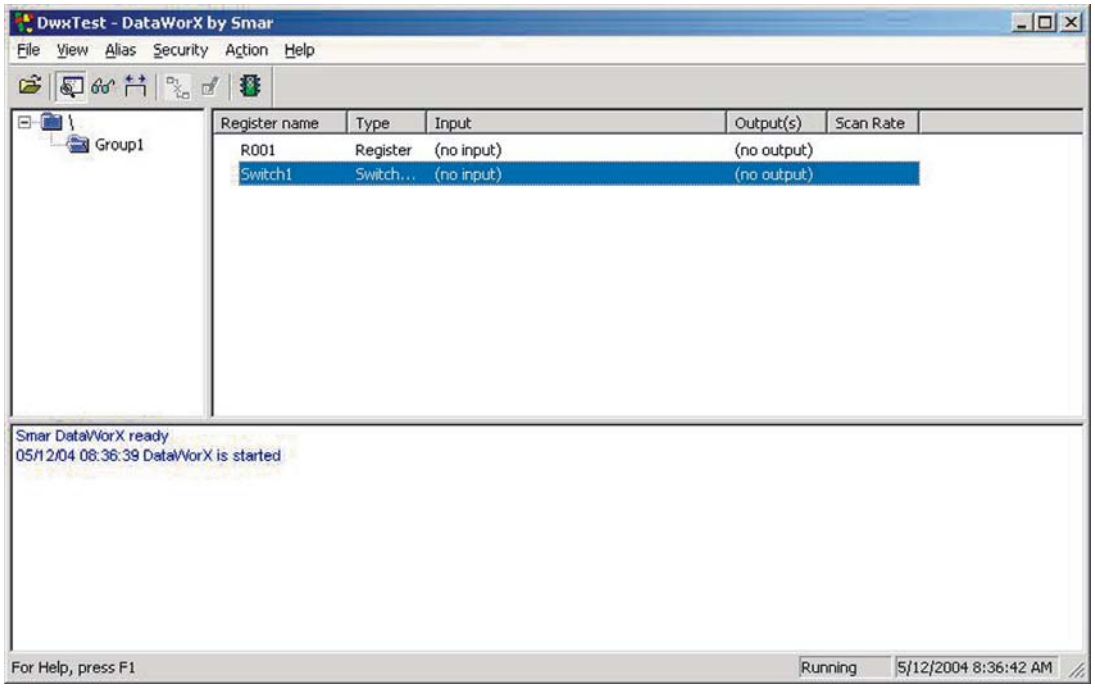
DataWorX

smar

FIRST IN FIELDBUS

USER´S GUIDE

DataWorX



JUN / 04

DataWorX

VERSION 7.1

FOUNDATION

PVI EWDWXME

smar

web: www.smar.com

Specifications and information are subject to change without notice.
For the latest updates, please visit the SMAR website above.

BRAZIL
Smar Equipamentos Ind. Ltda.
Rua Dr. Antonio Furlan Jr., 1028
Sertãozinho SP 14170-480
Tel.: +55 16 3946-3510
Fax: +55 16 3946-3554
e-mail: smarinfo@smar.com

ARGENTINA
Smar Argentina
Soldado de La Independencia, 1259
(1429) Capital Federal – Argentina
Telefax: 00 (5411) 4776 -1300 / 3131
e-mail: smarinfo@smarperifericos.com

CHINA
Smar China Corp.
3 Baishiqiao Road, Suite 30233
Beijing 100873, P.R.C.
Tel.: +86 10 6849-8643
Fax: +86-10-6894-0898
e-mail: info@smar.com.cn

FRANCE
Smar France S. A. R. L.
42, rue du Pavé des Gardes
F-92370 Chaville
Tel.: +33 1 41 15-0220
Fax: +33 1 41 15-0219
e-mail: smar.am@wanadoo.fr

GERMANY
Smar GmbH
Rheingaustrasse 9
55545 Bad Kreuznach
Germany
Tel: + 49 671-794680
Fax: + 49 671-7946829
e-mail: infoservice@smar.de

MEXICO
Smar México
Cerro de las Campanas #3 desp 119
Col. San Andrés Atenco
Tlalnepantla Edo. Del Méx - C.P. 54040
Tel.: +53 78 46 00 al 02
Fax: +53 78 46 03
e-mail: ventas@smar.com

SINGAPORE
Smar Singapore Pte. Ltd.
315 Outram Road
#06-07, Tan Boon Liat Building
Singapore 169074
Tel.: +65 6324-0182
Fax: +65 6324-0183
e-mail: info@smar.com.sg

USA
Smar International Corporation
6001 Stonington Street, Suite 100
Houston, TX 77040
Tel.: +1 713 849-2021
Fax: +1 713 849-2022
e-mail: sales@smar.com

Smar Laboratories Corporation
10960 Millridge North, Suite 107
Houston, TX 77070
Tel.: +1 281 807-1501
Fax: +1 281 807-1506
e-mail: smarlabs@swbell.net

Smar Research Corporation
4250 Veterans Memorial Hwy.
Suite 156
Holbrook , NY 11741
Tel: +1-631-737-3111
Fax: +1-631-737-3892
e-mail: sales@smarresearch.com

Index

Getting Started

Introduction

DataWorX is an OPC-compliant client and server application providing multiple functionality. DataWorX is a component of the ProcessView product family, and it serves as a project-level data system for ProcessView applications. Acting as a bridge between various OPC Servers, DataWorX provides different OPC Data Channels. Once multiple IO channels are established between PCs, DataWorX will switch between a primary PC (node) and a backup PC on the network. Should the primary PC be disabled, DattaWorX will automatically (should the options be specified) default to the backup PC and vice versa. Another feature of DataWorX is the use of global variables that are accessible from multiple clients.

About DataWorX

DataWorX is a 32-bit multithreaded application that runs under Windows 95, 98, and NT 4.0.

Software Licensing Protection Necessary for Runtime Mode

DataWorX is installable as a stand-alone COM component. It registers itself with the ProcessView license copy protection mechanisms. DataWorX permits unlimited configuration capabilities with or without the software license protection present. It requires a software license protection to be present for runtime mode. If such licensing is not present, DataWorX runs for 2 hours in runtime mode before timing out.

DataWorX Interface With the ProcessView Security Application

From within the security configuration application, you can restrict access to the various features and/or data available via DataWorX.

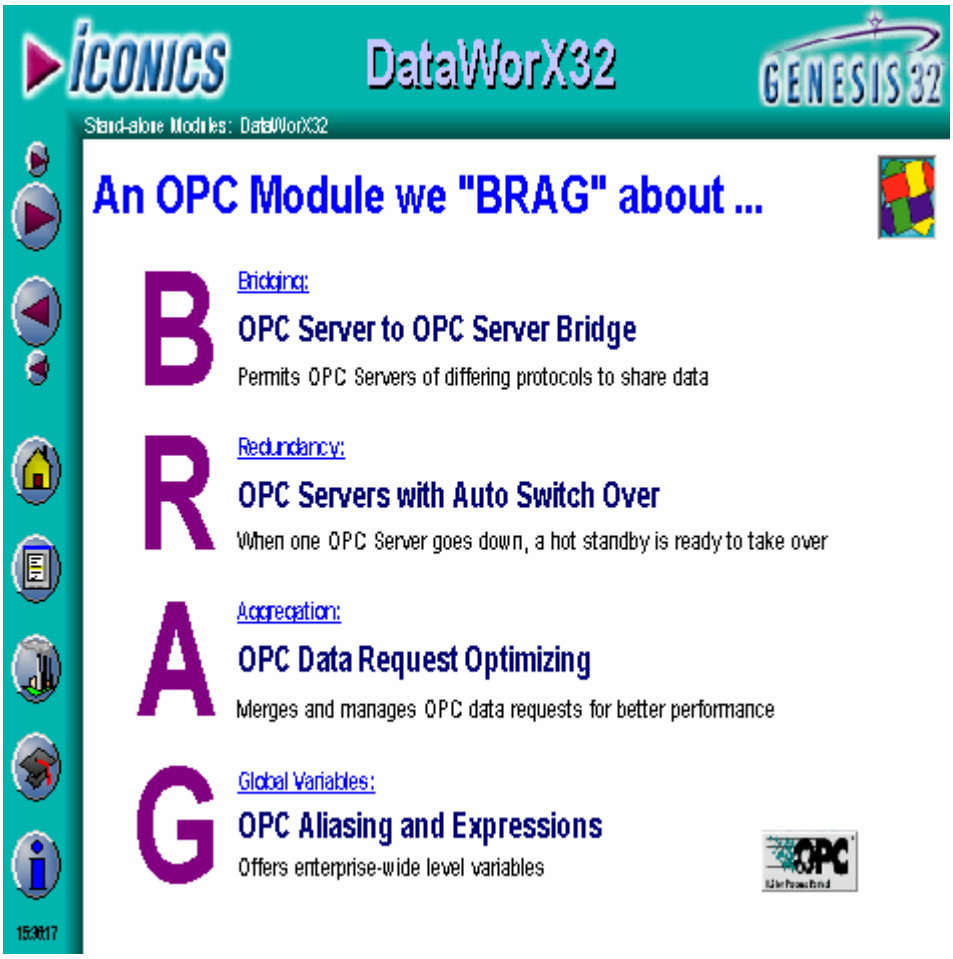
DataWorX supports multiple languages. That is, menu items, forms and various other text are available for translation into other international languages through the Smar Language Configurator.

Key DataWorX Capabilities

DataWorX is an OPC 1.0 and 2.0 compliant server and client. It conforms to Microsoft COM/DCOM program practice.

The main features of DataWorX include:

- OPC data bridging between clients and servers.
- OPC server redundancy.
- OPC aggregation.
- Global variables.



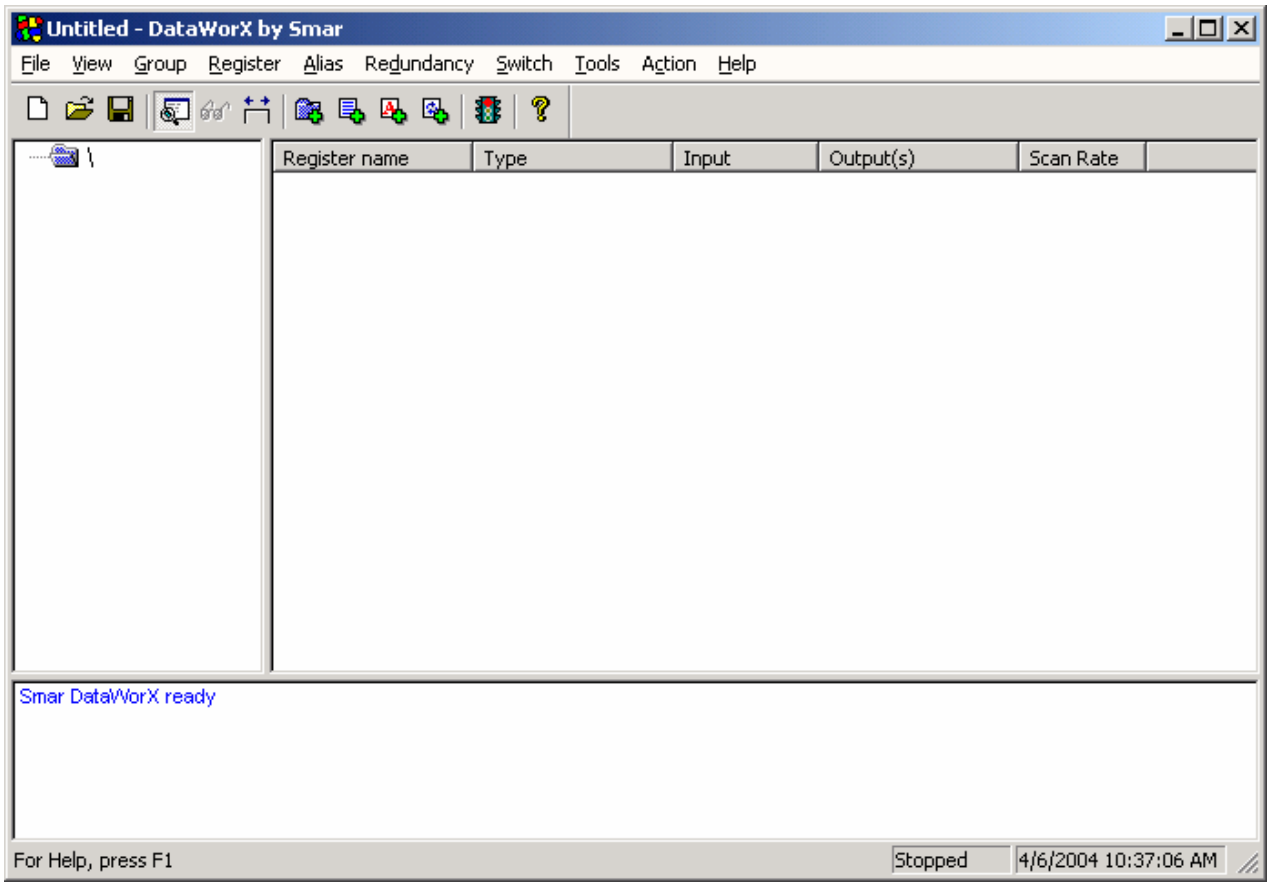
The DataWorX Container allows for configuration in the configuration mode, and a runtime monitor enables you to view current values in the runtime mode.

