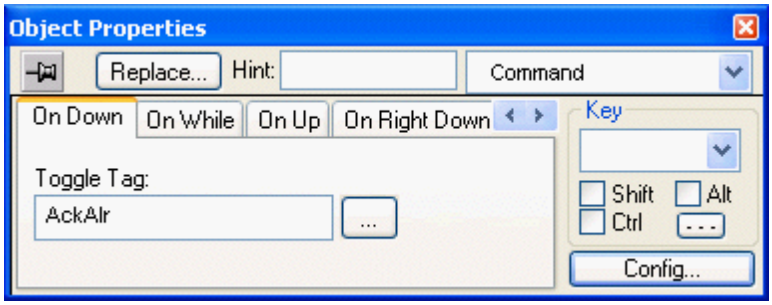
Configuring the Alarm Advanced Settings

☑ Create two buttons on your screen to acknowledge these alarms:



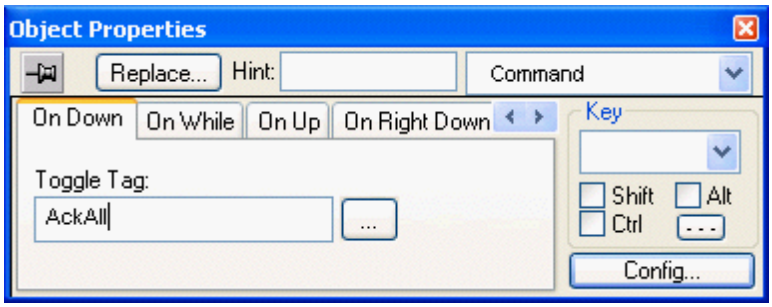
New Alarm Buttons

– 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:



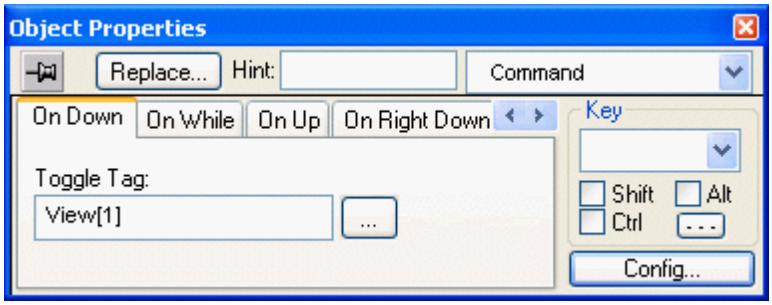
Configuring the Ack All Button

☑ 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:

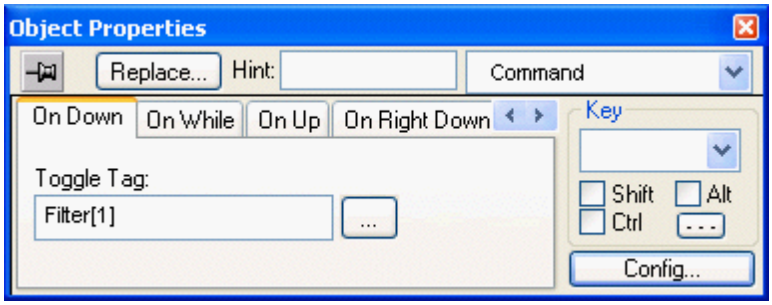
Alarm Buttons

– 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:



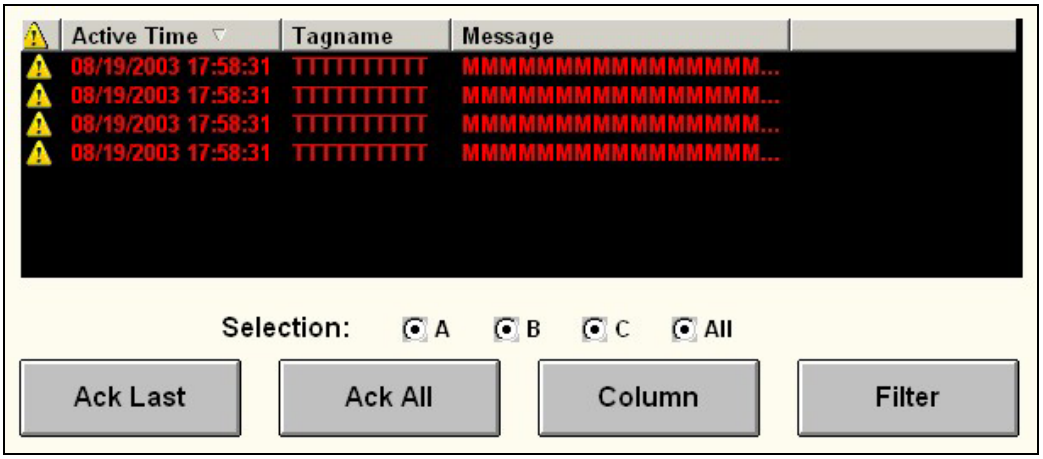
Configuring the Column Button

– 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:



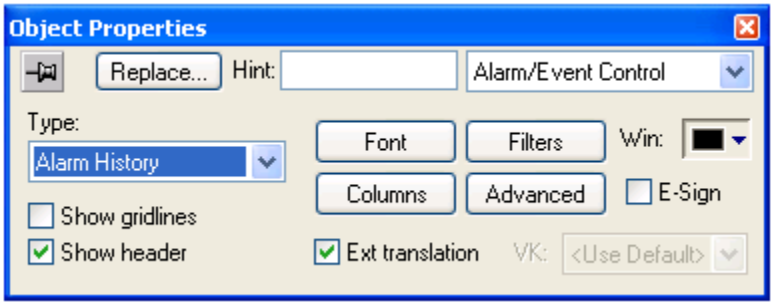
Finished Online Alarm Screen

☑ Save the screen and continue to the next section.

Exercise: Creating the Historical Alarm Screen

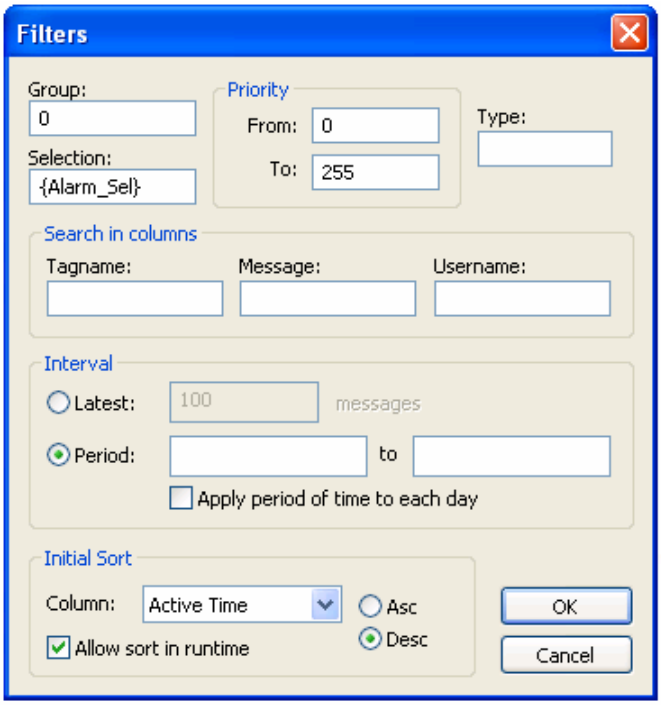
To create a *Historical Alarm* screen, use the following procedure:

- ☑ 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.



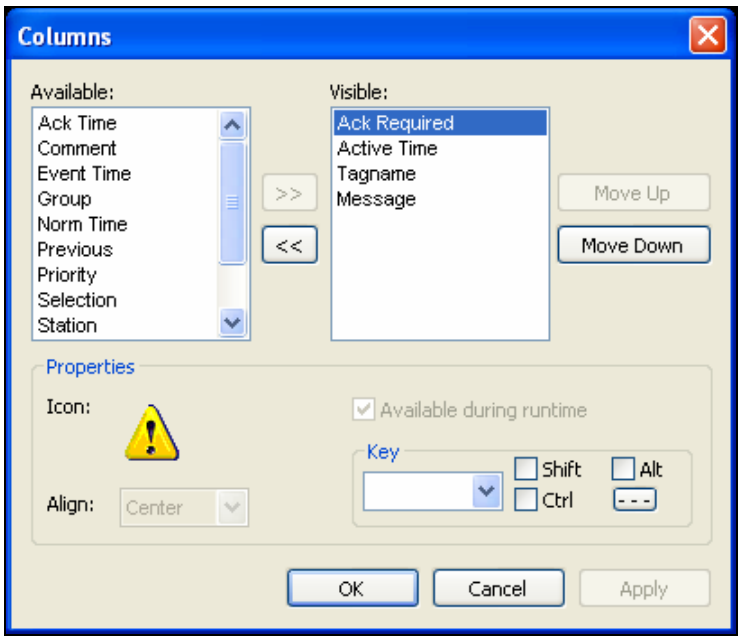
Alarm History Object Properties

- ☑ Click the **Filter** button and use the following figure to configure the filter**:**



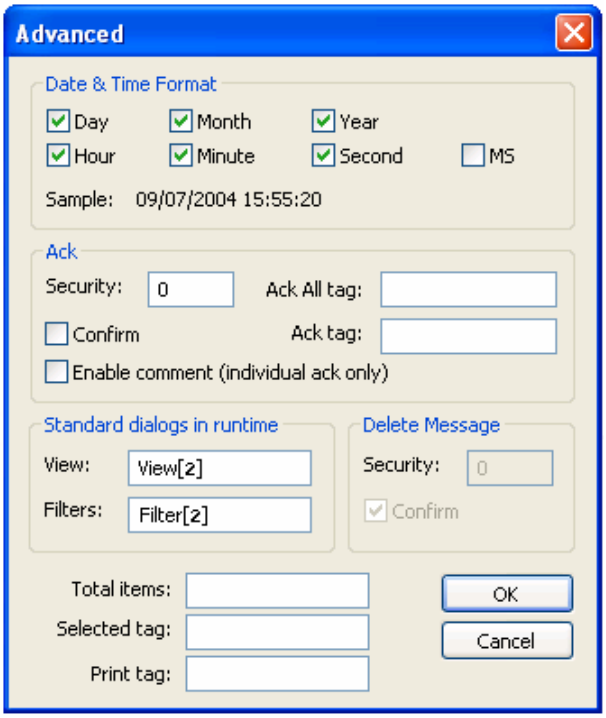
Configuring the Alarm History Filters

☑ Click the **Column** button and use the following figure to configure the filter**:**