This OPC 2.0 compliant client and server application allows Aliasing and the use of Expressions. DataWorX also contains a Statistics and Performance analyzer, which can be accessed via the menu bar.

Starting DataWorX

To start DataWorX from the Windows **Programs** menu, select **Smar ProcessView > Tools > DataWorX.**

The DataWorX screen is displayed as shown below.



DataWorX Screen

Now you can create a new file or open an existing file and configure DataWorX. You can toggle between views of the screen with and without the output window (bottom pane) by using the toolbar buttons and the menu commands.

Features of DataWorX

Main Features

The main features of DataWorX include the following:

- Serves as an OPC bridge between clients and servers.
- OPC server redundancy with auto-switch-over performance.
- OPC client request aggregation.
- Global variables.
- Register grouping and organization.
- OPC 2.0 Interface on both the client and server side.
- Switch alias functionality.
- OLE Automation interface.
- Conditions that can be used as register inputs.
- Time stamp and quality to points accessed via automation.
- Output to NT Event Logger.
- New support for GenBroker OPC over TCP/IP communications.
- New filter support.
- Statistics and performance analyzer.
- Startup with a specified file.

- Primary node status register.
- CSV file import and export.
- Runtime command line option.
- Global aliasing support for OPC inputs and outputs.
- Changes made to DataWorX registers that are made together (in a single call to the OPC interface) are also made together. OPC clients connected to DataWorX are notified about all the changes in a single update.
- Support for GenBroker OPC over TCP/IP communications.
- Filter support.
- Improved redundancy with auto-switch-over performance.
- NATIVE register type for array support.
- Optional Refresh mechanisms
- Integrated Expression Editor
- Disable OPC propagation support
- New way of registering running instances

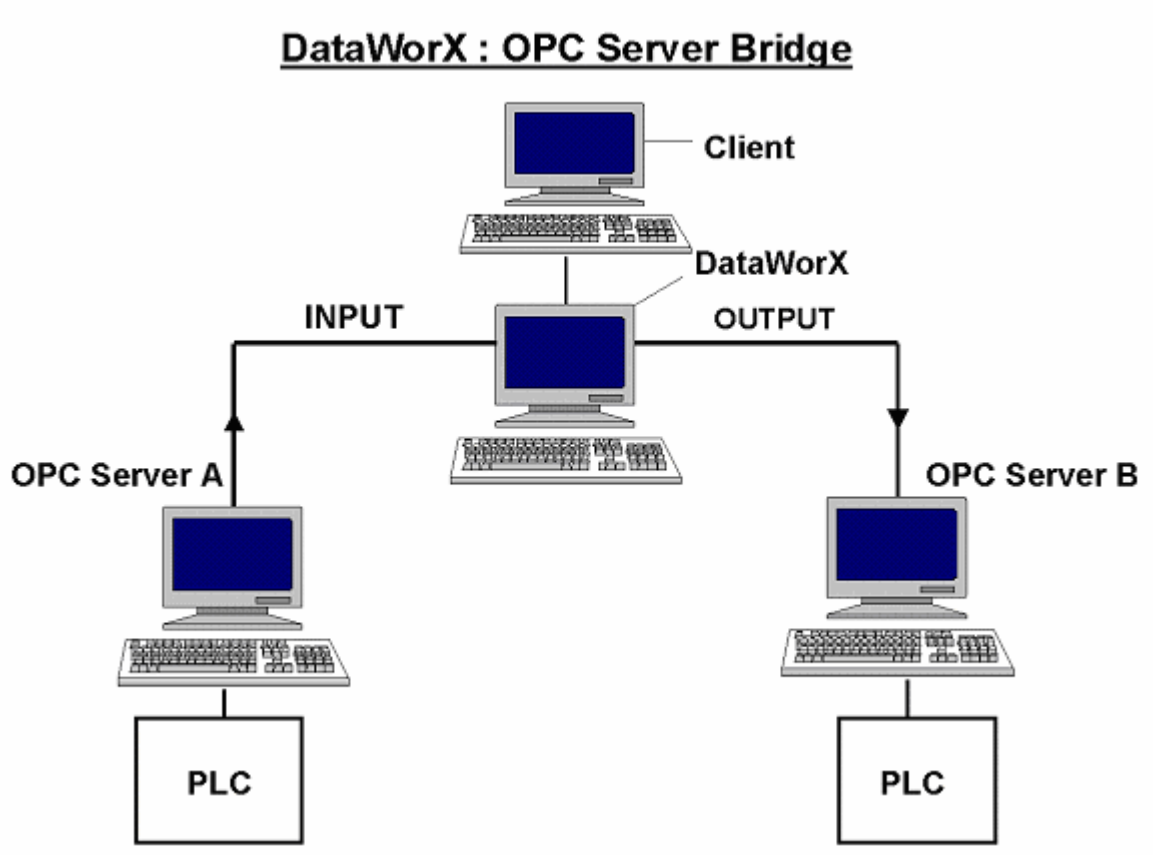
These features are described in detail in this section with illustrations and examples of their functionality.

DataWorX As an "OPC Bridge"

Customers sometimes need to pass information from one device on to another. For example, data from one brand of PLC need to be shared with another I/O device. In the past, users had to write their own programs to translate and move data from one server to another. DataWorX provides this server-to-server data exchange. It serves as an "OPC bridge" between two or more servers.
OPC bridging is a unique and powerful feature of DataWorX. As its name suggests, its purpose is to provide "bridging" between OPC servers of various types.
Should you be required to scan input from one OPC server and supply it to another OPC server, you may "perform" expressions before sending the output to the other server via DataWorX.
OPC bridging in DataWorX involves the following:

- A register is defined in the **Properties** tab of the **Register Settings** dialog box.
- The input and output for the register are defined in the **Input** tab and **Output** tab of the **Register Settings** dialog box.
- This register is also available to other clients while, depending on the options set in the **Input** tab, DataWorX continues to write values to output tags.
- There can be more than one output tag. This way DataWorX performs "bridging" between OPC servers by simultaneously reading values from one or more OPC servers and writing to one or more OPC servers.
- No scripting or programming is required for bridging.

The figure below illustrates the functionality of DataWorX as an OPC bridge between clients and servers. For another example of bridging, open the "DataWorX_Bridging.gdf" display in the "Smar/ProcessView/Examples/ProcViewDEMO" directory.

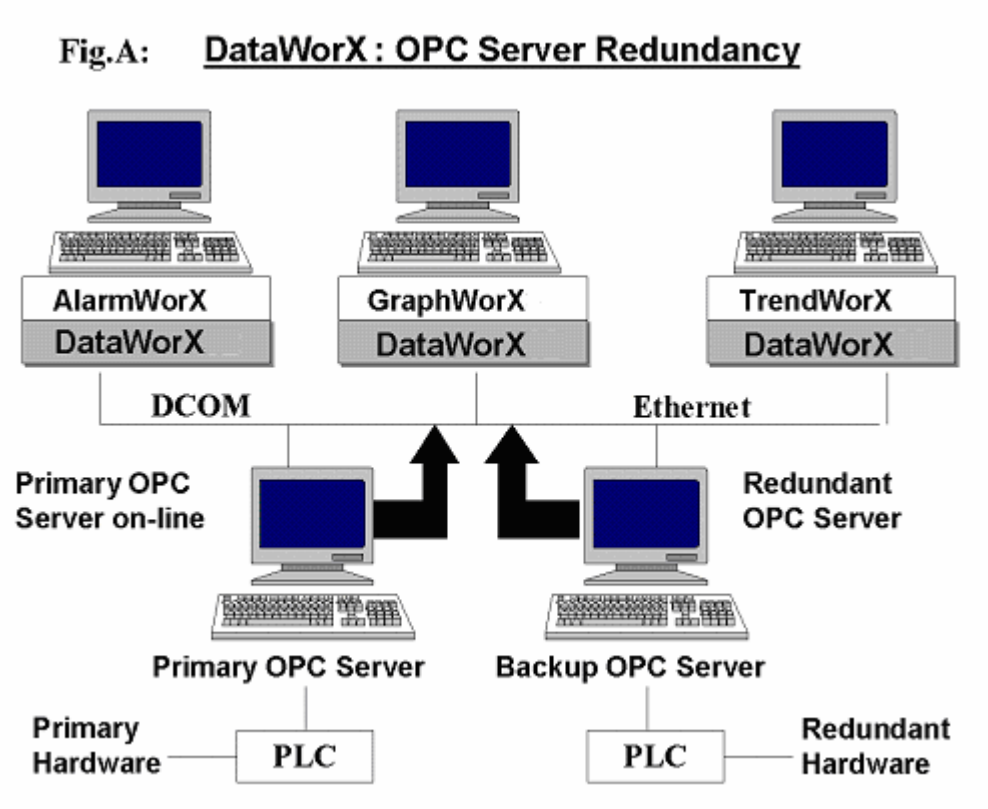


OPC Server Bridging

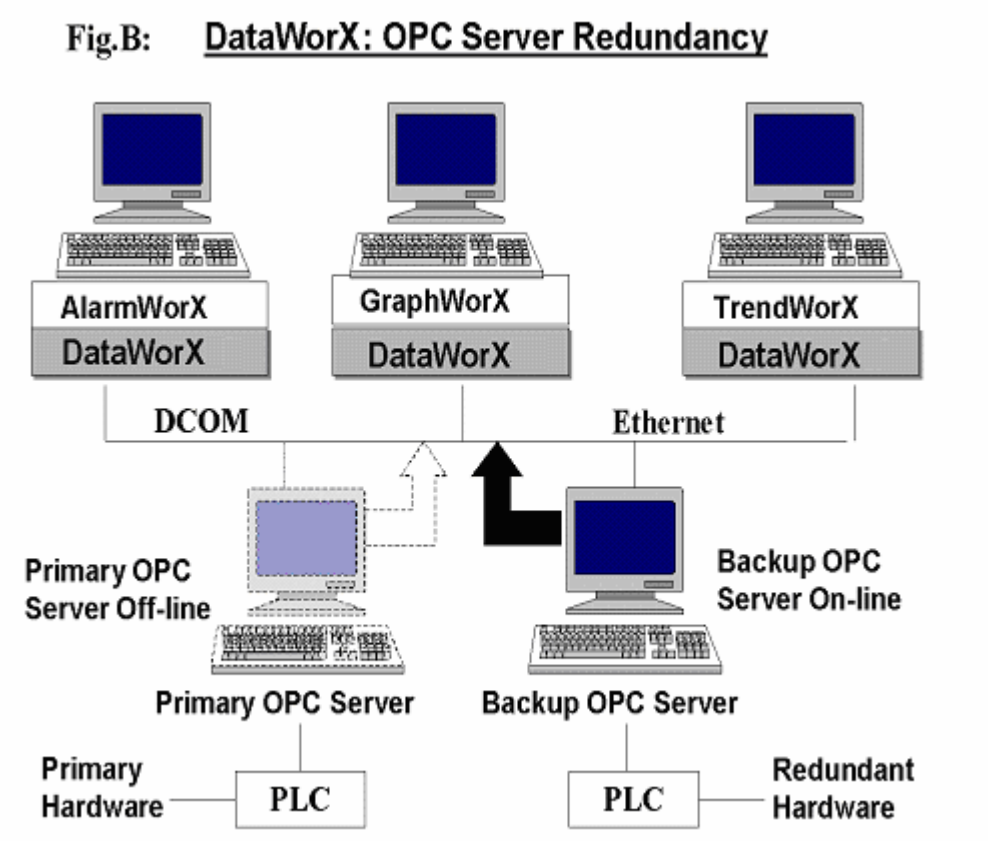
DataWorX OPC Server Redundancy

DataWorX provides 100 percent OPC server redundancy using OPC servers to any OPC client through the network. This means that users can designate alternative machines as backup servers should a designated Primary server go offline. DataWorX scans the OPC server status and switches to the Backup node in case of Primary node failure. This means that, once a Primary server does go offline, DataWorX will default to the Backup server or servers in the order in which the backup servers were designated. A special digital tag is provided to start events in case of a switchover from the Backup to the Primary server. If the **Automatic Switch Back to Primary Server** option is selected in the **Redundant Server Configuration** dialog box, DataWorX will default to the Primary server once it returns online.

Figure A and Figure B shown below illustrate DataWorX OPC Server Redundancy. For another example of redundancy, open the "DataWorX_Redundancy.gdf" display in the "Smar/ProcessView/Examples/ProcViewDEMO" directory.