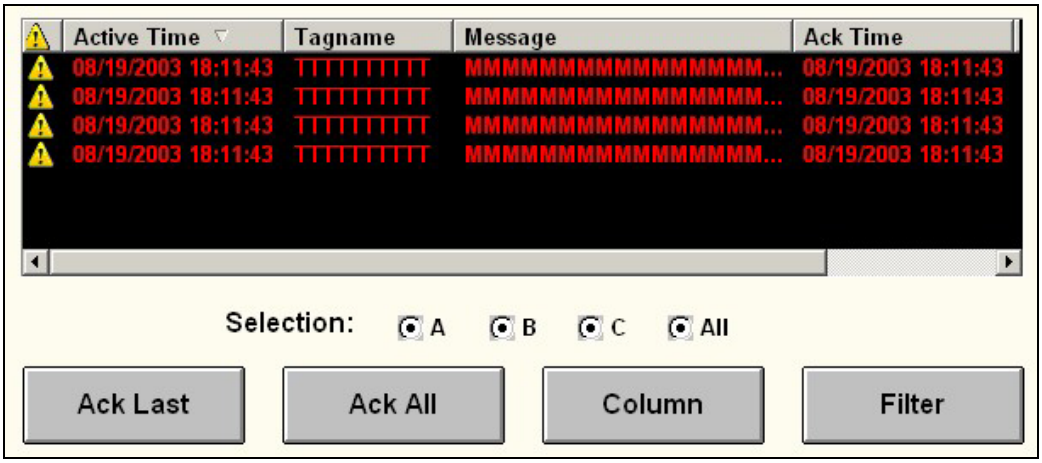
Configuring the Alarm History Columns

☑ 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.

Chapter 16. Driver Communication

Communication (for this class) means to read values from the devices' memory 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 select drivers,

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:

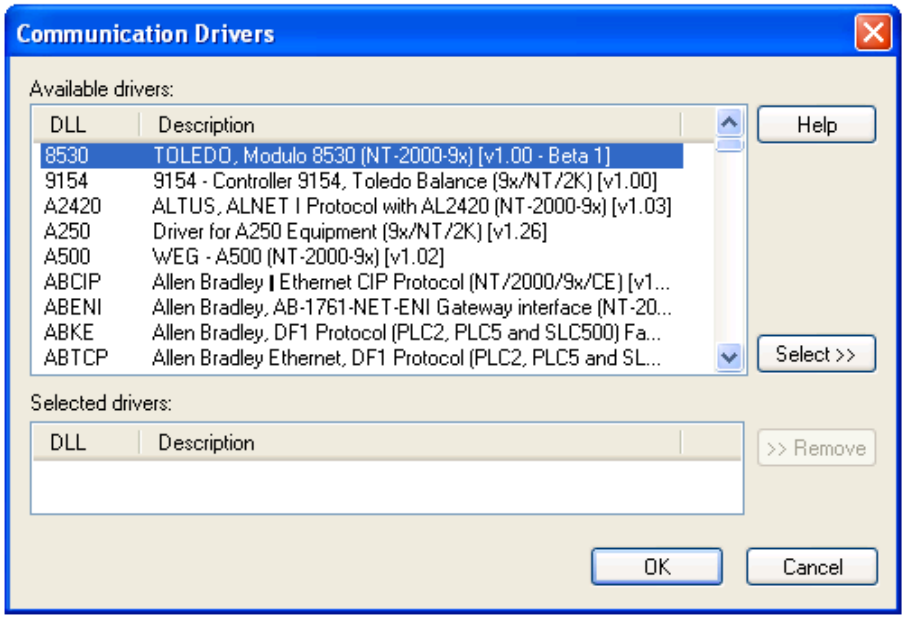
- Allen Bradley: DF1
- Siemens: S5 – AS511 PG Port
- Profibus DP Master and Slave (Hilscher)
- Allen Bradley: ControlNet Slave
- OMRON: Host Link
- GE FANUC: SNP, 90-30 90-70 Series
- Modbus: Schneider 984 series
- Profibus DP Master
- Cutler-Hammer: D50 – D300 Series
- Hitachi: H series
- Toshiba: Prosec T1/T2

Configuring a Driver

When you installed InduSoft Web Studio, you also installed all the drivers.

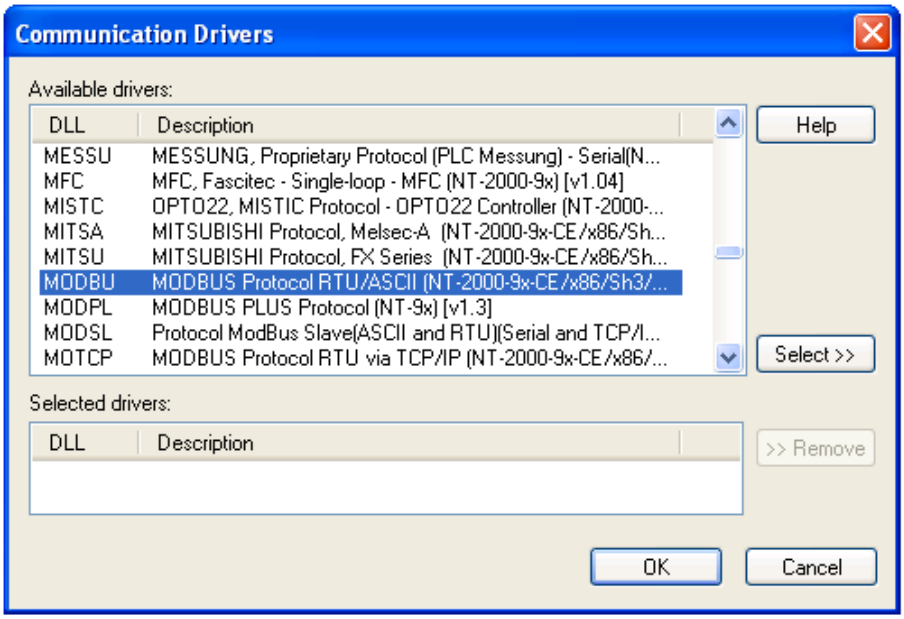
Use the following process to configure any driver:

- ☑ 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.

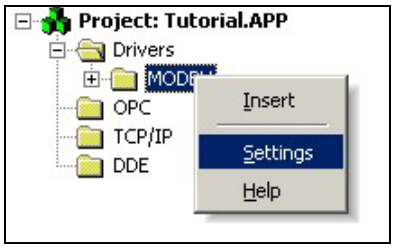


Selecting the MODBUS Driver

Next, configure the serial channel and other parameters for the driver.

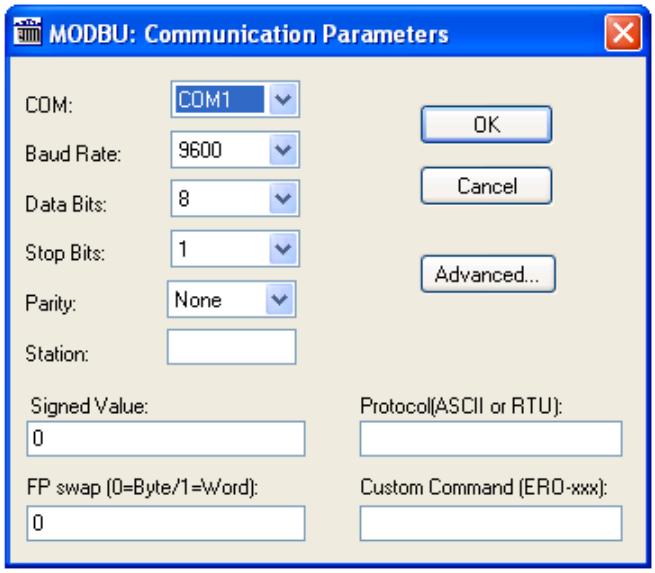
> **Note:**
> The values you configure will be stored immediately in a system-related file. *You cannot save or cancel this operation.* Any changes you make will take effect only after the driver is initialized. Consequently, if the driver is running and you change any of the driver parameters, your 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.

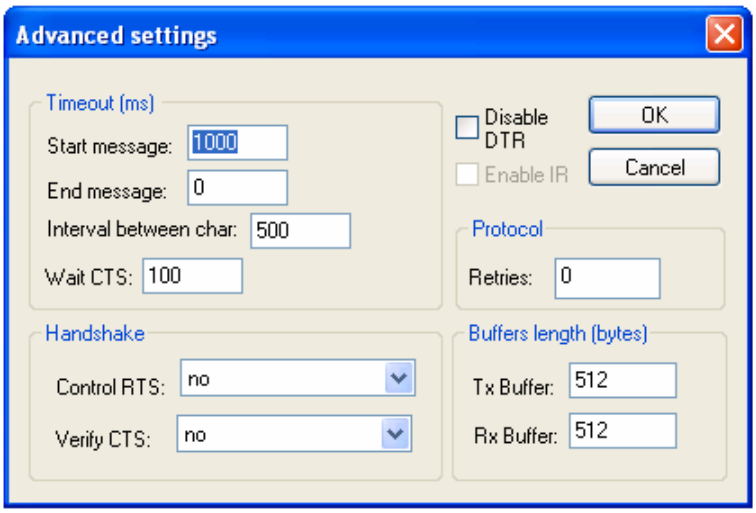
Note:
You *must* configure your device with the *same* values defined in the *Communication Parameters* dialog.

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.

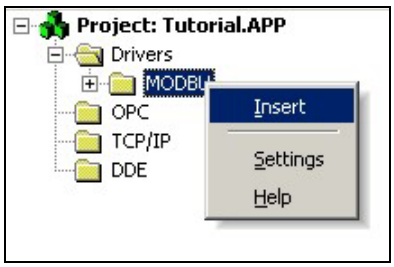
Note:
Generally, you change these parameters only when using a *DCE* (Data Communication Equipment) converter (232/485, for example), modem, and so forth between the PC where the driver is running and the Host. You must know the features of the DCE before setting the *Advanced* parameters. The advanced 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).



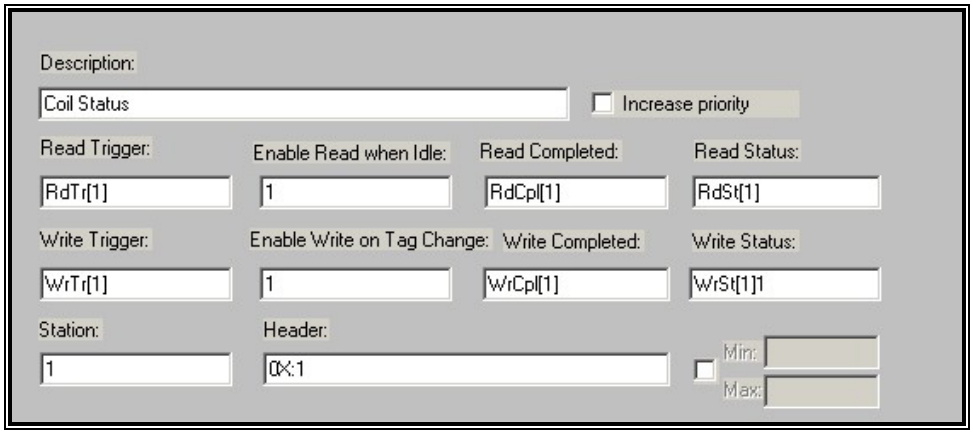
Inserting a MODBU Driver Worksheet

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:

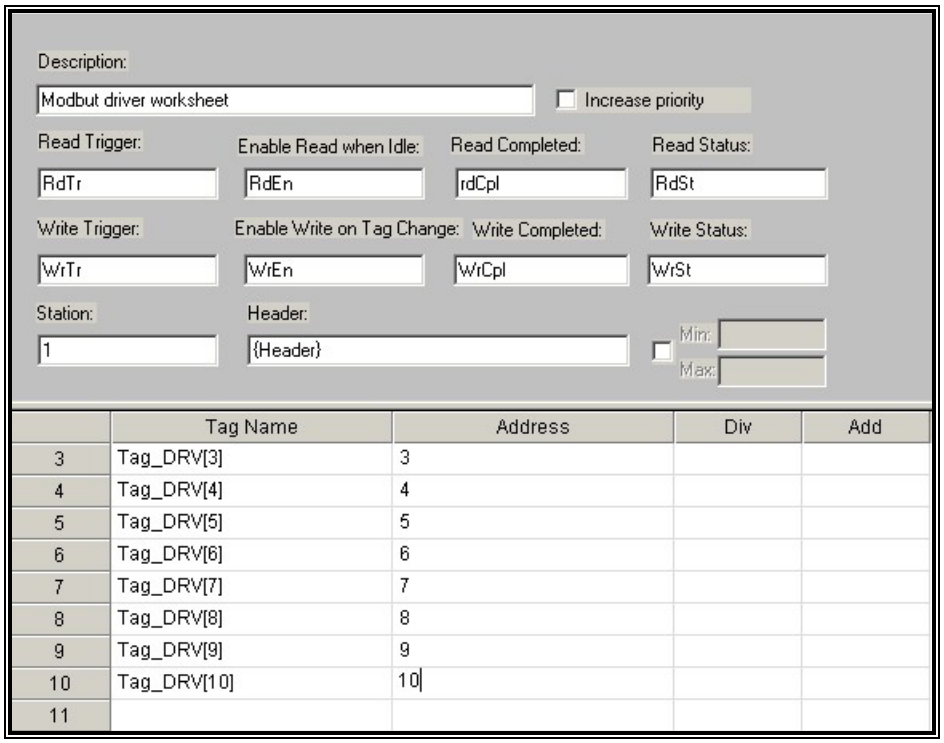
☑ Specify the following tags in the *Driver* worksheet **Header** fields. All of the tags will be arrays, and you must type each element on each worksheet.

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

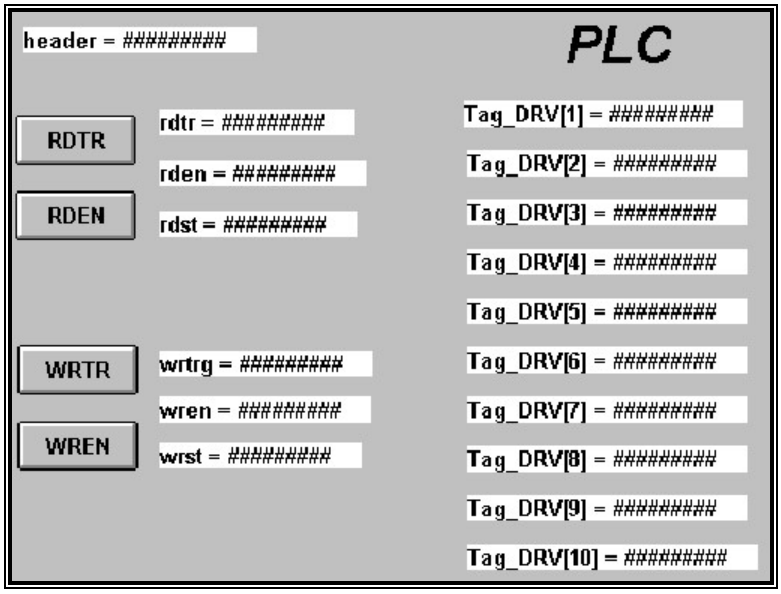
☑ Specify **TAG_DRV**, size 10 for the communication tags.

☑ 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

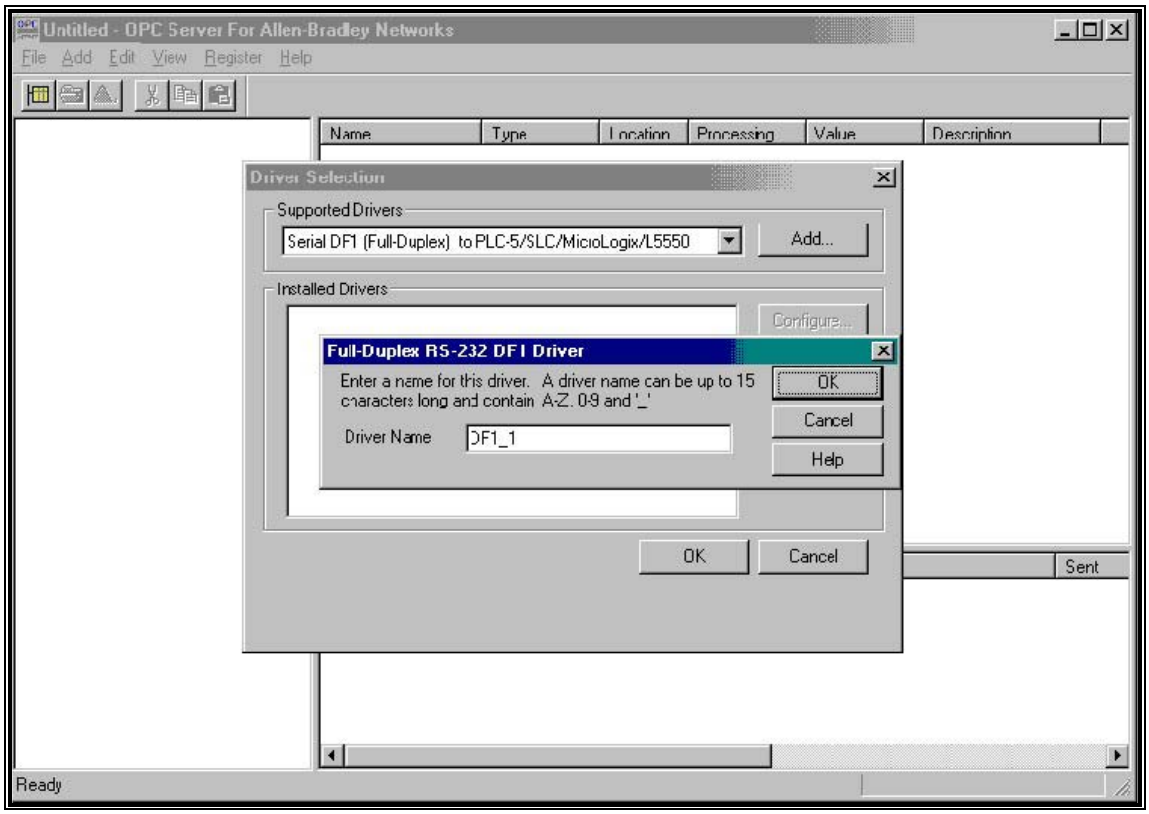
The InduSoft Web Studio *OPC Client* module enables the InduSoft Web Studio system to communicate with any device that implements an OPC Server. This module implements the OPC standard as described in the document "OLE for Process Control Data Access Standard," available at the site http://www.opcfoundation.com.

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*.

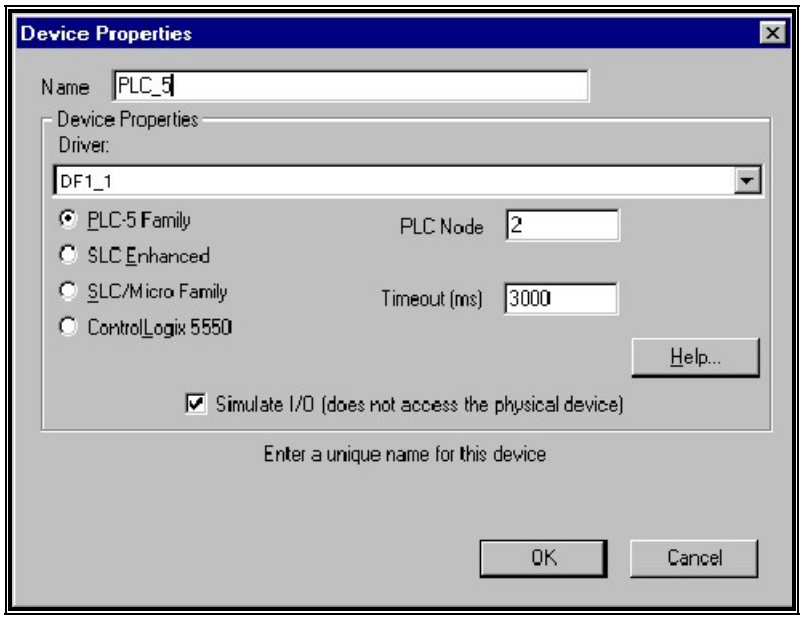
- ☑ Click **Add → New Device**. You will be prompted to create a new driver. Accept it.
- ☑ The first option is the **DF1 Serial**. Because the communication method does not matter (we are going to simulate a communication), you can choose the **DF1**.