Redundancy When the Primary OPC Server Is Online



Redundancy When the Primary OPC Server Is Offline

1. Users must designate one OPC server as the "Primary" server in each set.

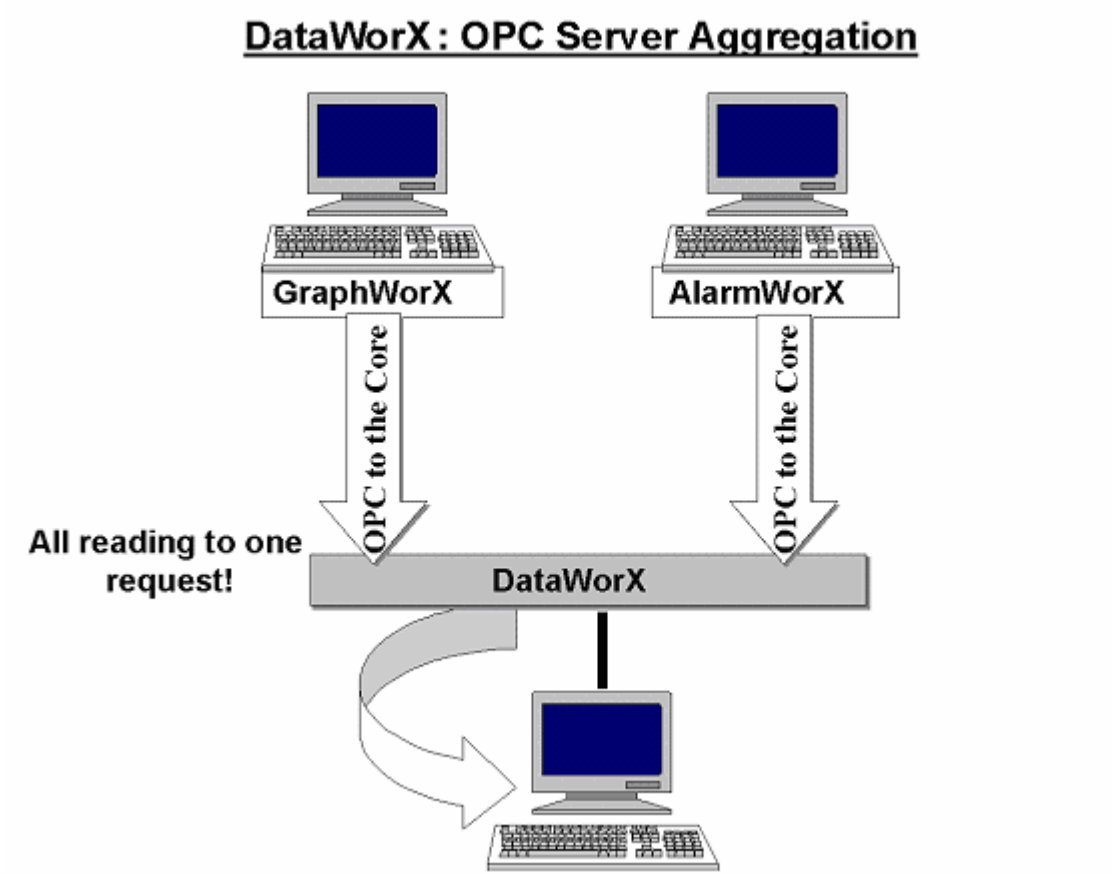
2. Users may designate one or more OPC servers as the "Backup" servers in each set. (This number of servers is not restricted by DataWorX itself; rather it is limited only by system resources). If more than one backup server is specified, they should be ordered (2nd, 3rd, 4th, etc.) The user will see a message outlining the details of the discrepancies and is allowed to either accept it as is, or permit reconfiguration.

3. The various OPC client applications request data from DataWorX, rather than from the OPC server directly. This way if a Primary OPC server failure occurs (due to any number of conditions), an automatic switchover to the Backup OPC server occurs.

DataWorX OPC Aggregation

Often in very large projects, several OPC client applications request the same points from an OPC server. For example, GraphWorX may need to display a tank level value, and AlarmWorX may need to monitor and alarm that same value. This may increase the load of the OPC server, as it now has to provide the same data more than once. Thus, when multiple clients request data from an OPC server, DataWorX monitors the OPC server and aggregates the data to the requesting clients.

Often it is desirable to optimize the work performed by the lower-level I/O servers (for example, greater throughput can be achieved). DataWorX can serve as a "middle-man" between clients and servers and assist in this optimization process. This is beneficial especially with remote servers over the network. The figure below illustrates the DataWorX OPC client request aggregation. For another example of aggregation, open the "DataWorX_Aggregation.gdf" display in the "Smar/ProcessView/Examples/ProcViewDEMO" directory.

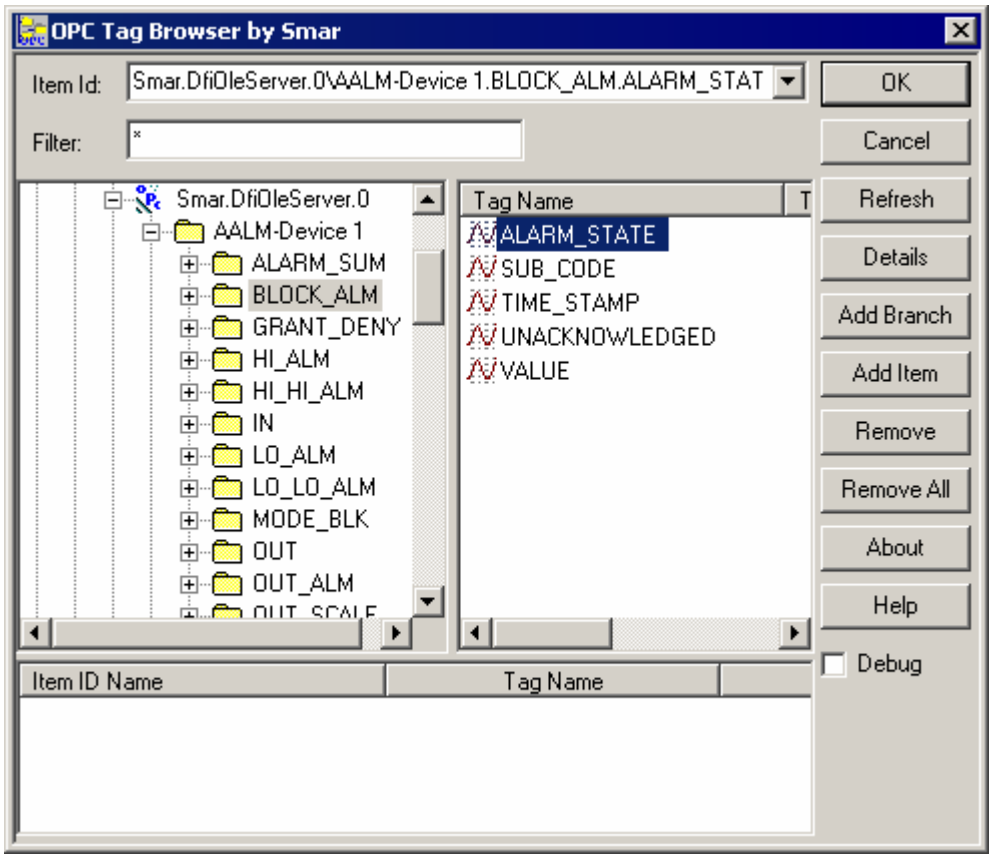


OPC Server Aggregation

Example of OPC Server Aggregation

It is possible to use the OPC Server Aggregation feature through the OPC Universal Tag Browser from other clients. The following example illustrates how it is possible to use the Aggregation feature from within the client application, in this case SMAR GraphWorX:

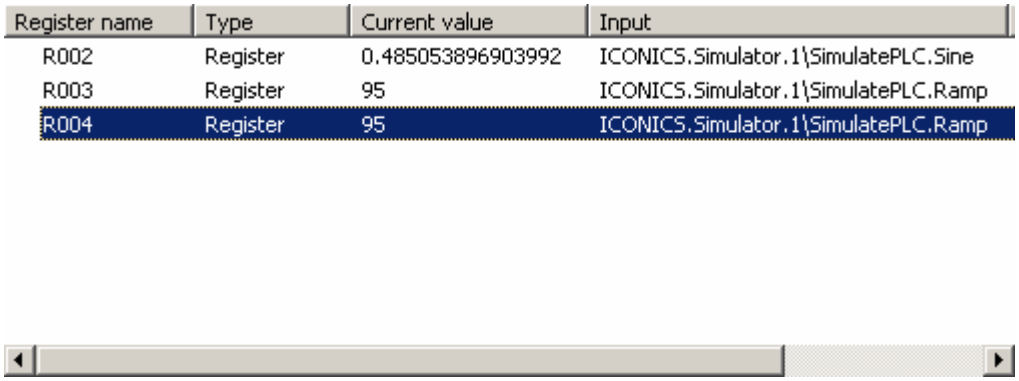
1. Open GraphWorX (the client). Open the **Property Inspector,** and then click the **OPC Tags** button to display the **OPC Universal Tag Browser,** as shown below.



OPC Universal Tag Browser

2. Click on the **Data Access** tree, and then select **Smar.DataWorX.1** server. Then select **My Computer,** and then the server name (in this case Smar.Simulator) to display your chosen register with tag names. Select a tag, and then click **OK.**

3. Within your selected display in runtime mode, you can view how instead of having each client each send its own request to the server, DataWorX organizes and optimizes the requests itself, thus relieving the server of a potentially large communication load, as shown below.



DataWorX Aggregation

Global Variables

Many clients require a common variable to share values. DataWorX makes it possible to define many variables that are accessible to many clients simultaneously. The DataWorX registers can be used as global variables. These variables can also act as aliases for clients.

DataWorX Registers

In the various client applications, the user can configure global variables for use within that particular application, or possibly only within a particular document for that application. One of the key functions of DataWorX is to provide a mechanism for defining variables and making them available to *all* ProcessView clients and OPC clients. DataWorX uses the concept of **registers** to achieve this.

Using Registers

Global variables are realized through registers. The "Global Variable" name is simply the register's name. It can be given values through OPC, expressions, constants, or even VBA. OPC server data bridging is realized through these registers by assigning the input of a register to one OPC data point, and assigning the output to different (even multiple!) OPC data points.

Register Names

Each register has a unique name composed of only numbers, letters, and the underscore "_" character. DataWorX verifies the names are indeed unique, prompting the user if this is not the case. The **Register name** field does not accept spaces. paces are disabled intentionally because the register names are used by OPC clients as tag names. The OPC spec allows for using spaces, but it is not recommended.

An **Alias Register** is a register that is used to keep the alias value and is of string data type. The user can access it like an ordinary register. The [[and]] are used to expand the alias, i.e. to replace the name of the alias with its value. For example:

Register 'REG' = 10; Alias Register 'MODBUS' = "SMAR.ModbusOPCServer".

'DataWorX\REG' = 10

'DataWorX\MODBUS' = "SMAR.ModbusOPCServer"

'DataWorX\[[MODBUS]]Dev1.Tag1' = SMAR.ModbusOPCServer\Dev1.Tag1 (value)

DataWorX registers are accessible through the OPC Universal Tag Browser.

DataWorX registers are accessible from VB using the "GetRegister" function of the Automation interface.

The count limit is not restricted by DataWorX itself, but is limited by system resources.

Register Inputs

A register has one input (or source) to define its content.

- An input may be assigned to an expression that, when resolved, defines the register's content.
- An input may be assigned to **None** to create a global variable. (Initial value is selectable.)

- An input may be assigned to an OPC data point (including the value itself, quality, and time stamp), using the OLExpress naming convention.
- An input may be assigned to the output of another DataWorX register.
- Besides OPC inputs, registers, and expressions, conditions can also be used as register inputs. The condition itself is connected to multiple OPC items or registers. One of these inputs is chosen depending on the selected criteria. Using conditions as register inputs is particularly useful when used in conjunction with switch aliases.

Register Outputs

When the value of a register's input changes, it is written to all of the outputs assigned to that register (if any).

- An output may be assigned to an item of an OPC server. Thus, the register writes values to the server.
- An output may be assigned to the input of another DataWorX register.
- A register may be designated as "Read-Only." The register will still write to its outputs, but the value can only come from an input, such as another OPC server or an expression, not from an OPC client. If this is the case, clients may only view its contents.

Registers, aliases, and other objects in DataWorX can be hierarchically organized into groups. This hierarchy can be viewed as a tree structure when using the Tag Browser under the Registers tree.

DataWorX Aliases

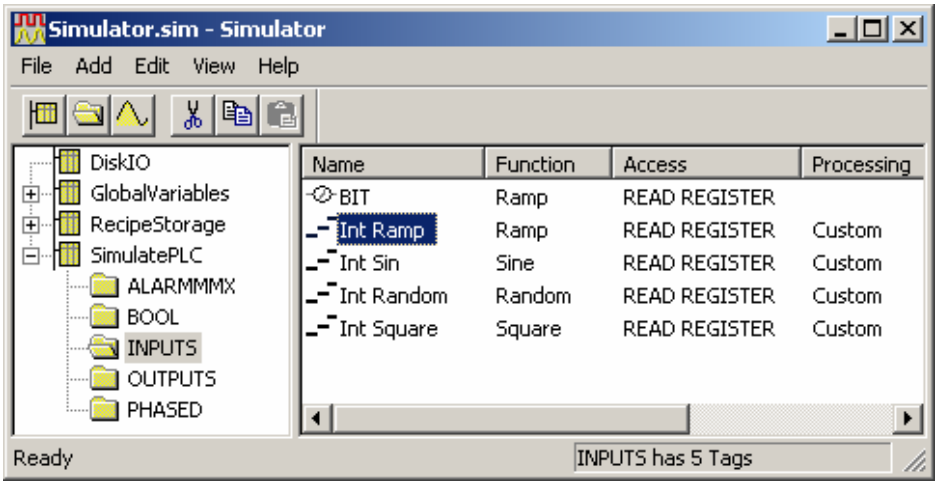
DataWorX contains a mechanism for defining aliases. **Aliases** are symbols that are expanded to strings, known as "alias values," during runtime. Aliases are enclosed in double brackets "[[" and "]]".