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*):

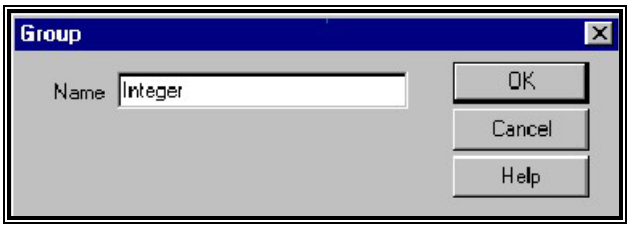


Device Properties Dialog

Now you can add the new OPC Server items.

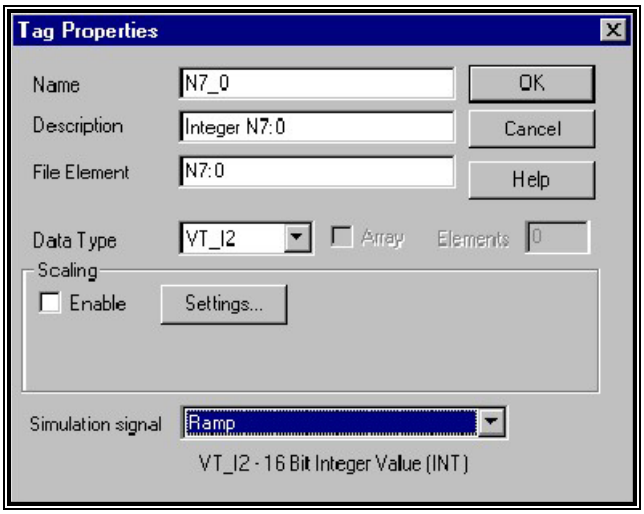
You must tell the OPC Server with which PLC address to communicate. Organize it in groups, sub-groups, and so forth. For this example, just create two groups.

☑ 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*.

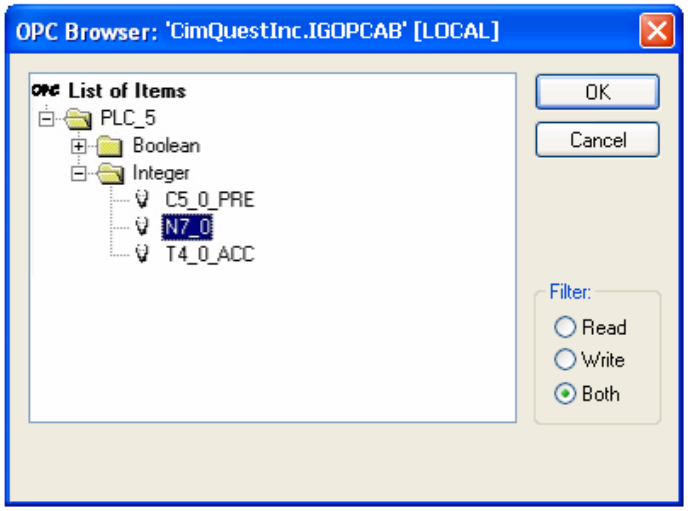
  Before you can use a control (or a COM Server), you must create specific entries in the Windows registry to declare its availability to the OLE client application.
- ☑ Select the **Database** tab and create a new set of tags to communicate with the OPC Server, as shown in the following figure (review page 5–7 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

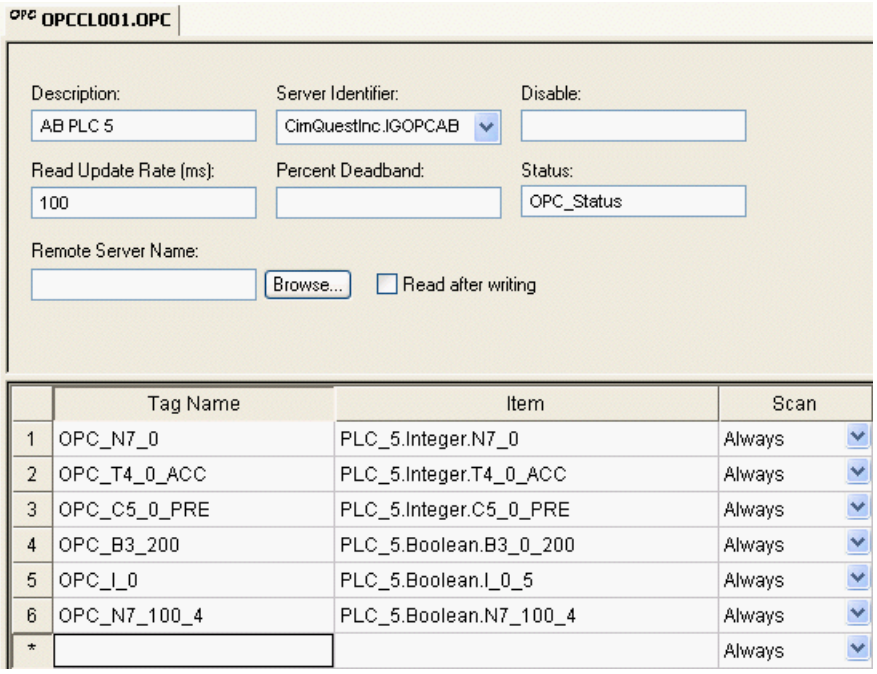
Creating Tags to Communicate with OPC Server

- ☑ 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.



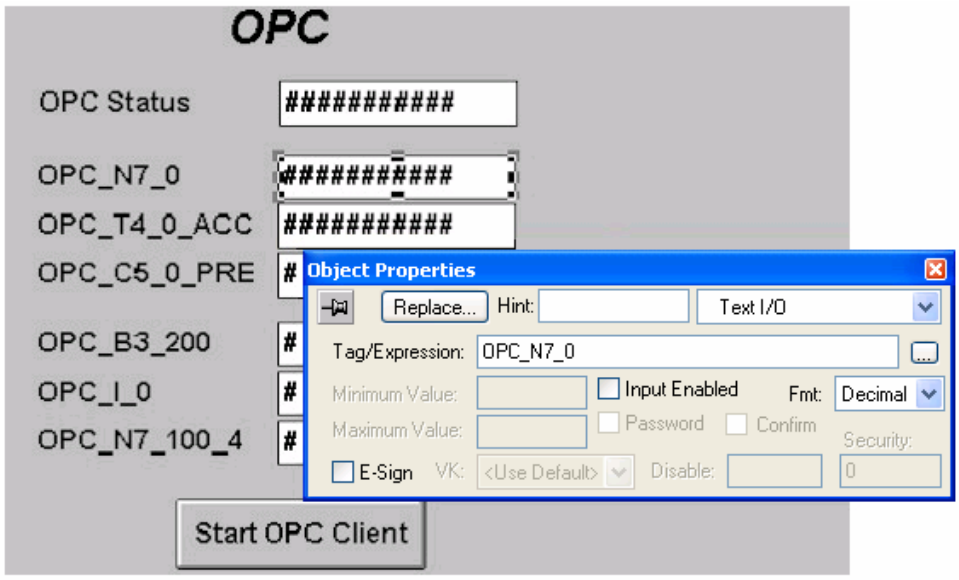
Select N7_0

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 17.

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

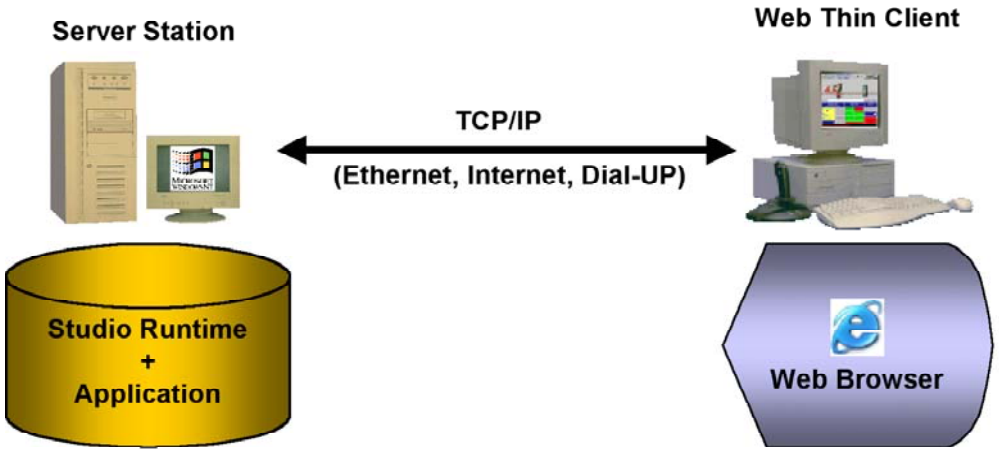
You can configure the following parameters for the **Application Configuration** (**.app**) file:

- **[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 17. 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).

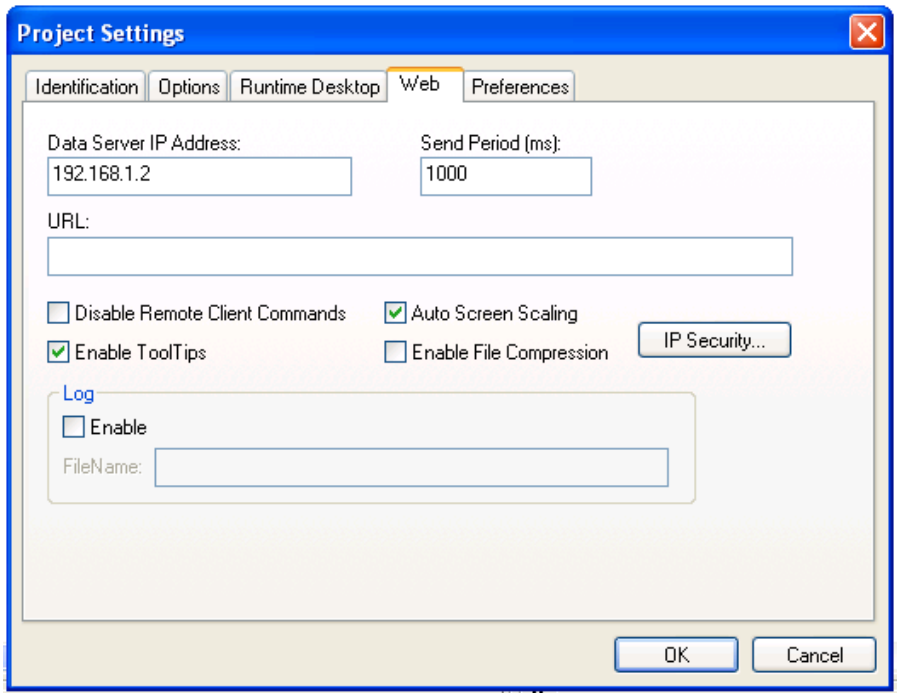
Configuration example:

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**

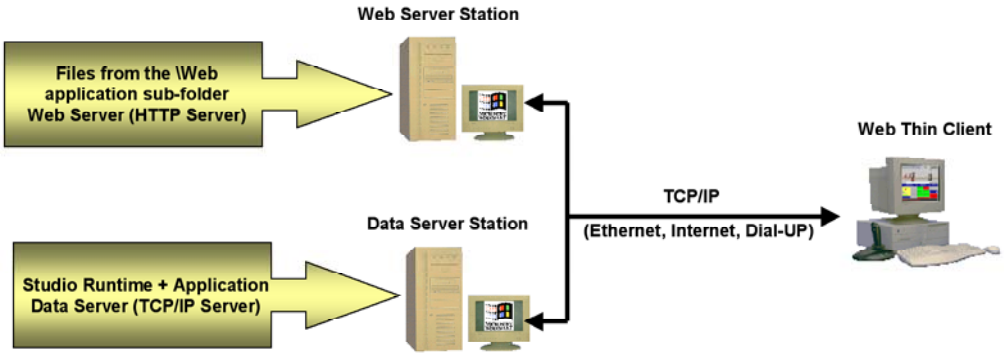
The **Project Settings → Web** interface must be configured as follows:



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).

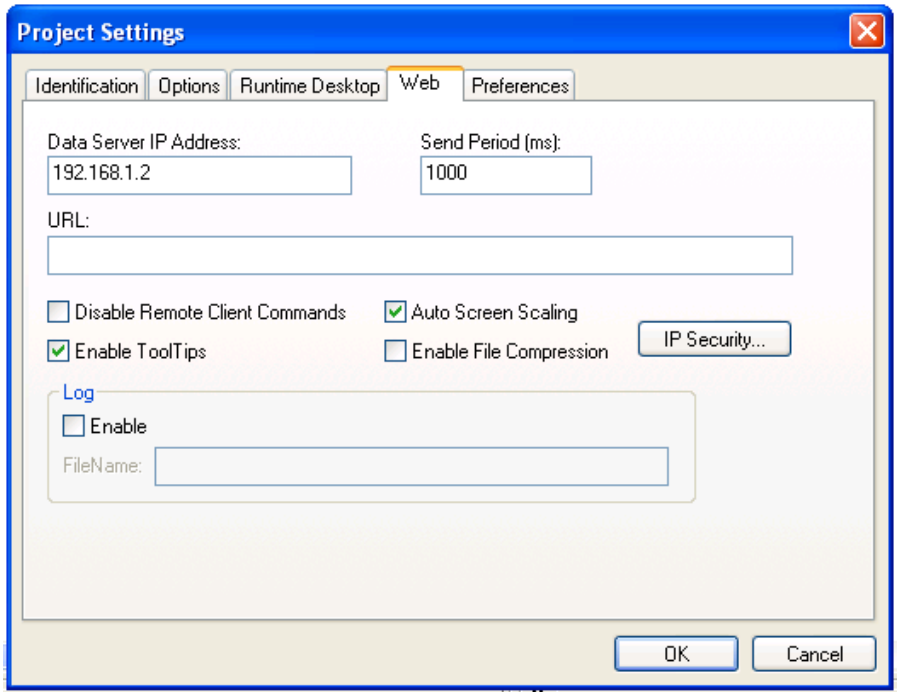
Configuration Example:

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

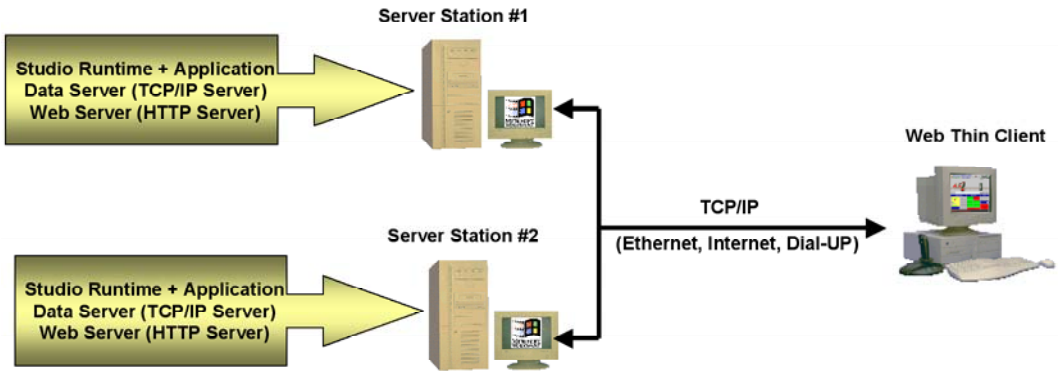
You must type the following address on the remote web browser to access a screen (e.g. **myscreen**) from the server: **http://192.168.1.1/myscreen.html**

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:

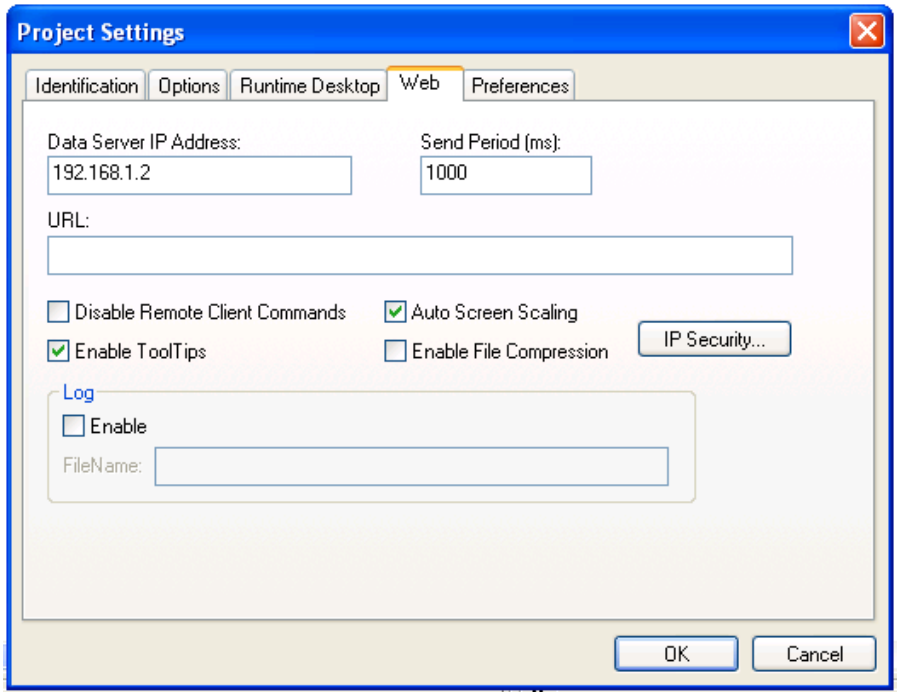
- IP address of server station #1 on the network: **192.168.1.1**
- IP address of server station #2 on the network: **192.168.1.2**
- Home directory of the web server (HTTP server) on server station #1: **\Web** sub-folder of the application stored on server station #1.
- Home directory of the web server (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**

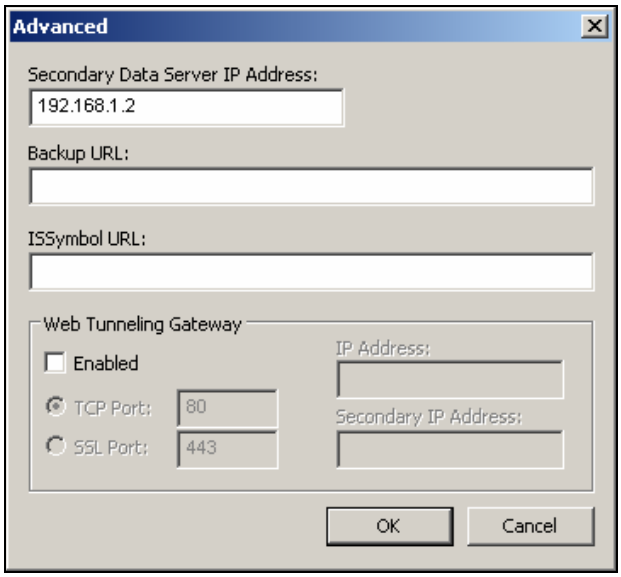
The user must type the following address on the remote web browser to access a screen (e.g. **myscreen**) from server station #2: **http://192.168.1.2/myscreen.html**

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*.

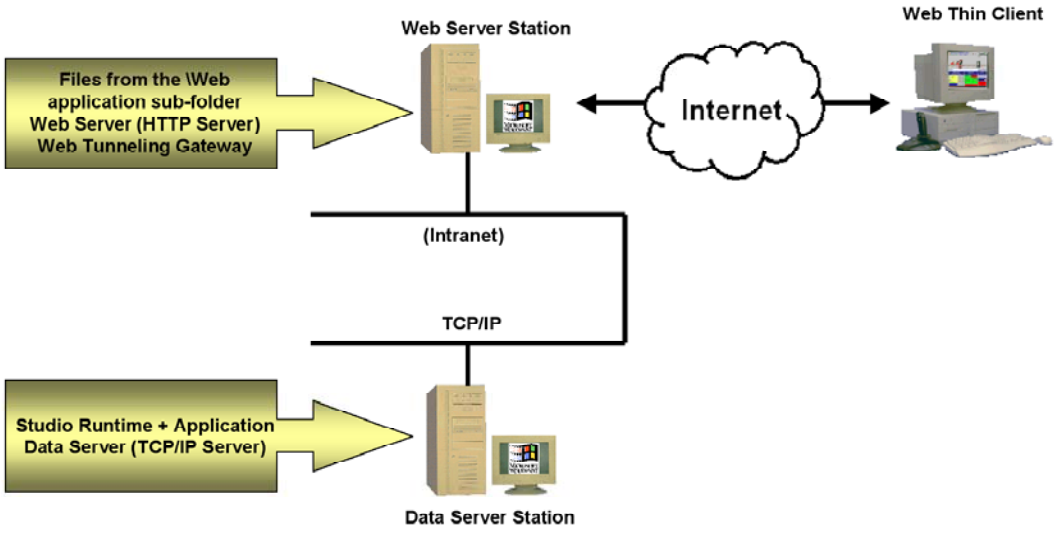


Project Settings → Web Interface



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.