- Every alias must be defined before it is first used (that is, before the first item is requested with this alias). Otherwise the item name containing the alias is treated as invalid.
- If any name is syntactically valid and contains only aliases that have already been defined, it is treated as valid, even though after expanding aliases the name may not correspond to any existing item.
- The resolution of these aliases ("alias values") can change during runtime mode. When an alias changes, the items having the alias in their names stay valid. They simply change reference to another item after the change is made.

Aliasing Example

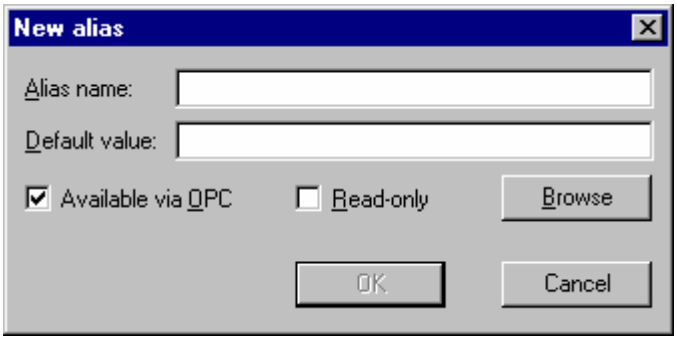
The use of aliases as global variables is illustrated by the following example.

1. Enter tags in the register using the SMAR OPC Simulator, as shown below.



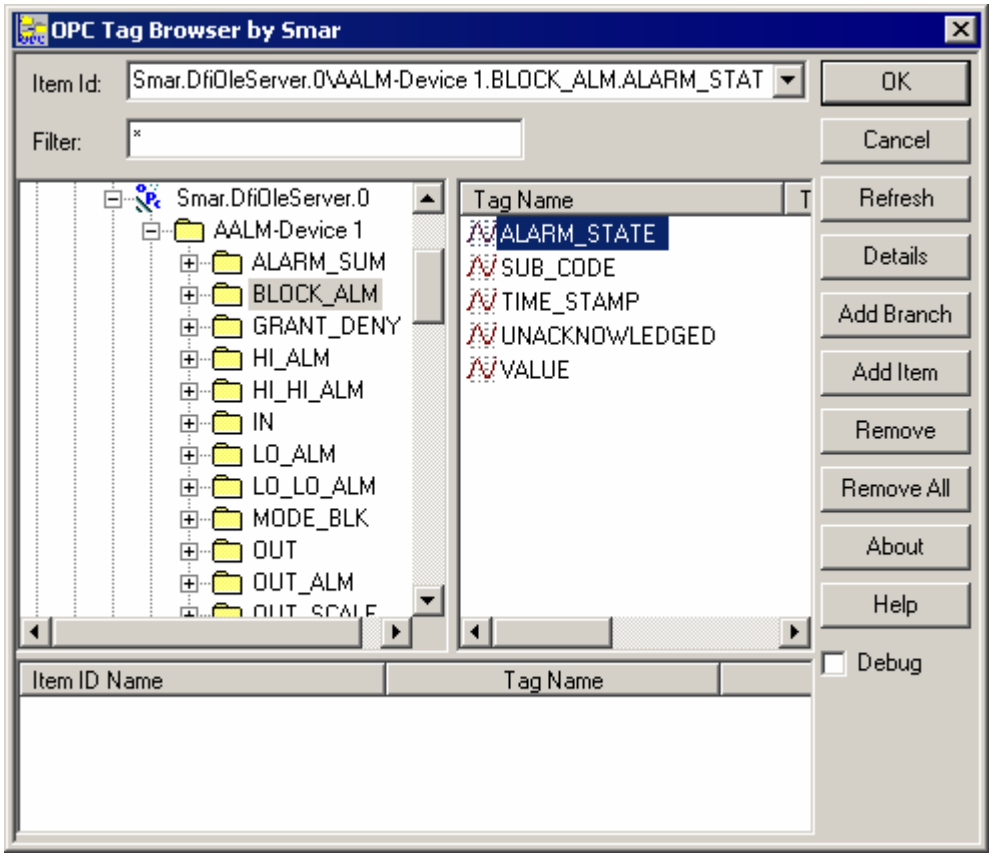
OPC Simulator

2. Add an alias to the register by selecting the **New Alias** from the **Alias** menu in DataWorX. This opens the **New Alias** dialog box, shown below.



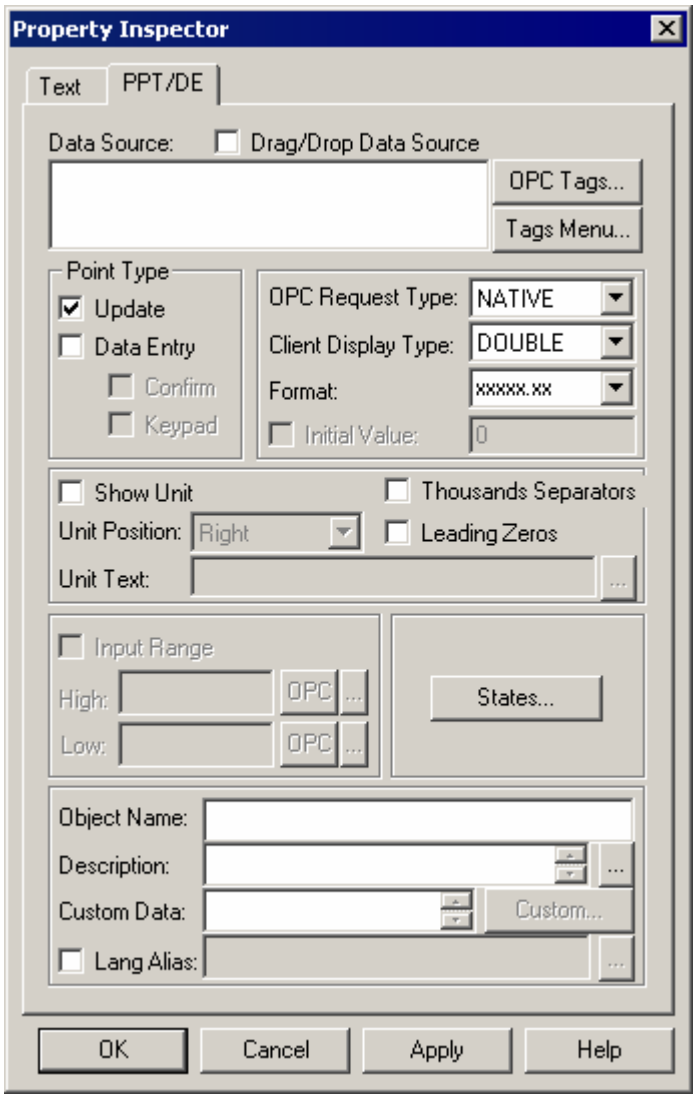
Creating a New Alias

3. Enter an **Alias name.** Click the **Browse** button to open the **OPC Universal Tag Browser.** Select a tag, as shown below. The tag is displayed in the **Default value** field in the **New alias** dialog box, as shown above.



Selecting a Tag

4. Open a client application, such as GraphWorX, and open the **Property Inspector,** shown below. To change the value of the previously defined alias from within GraphWorX in the Property Inspector, check the **Data Entry** check box, specify an **OPC Request Type,** and change the number in the **Format** field so that the full tag name fits. You can also use the Tag Browser to browse for individual tags.



Property Inspector

The Alias will give the value "pointed to" by the register value.

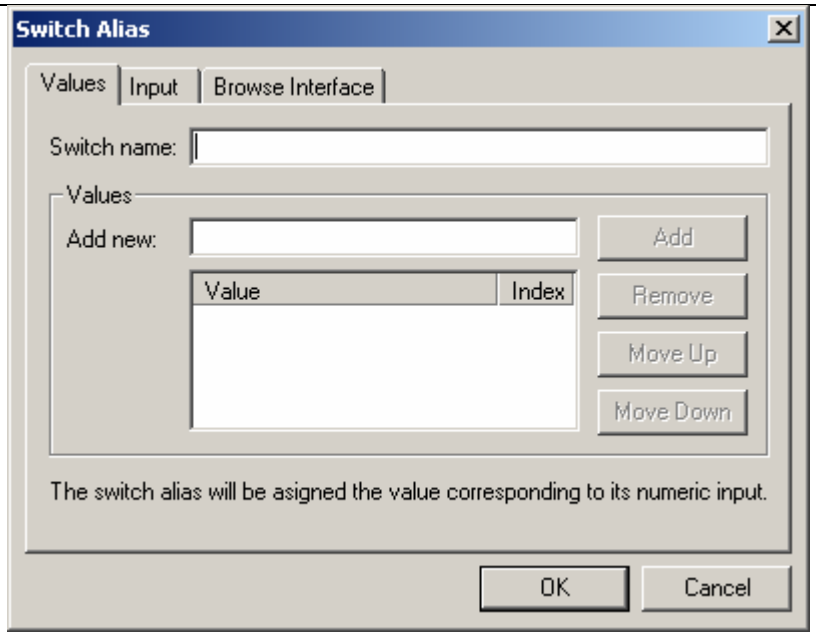
5. Within the client, (in this example GraphWorX) during runtime, you can edit the tag so that the alias refers to another tag and updates the values being received.

Switch Aliases

The **switch alias** is a special kind of alias. Unlike a regular alias, it has a numeric input. It contains a predefined set of values. The value of the alias is the one of the predefined set that corresponds to the input value.

Creating a Switch Alias

To create a switch alias:



Switch Alias Dialog Box: Values Tab

1. Select the **Add** from the **Switch** menu in DataWorX. This opens the **Switch Alias** dialog box, shown below.

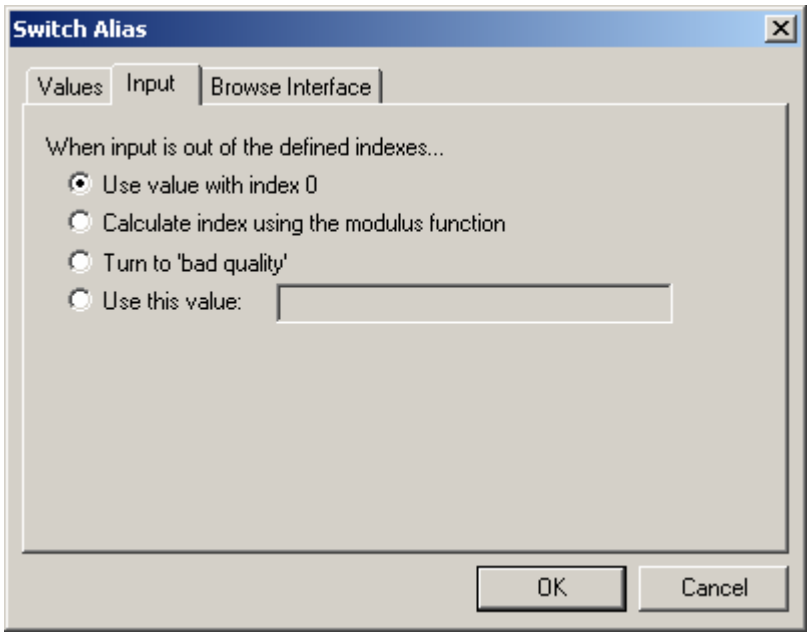
2. Use the **Values** tab to define the name of the switch alias and its values.

3. The name of the switch alias in the **Switch Name** field must meet the conditions for register name (unique, consisting of letters, numbers, and underscores). The possible values of the alias are listed in the list control below.

4. Use the **Add New** field and the **Add** button to add new values. Use the **Move Up, Move Down,** and **Remove** buttons to change the order of the values.