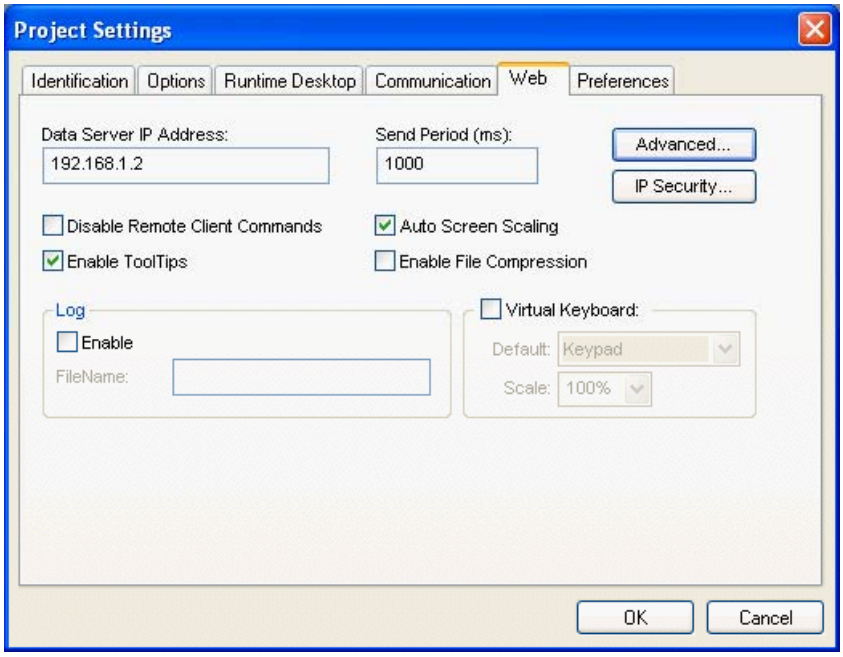
Configuration example:

This example is based on the following premises:

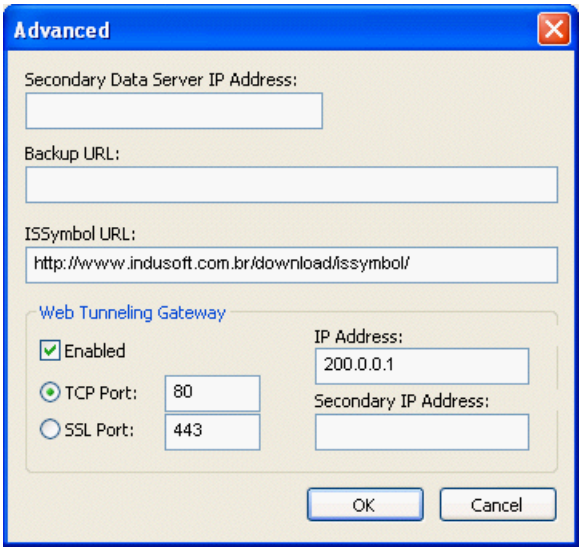
- IP address of the web server station (internet): **200.0.0.1**
- IP address of the web server station (intranet): **192.168.1.1**
- IP address of the data server station on the intranet: **192.168.1.2**
- Home directory of the web server (HTTP server) on the web server station: **\Web** sub-folder 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

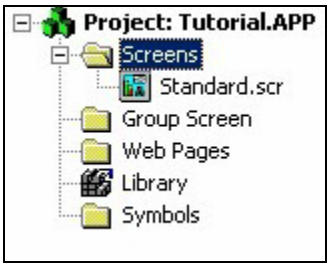
Note:
If your web server is able to provide files via HTTPS (SSL – Secure Socket Layer), you can select this option on the *Advanced* dialog window from the **Project Settings → Web** interface.

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:

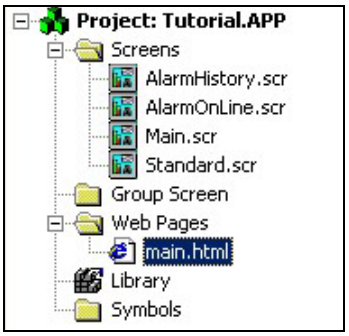
☑ Expand the *Screens* folder and double-click on your *Main.scr* screen.



Selecting a Screen

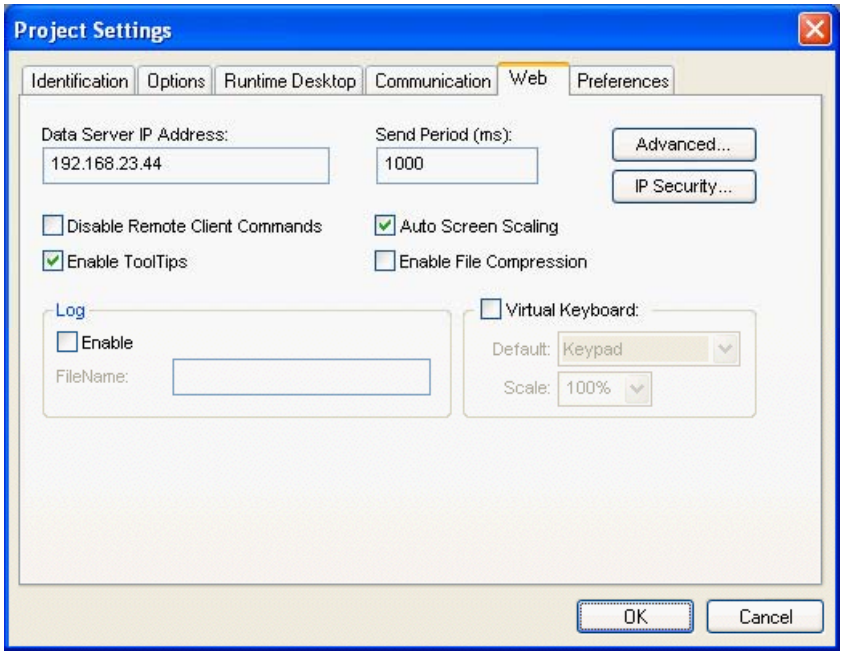
☑ Select **File →Save as HTML** to save the screen in HTML format.

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.

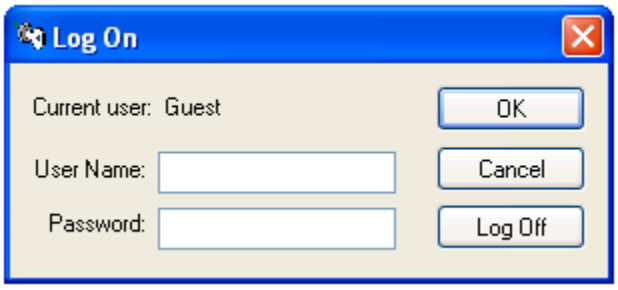
☑ After configuring the Web settings, click **OK** to close the *Project Settings* dialog.

☑ 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.

Chapter 18. Managing Applications Remotely

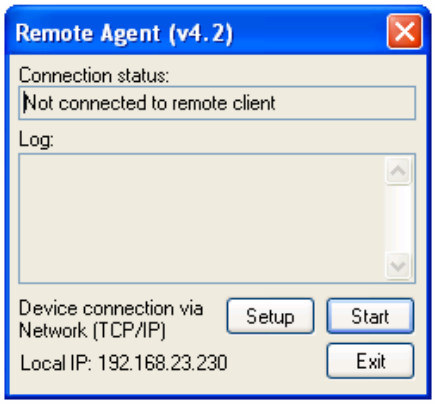
After configuring an application and testing it locally (on your development workstation), you can download the application to a remote run-time workstation that is running under Windows NT/2000/XP or CEView under Windows CE.

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.

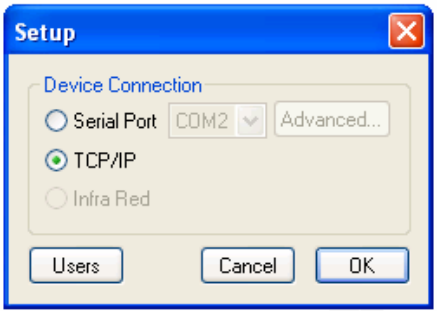
☑ Run the **CEServer.exe** to launch the *Remote Agent* dialog on the remote workstation.



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.

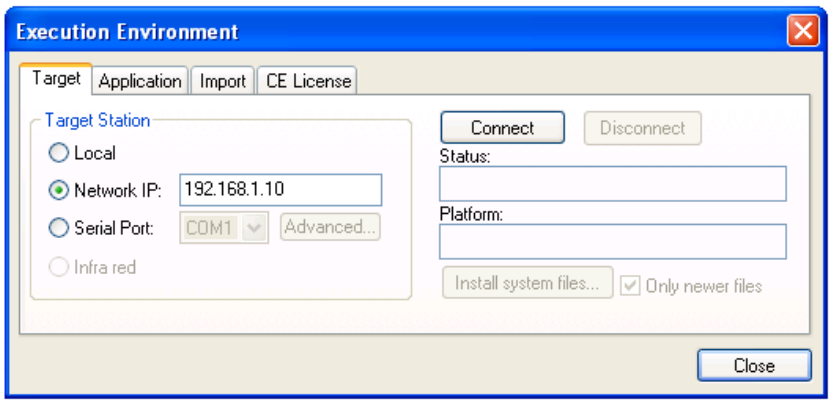


Remote Agent Setup Dialog

> **Note:**
> We recommend using TCP/IP instead of Serial Link for performance reasons.

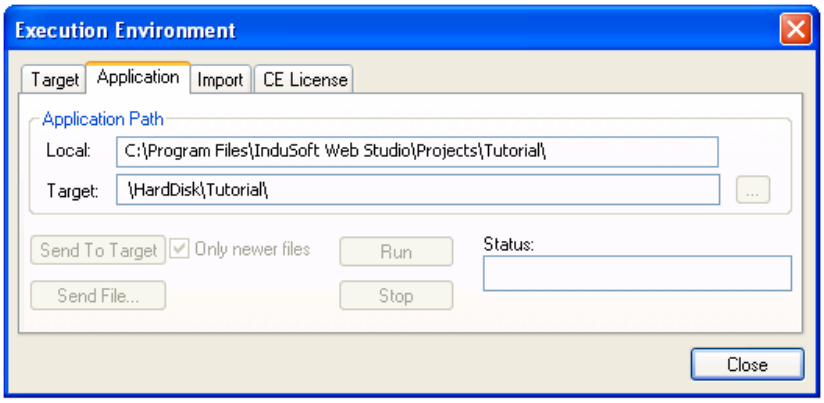
☑ Click **OK** to close the *Setup* dialog, but leave the *Remote Agent* program running in the remote workstation.