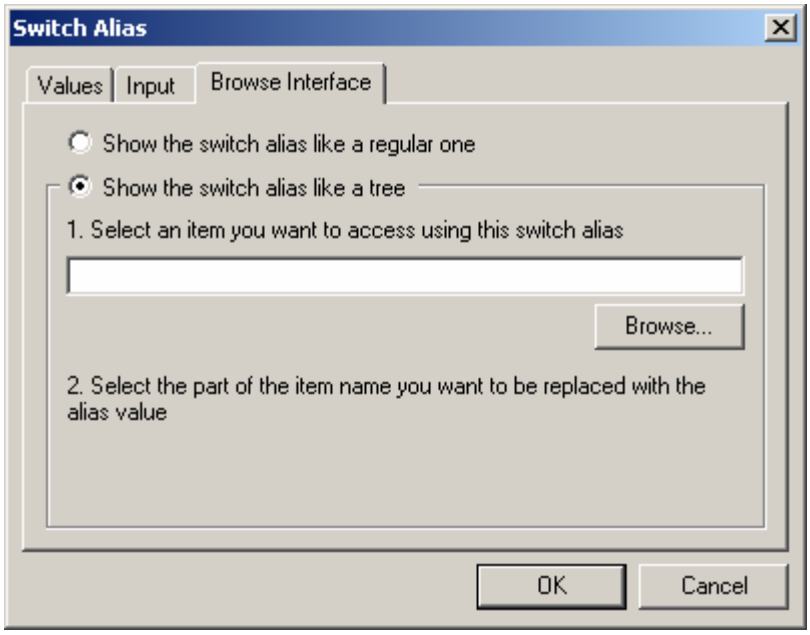
5. To edit a value, simply click it in the list control. The corresponding numbers for particular values are listed in the **Index** column.

6. When the input is outside the range of the defined indexes, you can select an input type using the **Input** tab of the **Switch Alias** dialog box, shown below. For example, suppose you have defined four values for the switch alias; thus, their indexes are "0," "1," "2," and "3." The **Input** tab allows you to define what should happen when the switch alias's input is less than "0" or greater than "3." The value with index "0" can be used, or the MODULUS function can be used. It can indicate "bad quality." You can also specify a value.



Switch Alias Dialog Box: Input Tab

7. You can choose how to show the switch alias using the **Browse Interface** tab of the **Switch Alias** dialog box, shown below. If you choose to show the switch alias like a tree, first select an item you want to access using the switch alias. Then select a part of the item name you selected in Step 1. This selected part is the one you intend to replace with the alias.

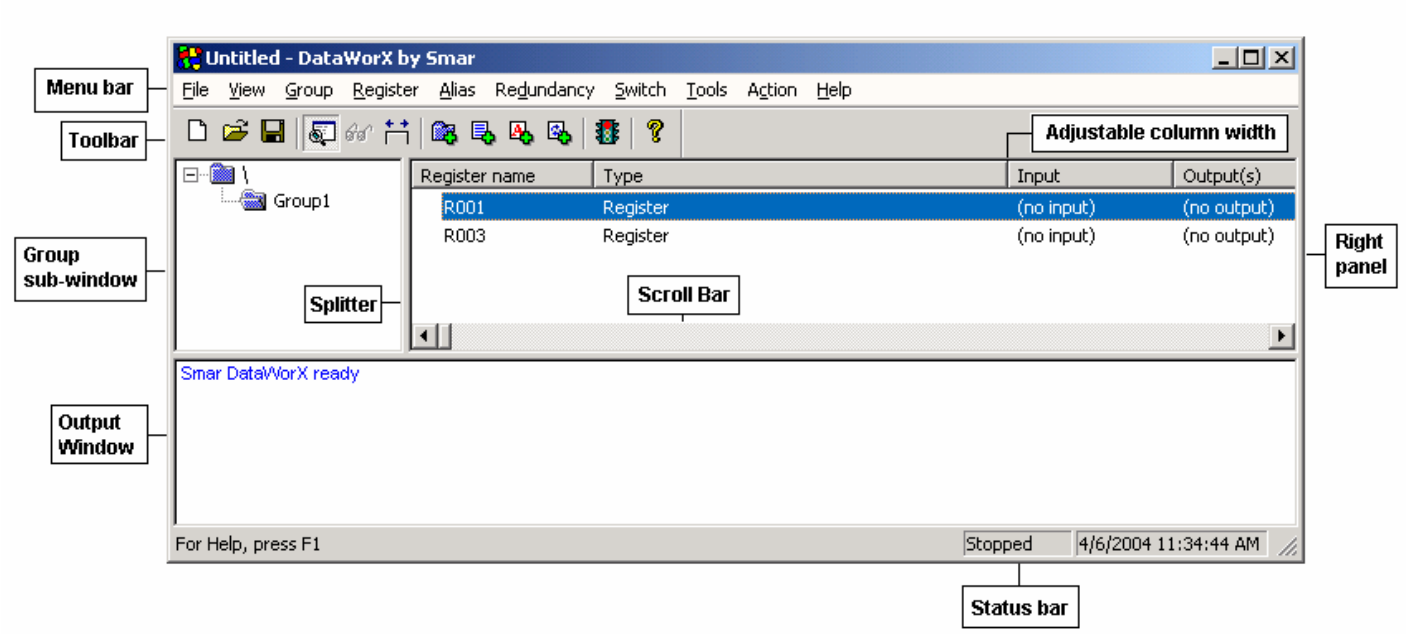


Switch Alias Dialog Box: Browse Interface Tab

User Interface

Introduction

The User Interface for the DataWorX application contains a splitter window with multiple views. Alternatively, the views can be switched one by one. It is possible to open up just one "project" file at a time within the main application. Within this one project file, using the Main menu bar and the Toolbar at the top of the screen, it is possible to display subwindows.



DataWorX Screen

The bar heading the top pane of the splitter screen lists in column format the items that are displayed in the top window: **Register Name, Type, Input** and **Output** and **Scan Rate** in the Registers view; and lists the **Alias Name, Default value, OPC Available,** and **Read-Only** in the Aliases view. The bottom pane of the splitter window, which is called the status window or the output window, defines the status of DataWorX, for example:

- "Smar DataWorX is ready."
- "DataWorX is started"
- "DataWorX is stopped"
- "Client is connected"
- "Client is disconnected"

You can toggle between views of the screen with the output window at the bottom or without the output window by clicking the appropriate icon on the toolbar or selecting the Output command on the **View** menu.

This section explains the various features available from the menu bar, toolbar, and status bar that allow you to manage and use DataWorX easily and efficiently.

DataWorX allows you to start the application with a specified file loaded.

Note
When closing a project, you cannot load another project file while a client is connected.

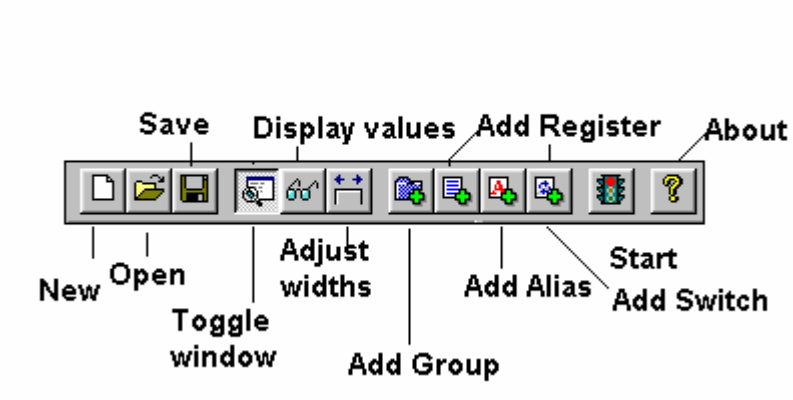
Screen Features

The DataWorX screen contains a main menu bar at the top with the following functions: The menu bar contains the following menus:

- **File**
- **View**
- **Group**
- **Register**
- **Alias**
- **Redundancy**
- **Switch**
- **Tools**
- **Action**
- **Help**

These menus are discussed in the following sections.

Along with the toolbar shown below, the menus provide the necessary options to configure and use DataWorX.



DataWorX Toolbar

File

The **File** menu contains the following commands:

- New - CTRL + N
- Open - CTRL + O
- Save - CTRL + S
- Save As
- Import CSV
- Reimport CSV
- Export CSV
- Re-export CSV
- Exit

These functions allow you to create a new DataWorX (.dwx) file, open an existing DataWorX project file, and save a DataWorX file.

In addition, you can Import or Export .csv files using the **File** menu. Selecting **Import CSV** from the **File** menu opens a dialog box that allows you to browse for a .csv file to import. Selecting **Export CSV** from the **File** menu opens the **Save As** dialog box, which allows you to browse for a .csv file to export.

View

The **View** menu contains the following commands:

- Toolbar
- Status bar
- Output
- Registers
- Aliases
- Monitor values
- Adjust column widths
- Statistics
- Select language
- Output window font

The **Adjust Column Widths** option changes the widths of the columns in the Aliases and Registers views so that everything in the columns is visible. You can also click the **Adjust Widths** button on the toolbar.

Selecting the **Output Window Font** command on the **View** menu opens the **Font** dialog box, which allows you to change font, size, and style of the text in the output window (bottom pane).

Toolbar/Status Bar

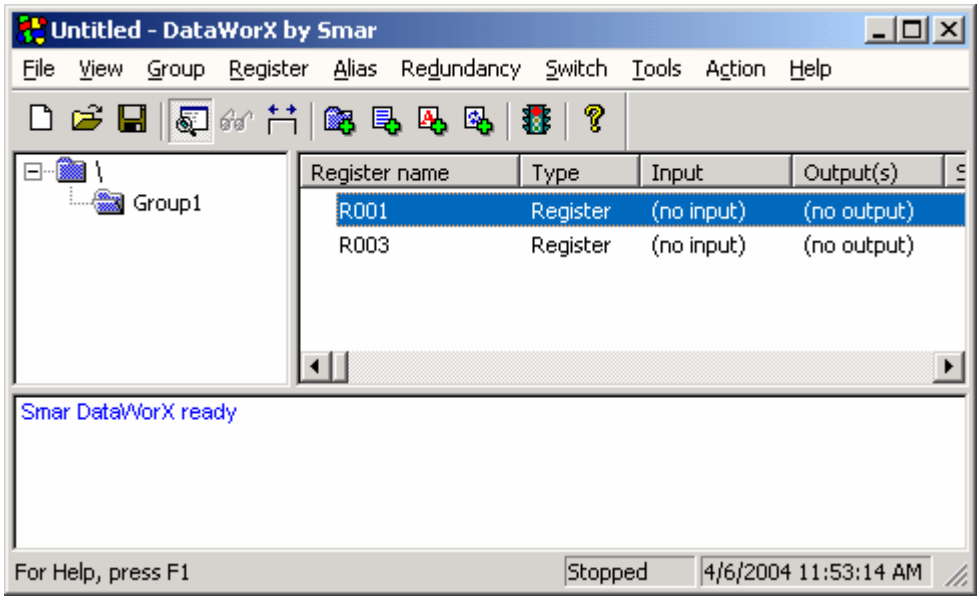
You can select the **Toolbar** command from the **View** menu to view the toolbar, or deselect the **Toolbar** command to hide the toolbar. Similarly, you can select the **Status Bar** command on the **View** menu to view the status bar, or deselect the **Status Bar** command to hide the status bar.

Output

Selecting the **Output** command from the **View** menu, or clicking the **Toggle Output Window** button on the toolbar, shown at left, displays the **Output** window in the lower pane of the screen.

Registers

To view the registers, select **Registers** from the **View** menu. The top pane of the splitter screen displays the **Register Name, Type, Input, Output,** and **Scan Rate** in column format, as shown below.



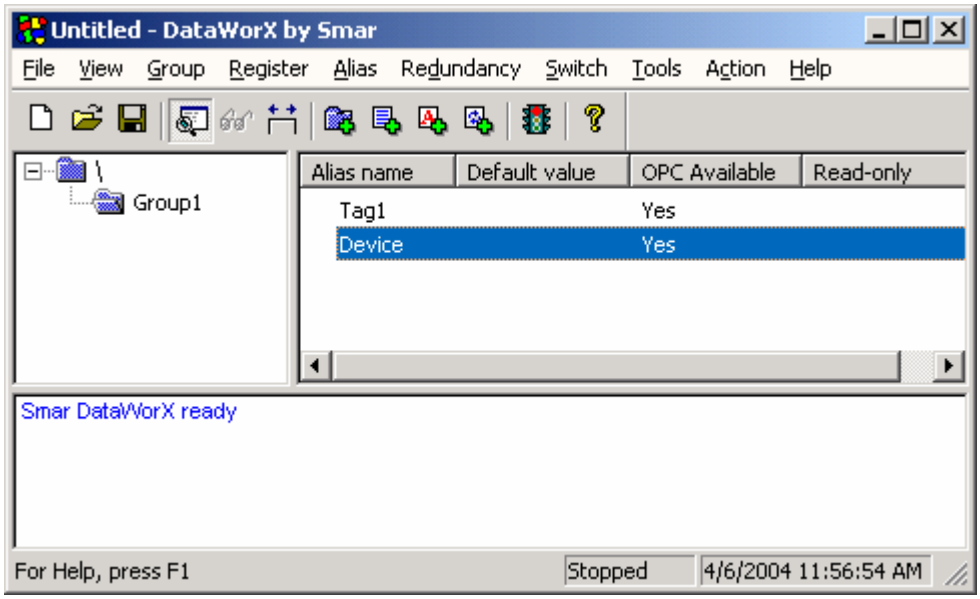
DataWorX Screen: Registers View

Registers View Columns

Register Name	Displays the name(s) of the register(s).
Type	Can be either Register or Status flag on a type of an alias (i.e. alias, switch alias, or redundancy alias). To see aliases as a register, check the appropriate option in the options dialog. The status flag is a special kind of register that is used with redundancy and that tells the user whether a primary or backup node is used.
Input and Output	Displays the names of OPC tags or registers that are connected to the selected register as input or output(s), respectively.
Scan Rate	Indicates the scan rate of the input OPC tag. This applies only to registers with input connected to an OPC tag or expression.