☑ 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.

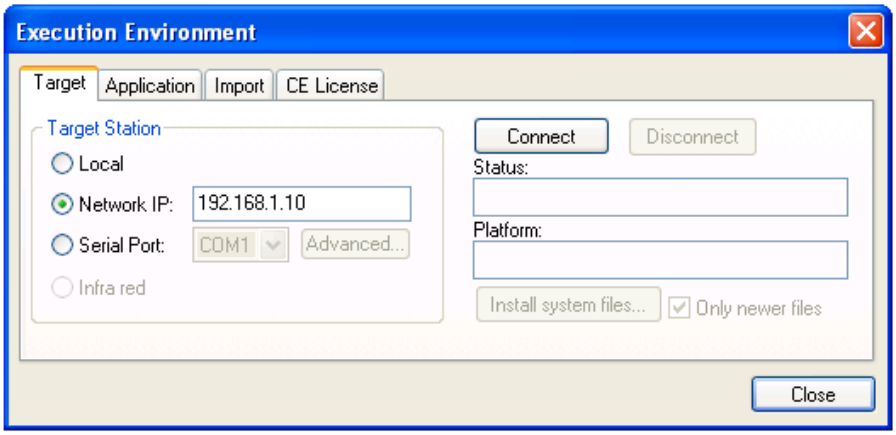
☑ In the *Workspace* window, select the **Application** tab and click the **Send to Target** button to download the application to the remote workstation.

☑ After downloading all application files, click the **Run** button to execute the application in the remote target workstation.

Downloading a CEView Application

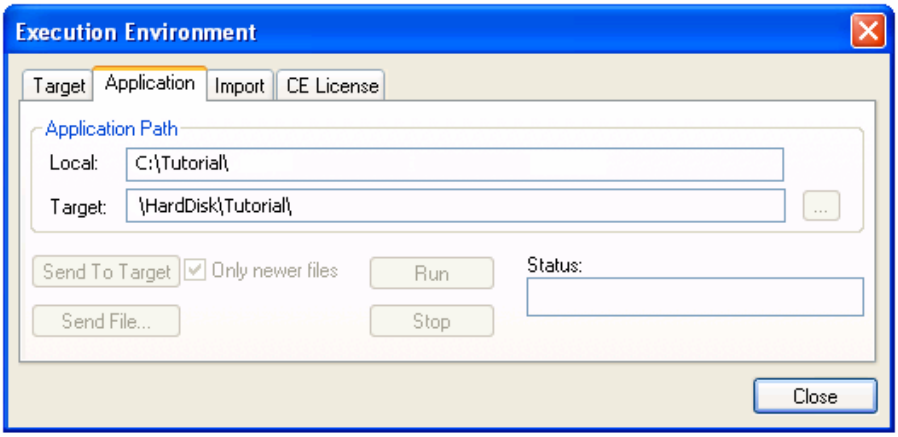
You must use the *Project Settings* dialog to configure the **CEServer.EXE** for TCP/IP communications before running your CE unit.

- ☑ 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.



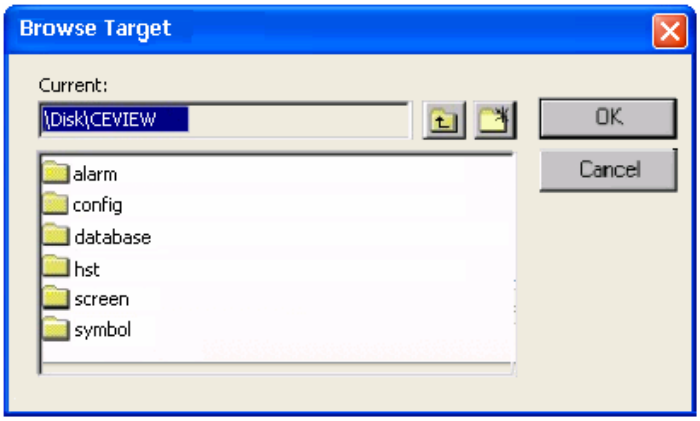
Execution Environment Dialog

- ☑ 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

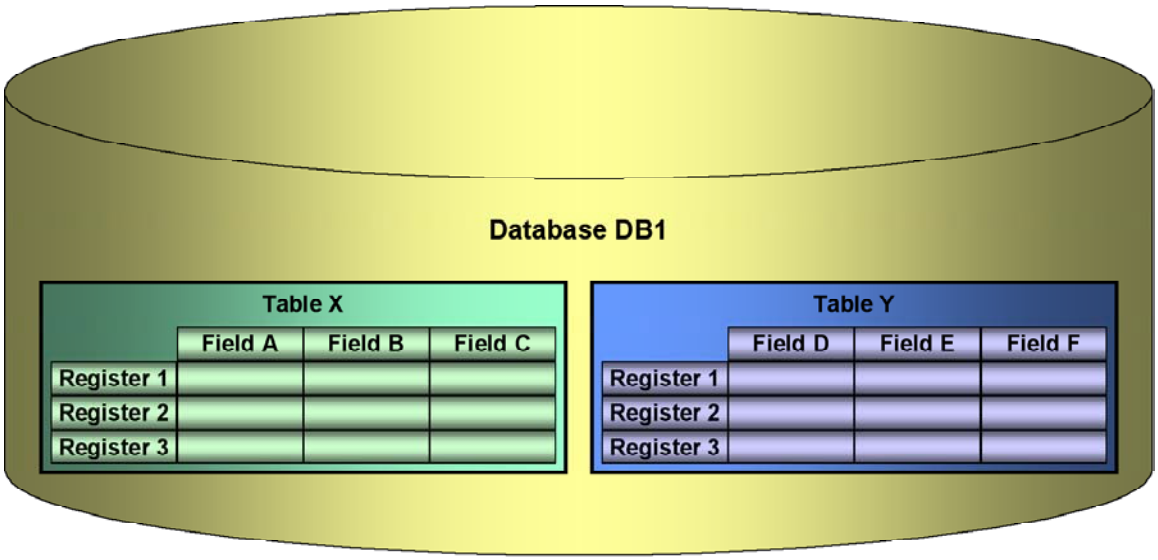
General Concepts

This section describes databases, database providers and the way IWS interfaces with different databases.

SQL Relational Databases

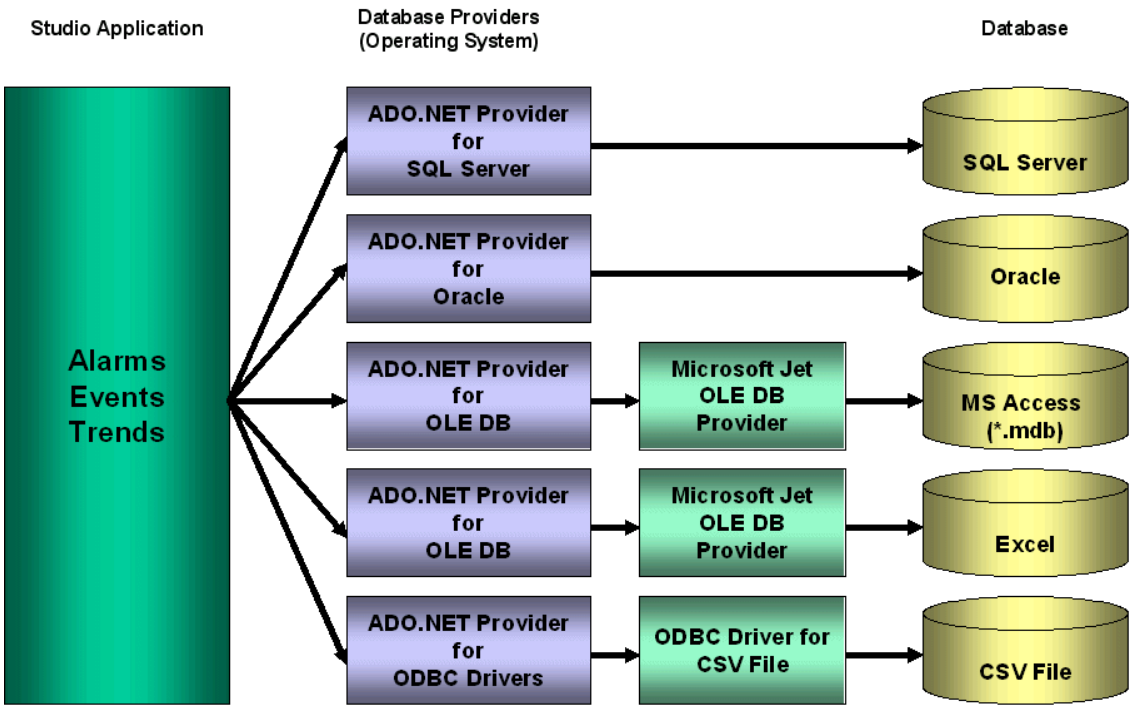
A SQL Relational Database is a set of information stored in tables with fields and registers, which support SQL commands.

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.