Aliases

To view aliases, select **Aliases** from the **View** menu. The top pane of the splitter screen displays the **Alias Name, Default Value, OPC Available,** and **Read-Only** in column format, as shown below. The **OPC Available** column indicates whether an OPC server is available or not. The **Read-Only** column indicates whether the file has a read-only attribute.



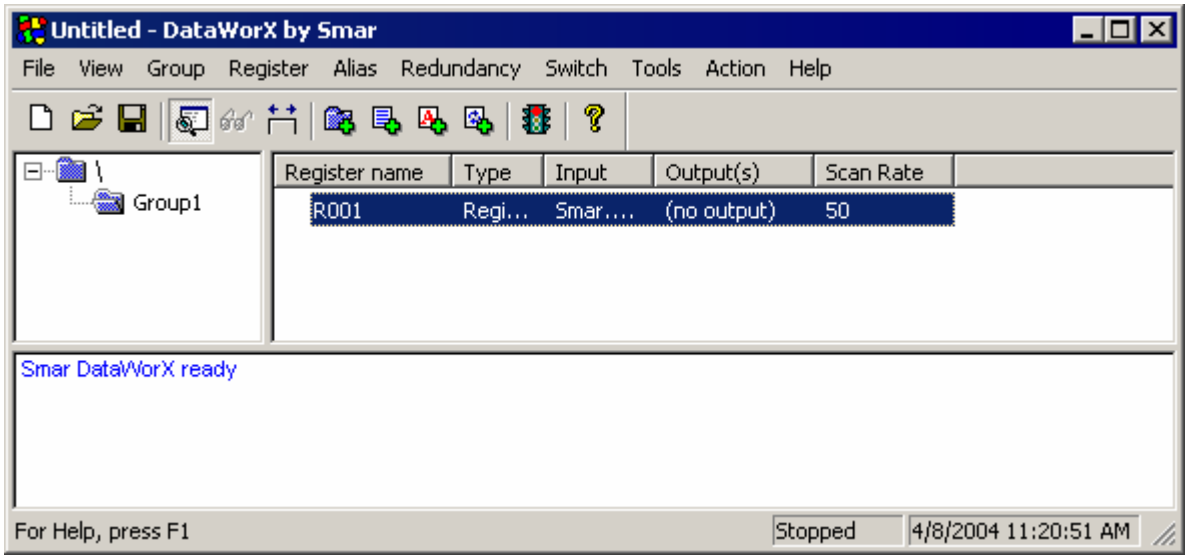
DataWorX Screen: Aliases View

Aliases View Columns

Alias Name	Displays the name(s) of the register.
Default Value	Displays the value to which the alias will be initially set.
OPC Available	Specifies whether the alias can be accessed (for reading and/or writing) through the OPC interface.
Read-Only	Specifies whether the item is read-only or writeable.

Monitor Values

Clicking the **Display Values** button on the toolbar, or selecting **Monitor Values** from the **View** menu, displays the current value of the tag in both the Registers view and the Aliases view, as shown below.

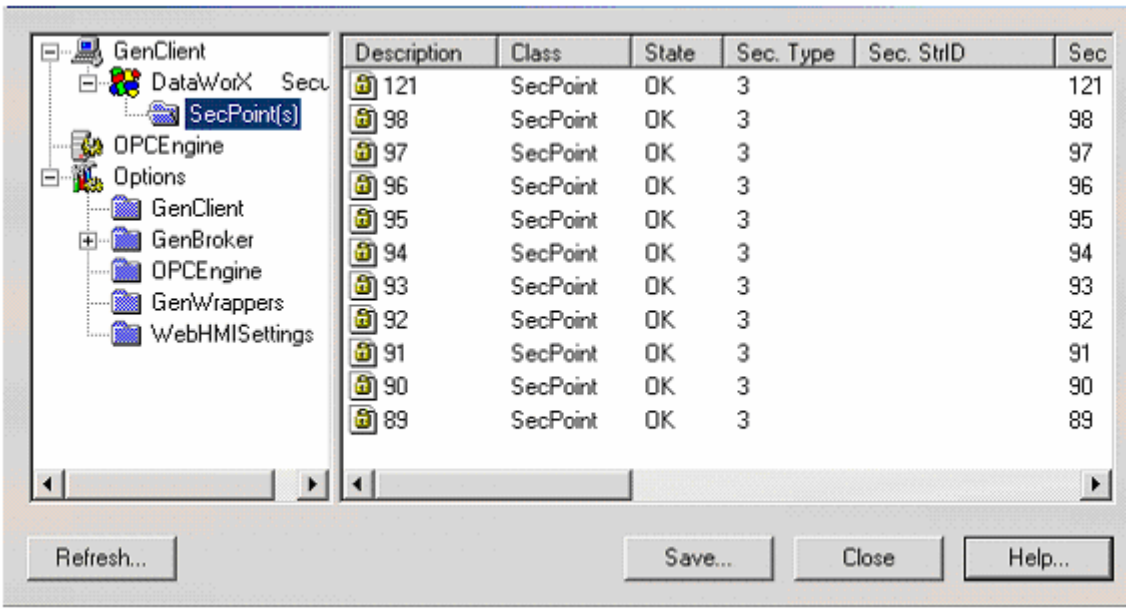


Screen Displaying Current Value

As shown in the figure above, the current value monitor now indicates "bad quality," which is displayed in parentheses.

Statistics

Selecting the **Statistics** command from the **View** menu opens the **Runtime Statistics** dialog box, shown below, which displays all the statistics relevant to the available OPC Servers in use, including **Name, Start Time, State, Vendor,** and **Version.** Click the **Help** button for additional information.



Runtime Statistics

Select Language

Choosing the **Select Language** command from the **View** menu opens the **Select Language** dialog box, shown below, which lists all of the available languages in alphabetical order.

Select Language Dialog Box

To select a language:

1. Select a language from the box on the left.

2. Select from the options available on the right, which are described in the table below.

3. Click **OK.**

Select Language Parameters

List	Check one of the radio buttons under the List field: English, Localized, or Native.
Installed Locales Only	Checking this check box limits your choice to languages installed on your machine..
Available Language Translations Only	Checking this check box limits your choice to languages available for translation.

Group

Using the **Group** menu, you can organize registers into groups. The **Group** menu contains the following commands:

- Add
- Remove

Selecting **Add** from the **Group** menu opens the **Group Properties** dialog box, shown below. Enter the name of the group, and then click **OK.**

You can also click the **Add Group** button on the toolbar to add a new group.

Group Properties Dialog Box

The new group will be added to the tree view in the upper left pane, and the hierarchical structure of the group can be seen in the Tag Browser when browsing DataWorX.

To edit the name of a group, right-click the group name in the tree view and select **Edit** from the popup menu. You can change the name directly.

To delete a group, right-click the group name in the tree view and select **Remove** from the popup menu. Or you can select the group and then choose **Remove** from the **Group** menu.

Registers

One of the key functions of DataWorX is to provide a mechanism for defining variables and making them available to *all* ProcessView clients and OPC clients. DataWorX uses the concept of **registers** to achieve this.

Using Registers

The "Global Variable" name is simply the register's name. It can be given values through OPC tags, expressions, constants, or even VBA. OPC server data bridging is accomplished by assigning the input of a register to one OPC data point, and assigning the output to different (even multiple!) OPC data points.

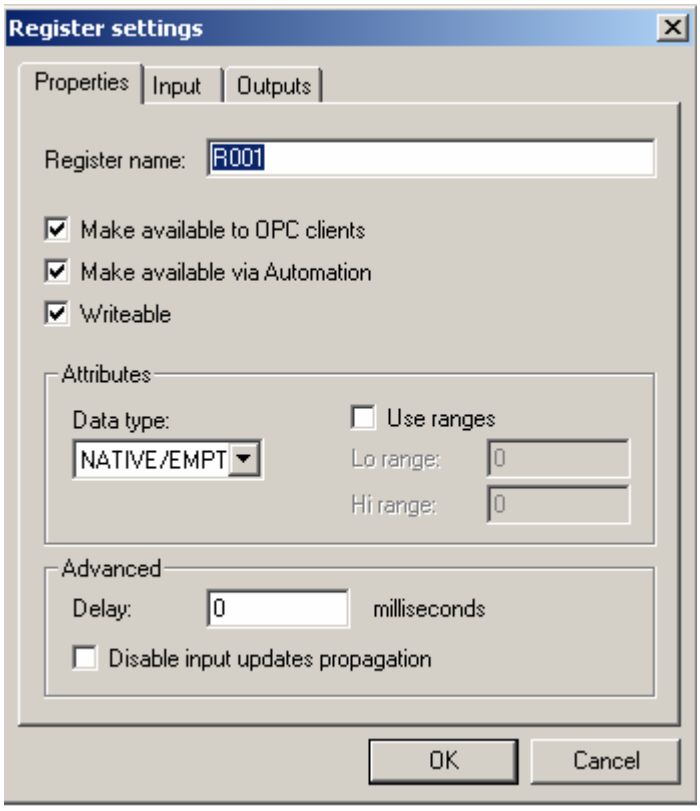
This section describes how to use the Registers functions on the DataWorX menu bar in order to connect more than two servers. For example, if you want a client PC to send a value from a Modbus OPC server to a Bristol OPC server, you can define the **Register Name** in the **Properties** dialog box, define the **Properties** for that register, set the **Input** to the Modbus OPC server's tag, and set the **Output** to the Bristol OPC server's tag for a Register Tag using the **Register Settings** dialog box. The **Register** menu in DataWorX contains the following commands:

- Add
- Edit
- Remove

Registering Running Instances
DataWorX now registers its dispatch pointer at GenRegistrar on startup and unregisters on exit. Dispatch pointers may be used by Visual Basic or other applications to control a running application. Now, a dispatch pointer to an instance of DataWorX running in both configuration mode and runtime can be obtained. In past versions of DataWorX, this was possible only in runtime.

Adding Registers

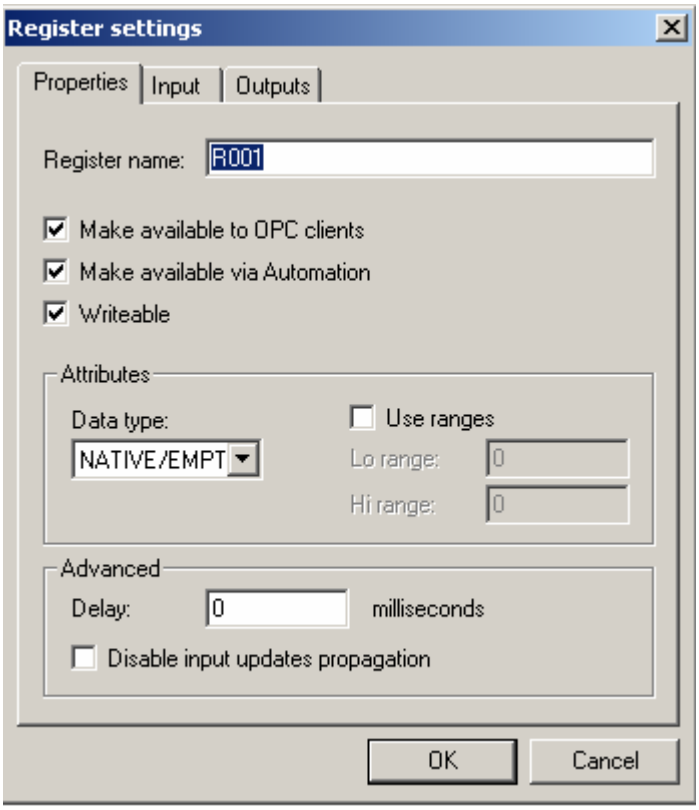
To add a register, click the **Add Register** button on the toolbar, shown at left, or select **Add** from the **Register** menu. This opens the **Register Settings** dialog box, shown below, which allows you to configure the **Properties, Input,** and **Output** settings as described in the following sections::



Register Settings Dialog Box

Register Settings: Properties Tab

The **Properties** tab in the **Register Settings** dialog box, shown below, allows you to set the following register parameters, as described in the table below.



Register Settings: Properties Tab

Properties Tab Parameters

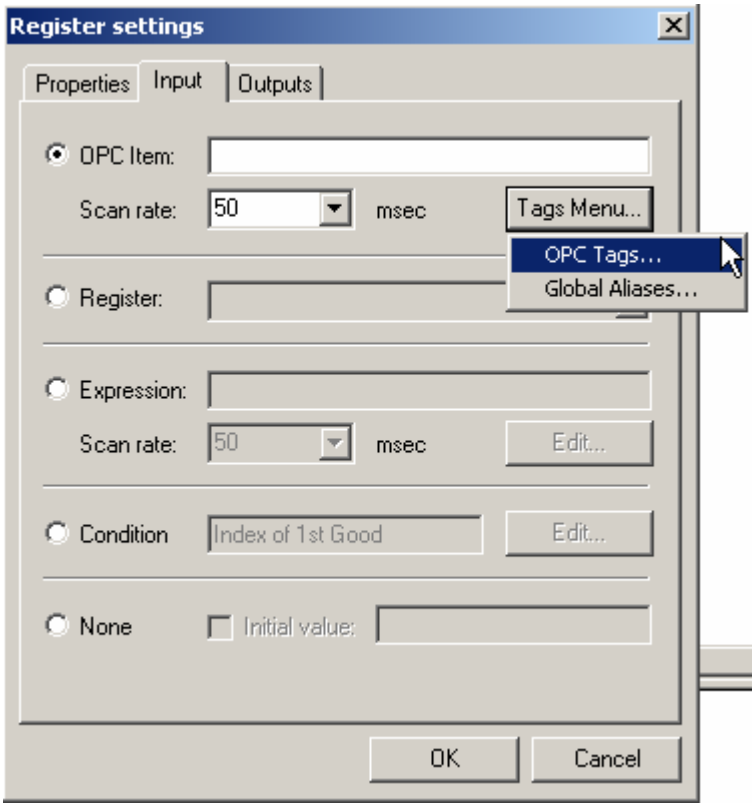
Register Name	A common tag name.
Make available to OPC clients	Checking this box makes the register available to OPC clients
Make available via automation	Checking this box makes the register available to VB applications so that it could be edited by a VB editor.
Writeable	Checking this box makes the register writeable. Uncheck the box for a read-only OPC client.
Data Type	Allows you to select from the following data types Native/Empty, Float, Double, Boolean, Byte, Word, DWord, Character, Short, Long, String. All data types available in GraphWorX are supported in registers for DataWorX.
Use Ranges	Checking this box enables the Hi and Lo ranges fields.
Hi Range	You can specify a Hi Range in the box provided.
Lo Range	You can specify a Lo Range in the box provided.
Delay	You can specify a delay time for the register (in milliseconds) in the box provided.

Disable OPC Propagation Support	When an OPC item is connected to a register as both input and output, a fast sequence of writes to that register may cause item value oscillation. (This may happen when two subsequent writes to the register are faster than the OPC item can perform the write operation. The acknowledge of the first write then overwrites the second written value already being stored in the register.) To avoid this behavior, Check the **Disable input updates propagation** check box in the register properties.

Note
By default, the data type and range information are obtained from the input of register.

Register Settings: Input Tab

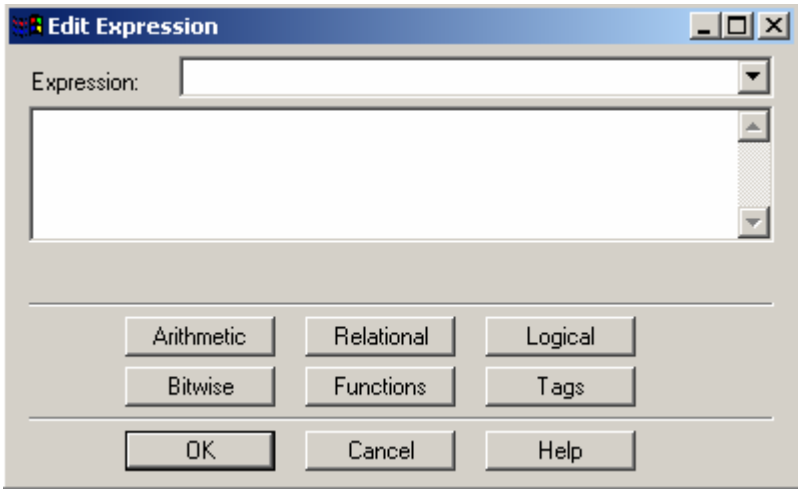
The **Input** tab in the **Register Settings** dialog box, shown below, allows you to set the following register parameters, as described in the table below.



Register Settings: Input Tab

Input Tab Parameters

OPC Item	Checking this radio button specifies that the input is an OPC item.
Item Name	Tag Name. You can type in an item name or browse for it by clicking the Browse button.
Scan Rate	Allows you to specify a scan rate in milliseconds.
Tags Menu	The **Tags Menu** button allows you to search for Tags under the OPC Servers. Clicking the **Tags Menu** button and selecting **OPC Tags** from the pop-up menu opens the OPC Universal Tag Browser. Selecting Global Aliases opens the **Global Alias Browser** (see below).
Register	Checking the Register radio button allows you to choose a register name from the drop-down list in the list box provided. This lists the registers defined in DataWorX. You may select an input coming from another register.
Condition	Besides OPC inputs, registers and expressions, conditions can be used as register inputs. The condition itself is connected to multiple OPC items or registers. One of these inputs is chosen depending on the selected criteria. Clicking the Edit button opens the Condition dialog box, which is described in the **Condition Inputs for Registers** section.
Expression	Selecting the Expression radio button allows you to specify an expression in the box provided. It also enables the Edit button. Clicking the Edit button displays the **Edit Expression** dialog box, shown below, which allows you to edit expressions using the Arithmetic, Relational, Logical, Bitwise, and Functions methods, as well as OPC tags and registers. The result of Expression includes a quality evaluation. The user may specify how the quality should be evaluated. Clicking the Registers button in the Edit Expression dialog box opens the Select Register dialog box, which enables you to select from a list of registers in DataWorX.

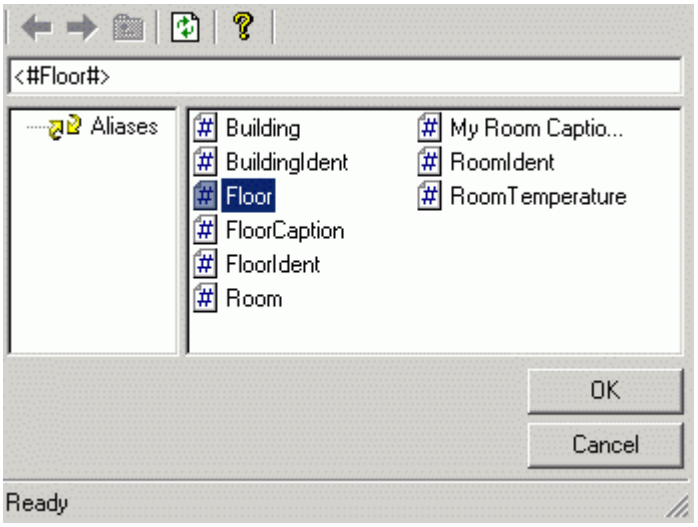


Edit Expression Dialog Box

None	Checking this button means no input will be provided to this register. This could be used when you want to write a value from the client and send it to many OPC servers at the same time. This is also the mechanism for creating a global variable.
Initial Value	Selecting this check box allows you to specify an initial value in the field to the right.

Clicking the OPC Tags button on the **Input** tab of the **Register Settings** dialog box and selecting Global Aliases from the pop-up menu opens the **Global Alias Browser,** shown in the figure below. Select a global alias from the Global Alias Browser, which includes all global aliases in the global alias database. This eliminates the need to manually type in the alias name. All global aliases that are configured in the Global Alias Engine Configurator are conveniently available to choose from inside the browser. The tree control of the Global Alias Engine Configurator is mimicked in the tree control of the Global Alias Browser. Select a global alias by double-clicking the alias name (e.g. "Floor" in the figure below). The alias name appears at the top of the browser, which automatically adds the <# and #> delimiters to the alias name. Click the **OK** button.

For more information, see the Global Aliasing Configurator Help documentation.



Selecting an Alias From the Global Alias Browser

Condition Inputs for Registers

In addition to OPC inputs, registers, and expressions, conditions can be used as register inputs. The condition itself is connected to multiple OPC items or registers. One of these inputs is chosen depending on the selected criteria (see below). The result of the condition (i.e. the value of the register having input set to this condition) may be:

- The value of the chosen input; the data type of the register will be the same as the data type of the input (by default).
- The zero-based index of the chosen input; the data type of the register will be a integer number in range 0..*N*-1, where *N* is the number of the condition inputs.
- The name of the chosen input; the data type of the register will be a string.

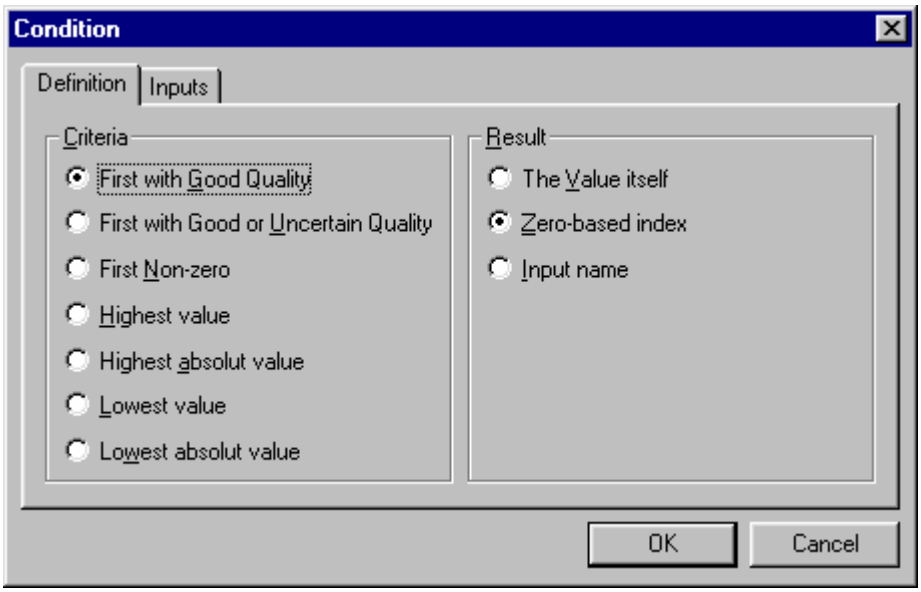
Possible criteria are:

- First with good quality
- First with good or uncertain quality
- First nonzero

- Highest value
- Highest absolute value
- Lowest value
- Lowest absolute value

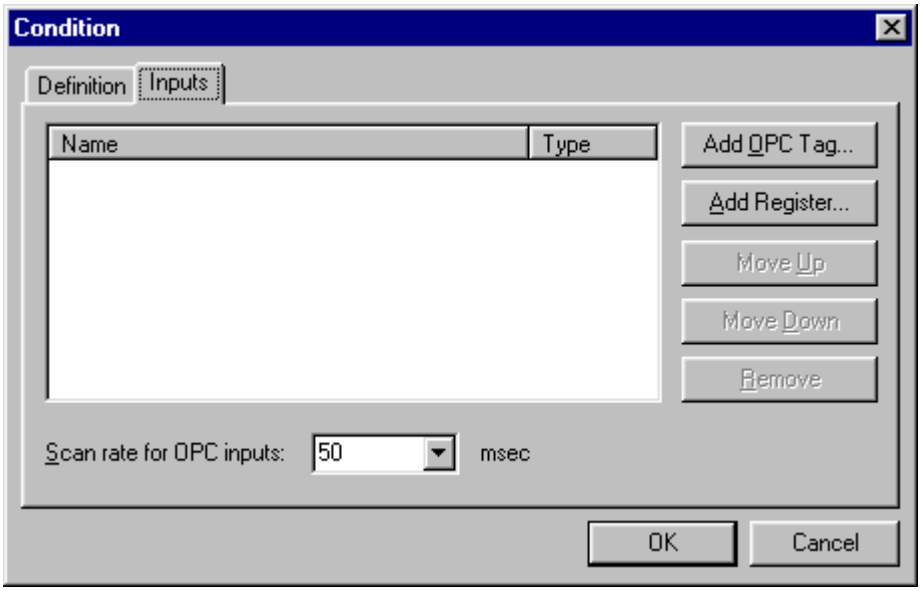
To define a condition input for a register:

1. Select **Edit** or **Add** from the **Register** menu. This opens the **Register Settings** dialog box.
2. Click the **Input** tab.
3. Select **Condition,** and then click the **Edit** button. This opens the **Condition** dialog box, shown below.
4. In the **Definition** tab, specify the criteria and results described above.



Condition Dialog Box: Definition Tab

The Input tab of the **Condition** dialog box, shown below, enables you to specify the Input(s) of the condition. You can select a tag from the OPC Universal Tag Browser by clicking **Add OPC Tag.** Clicking **Add Register** opens the **Select Register** dialog box, which enables you to select from a list of registers in DataWorX. You can also specify the scan rate for OPC inputs (in msec).