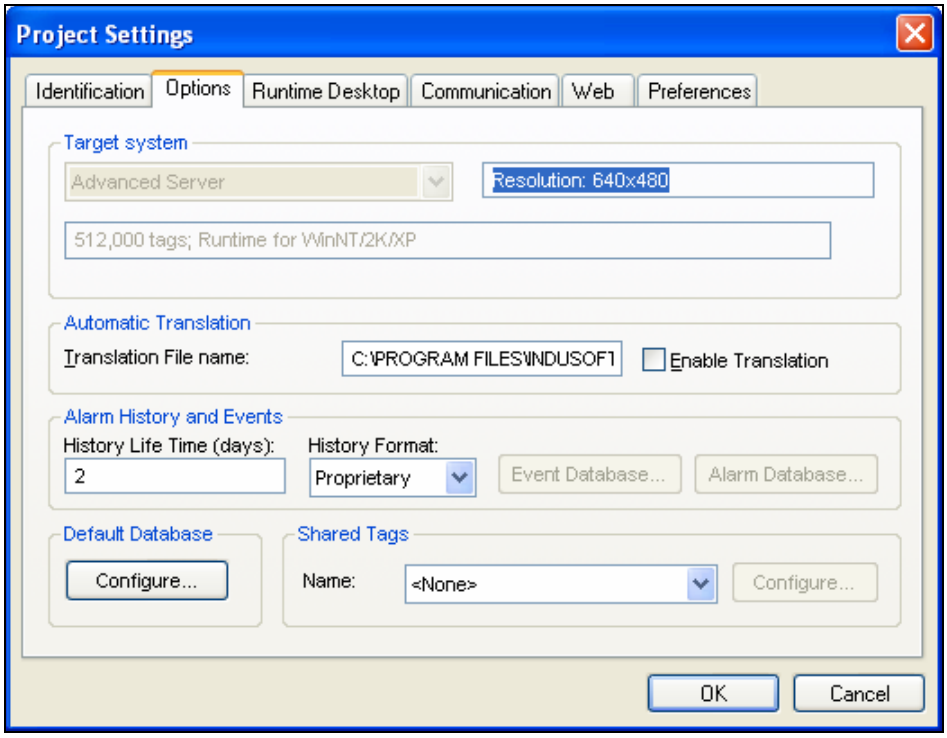
> **Note:**
> The Primary and Secondary can be different types of databases. However, they must have the same fields.

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).

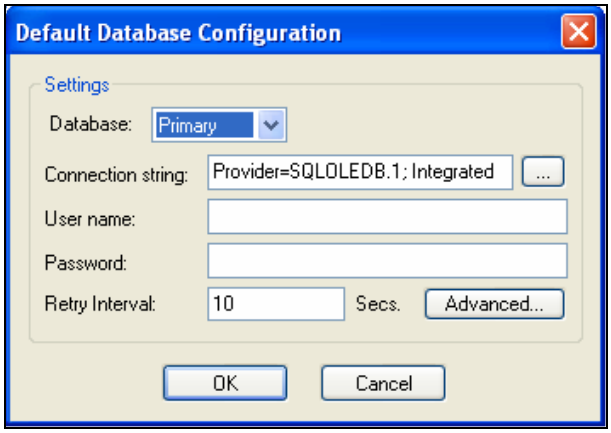
Default Database

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.

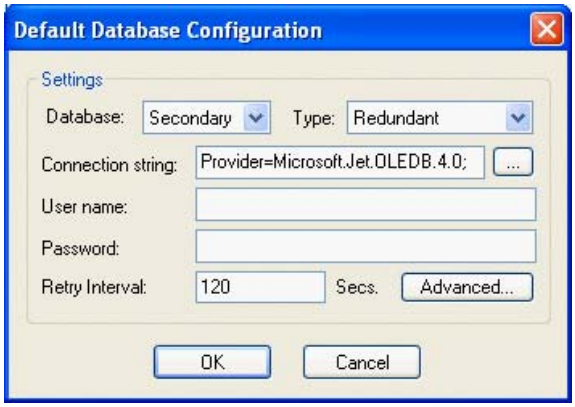
Note:

- 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.

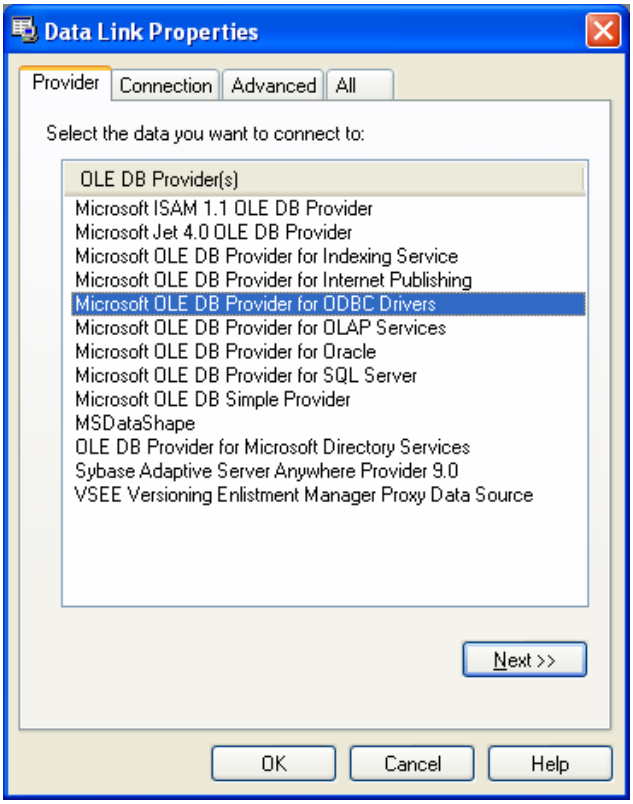
The following picture illustrates the Database Configuration dialog:



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.

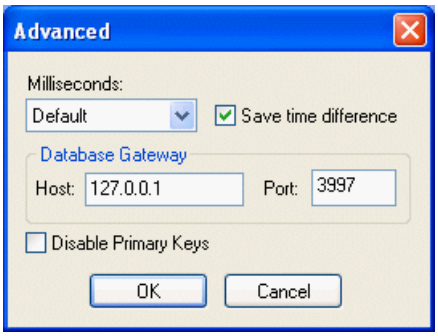
- **Connection string** field: This field defines the database where IWS will write and read values as well as the main parameters used when connecting to the database. Instead of writing the Connection string manually, you can press the browse button (…) and select the database type from the **Data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

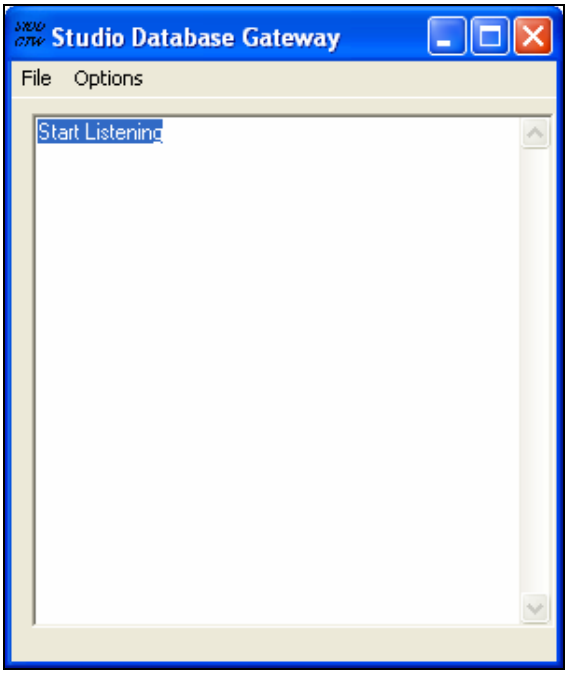
Studio Database Gateway

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 (?)

Parameter	Range of Values	Description
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.
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:

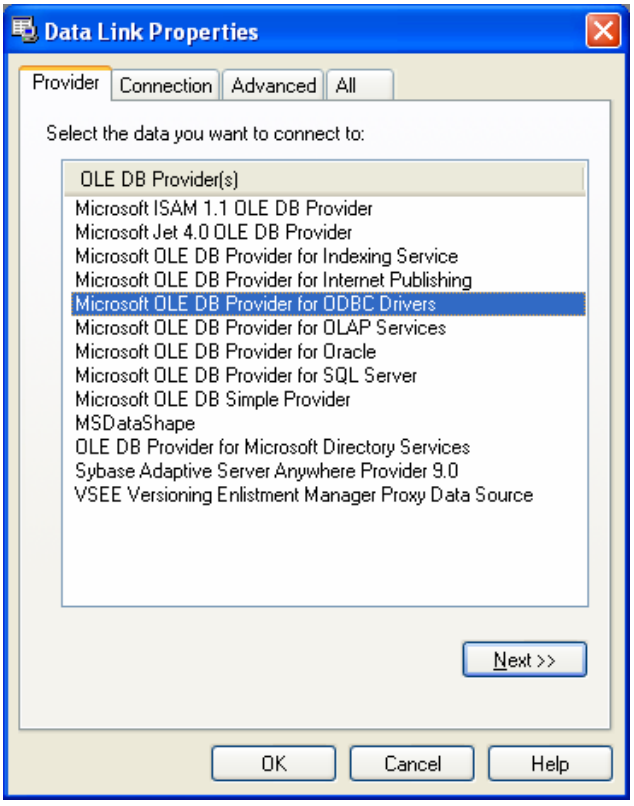
```
Provider1=MICROSOFT.JET.OLEDB
SaveMSec1=3

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

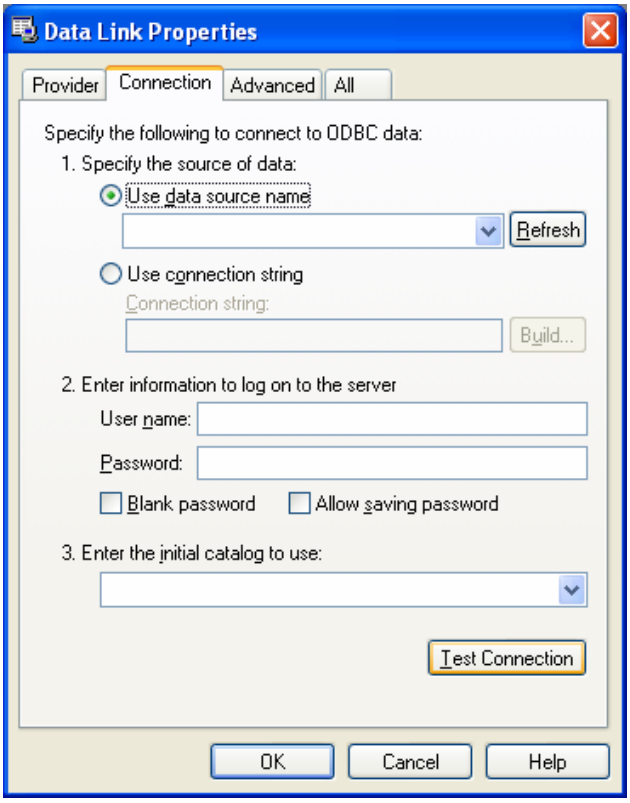
Almost every database provides an ODBC interface that can be used to interface with it. The database features provided by IWS can be used with ODBC drivers through the ADO.NET interface for ODBC. In order to use this capability, you must use Microsoft *.NET Framework 1.1* or higher.

> **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.