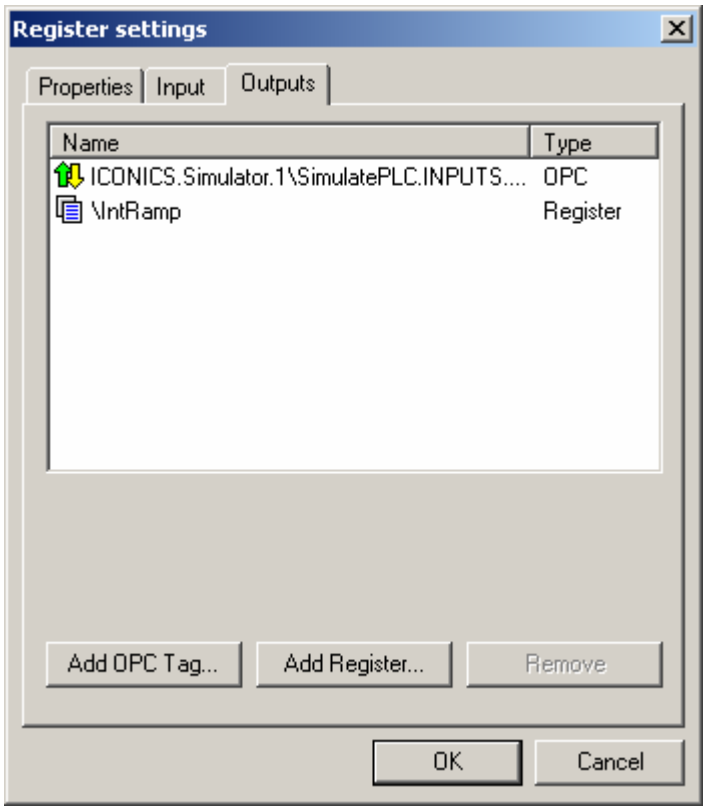


Condition Dialog Box: InputsTab

The conditions themselves may be useful, but their biggest advantage is in use with Switch Aliases.

Register Settings: Outputs Tab

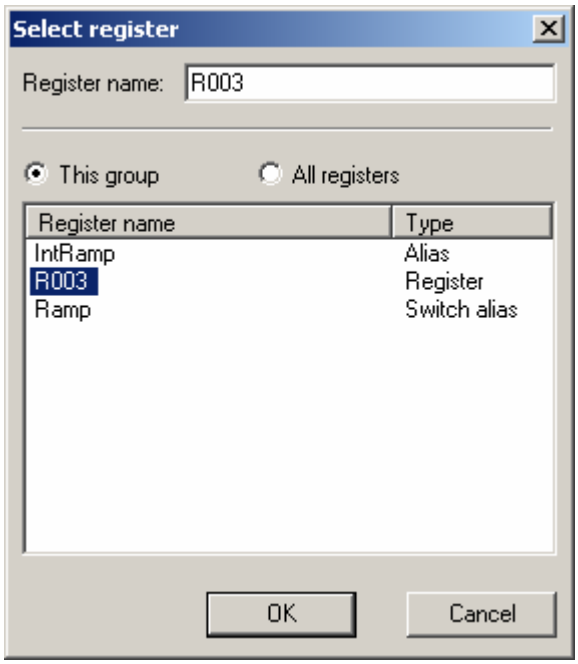
The **Outputs** tab in the **Register Settings** dialog box, shown below, allows you to set the following register parameters, as described in the table below.



Register Settings: Output Tab

Output Tab Parameters

Name	This column displays the name of the OPC tag and/or register.
Type	This column specifies whether the item is an OPC tag or a register.
Add OPC Tag	Clicking this button opens the OPC Universal Tag Browser, which allows you to select an OPC tag.
Add Register	Clicking this button displays the **Select Register** dialog box, shown below, which allows you to select a register name from the list of registers provided.



Select Register Dialog

Once you have configured the **Properties, Input,** and **Output** for the **Register Settings** dialog box, click **OK** and to add the new register.

Note
You may select more than one tag for the output.

Editing Registers

To edit a register:

1. Select a register under the **Register Name** column.
2. Select **Edit** from the **Register** menu to display the **Register Settings** dialog box, or right-click the register name and select **Edit** from the popup menu.
3. Configure the settings in the **Properties, Input,** and **Output** tabs, which have already been discussed in the previous section.
4. Click **OK.**

Removing Registers

To remove a register:

1. Select the Register you wish to remove under the **Register Name** column.

2. Select **Remove** from the **Register** menu, or right-click the register name and select **Remove** from the popup menu.

3. This will instantly remove the selected register from the database, and that register will disappear from the screen.

Right-Click Context-Sensitive Menus for Registers

Right-clicking on a register in the Registers view in configuration mode displays the popup menu shown below. You can either **Edit** or **Remove** the selected register. Right-clicking on a redundant node in the Registers view in configuration mode displays the same menu.

Right-clicking on an alias in the Registers view in configuration mode displays the popup menu shown below, which contains the same commands that are available on the **Aliases** menu. This menu allows you to edit the input and output for the selected alias.

You can select any one of the functions: to **Edit** the selected Alias, to **Edit as Register**, to **Change the Value** of the selected alias, or to **Remove** the selected alias.

Note: The **Edit as Register** command is only accessible when **Allow Editing Aliases As Registers** is selected in the **Options** dialog box.

Aliases

DataWorX contains a mechanism for defining aliases. **Aliases** are symbols that are expanded to strings, known as "alias values," during runtime. Aliases are enclosed in double brackets "[[" and "]]".

- Every alias must be defined before it is first used (that is, before the first item is requested with this alias). Otherwise the item name containing the alias is treated as invalid.
- If any name is syntactically valid and contains only aliases that have already been defined, it is treated as valid, even though after expanding aliases the name may not correspond to any existing item.
- The resolution of these aliases ("alias values") can change during runtime mode. When an alias changes, the items having the alias in their names stay valid. They simply change reference to another item after the change is made.

This section describes how to use commands on the **Alias** menu.

Main Requirements for Aliasing

1. Every alias must be defined before it is first used (that is, before the first item is requested with this alias). Otherwise the item name containing the alias is treated as invalid. DataWorX will provide a summary of invalid names to the user.

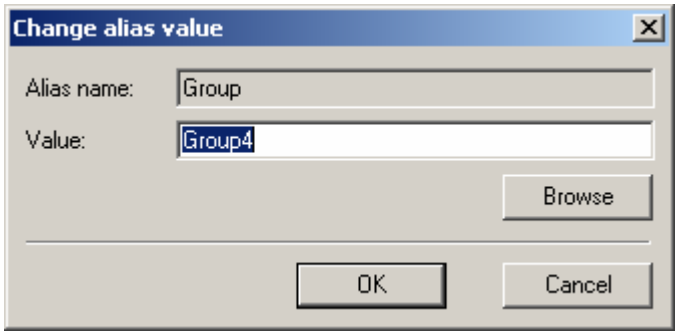
2. If the syntax of the name is valid and contains only aliases that have already been defined, it is treated as valid, even though after expanding aliases the name may not correspond to any existing item.

Changing Alias Values

To change an alias value:

1. Select an alias from the **Alias Name** column.

2. Select **Change Value** from the **Alias** menu. This opens the **Change Alias Value** dialog box, shown below.



Change Alias Value Dialog Box

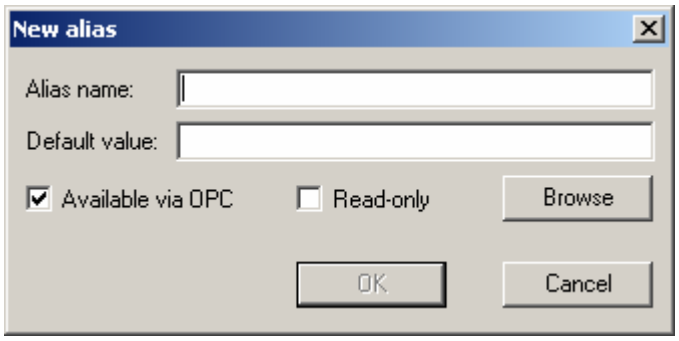
3. In the **Change Alias Value** dialog box, specify a new value in the **Value** field. You can browse for tags by clicking the **Browse** button, which launches the OPC Universal Tag Browser.

4. Click **OK.**

Creating a New Alias

To create a new alias:

1. Click the **Add Alias** button on the toolbar, or select **Add** from the **Alias** menu This opens the **New Alias** dialog box, shown below.



New Alias Dialog Box

2. In the **New Alias** dialog box, define the necessary parameters described in the table below.

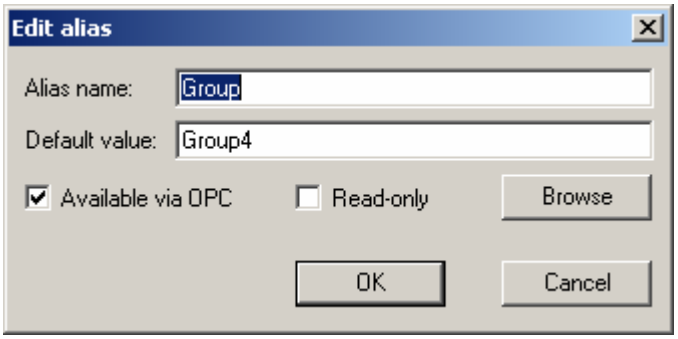
New alias name	Enter a common tag name.
Default value	Enter a default value.
Available via OPC	If checked, the data relevant to the new alias are available through an OPC server.
Read-only	Ensures that the new alias is read-only.
Browse	Clicking this button launches the OPC Tag Browser, which allows you to search the existing OPC Servers for tags.

3. Click **OK.**

Editing Aliases

To edit an alias:

1. Select an alias from the **Alias Name** column in the top pane of the splitter window.
2. Select Edit from the **Alias** menu. This opens the **Edit Alias** dialog box, shown below.
3. Define the necessary parameters. Enter the alias name and the default value. Select the option to make the Alias Tag available via OPC. You can make the item read-only if you choose to by checking the box provided.
4. You can browse the existing OPC servers for tags using the OPC Universal Tag Browser, which is launched by clicking the **Browse** button.
5. Click **OK.**



Edit Alias Dialog Box

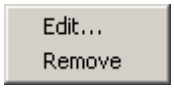
Removing Aliases

To remove an alias:

1. Select an alias from the **Alias Name** column in the top pane of the splitter window.
2. Select **Remove** from the **Alias** menu.
3. This instantly removes the selected alias name from the list of aliases.

Right-Click Context Sensitive Menus for Aliases

Right-clicking on an alias in the Aliases view in configuration mode displays the popup menu shown below. You can either **Edit** or **Remove** the selected alias.



Right-clicking on an alias in the Registers view in configuration mode displays the popup menu shown below, which contains the same commands that are available on the **Alias** menu. This menu allows you to edit the input and output for the selected alias.

You can select any one of the functions: to **Edit** the selected alias, to **Edit as Register**, to **Change the Value** of the selected alias, or to **Remove** the selected alias.

Switch Menu

The **Switch** menu contains the following commands for switch aliases:

- Add
- Edit
- Remove

This section describes the user interface for the switch alias. The switch alias is a special kind of alias. Unlike a regular alias, it has a numeric input. It contains a predefined set of values. The value of the alias is the one of the predefined set that corresponds to the input value. Refer to the **Switch Aliases** section for more information.

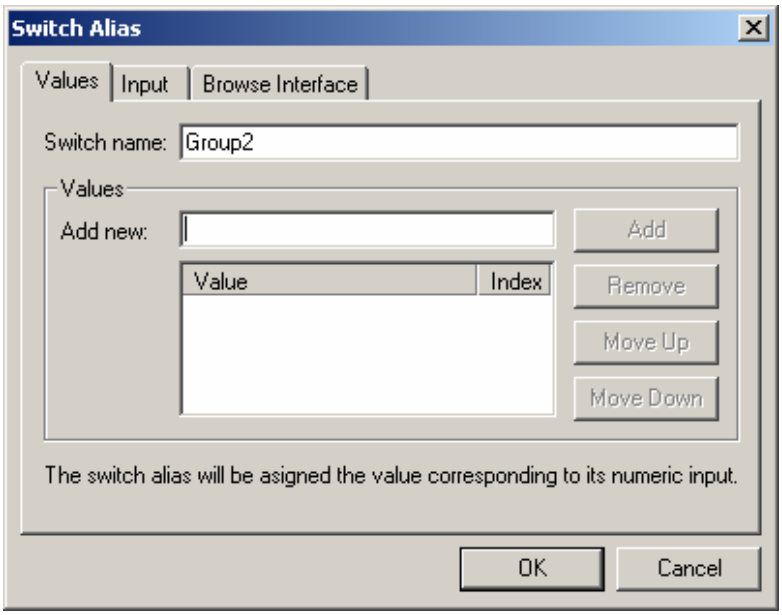
Creating a New Switch Alias

To create a new switch alias, select **Add** from the **Switch** menu. This opens the **Switch Alias** dialog box, which contains the following tabs.

- Values
- Input
- Browse Interface

Values Tab

You can use the **Values** tab of the **Switch Alias** dialog box, shown below, to define the name of the switch alias and its values.



Switch Alias Dialog Box: Values Tab

1. Enter the name of the switch alias is in the **Switch Name** field. The name must meet the conditions for register name (unique, consisting of letters, numbers, and underscores). The possible values of the alias are listed in the list control below.

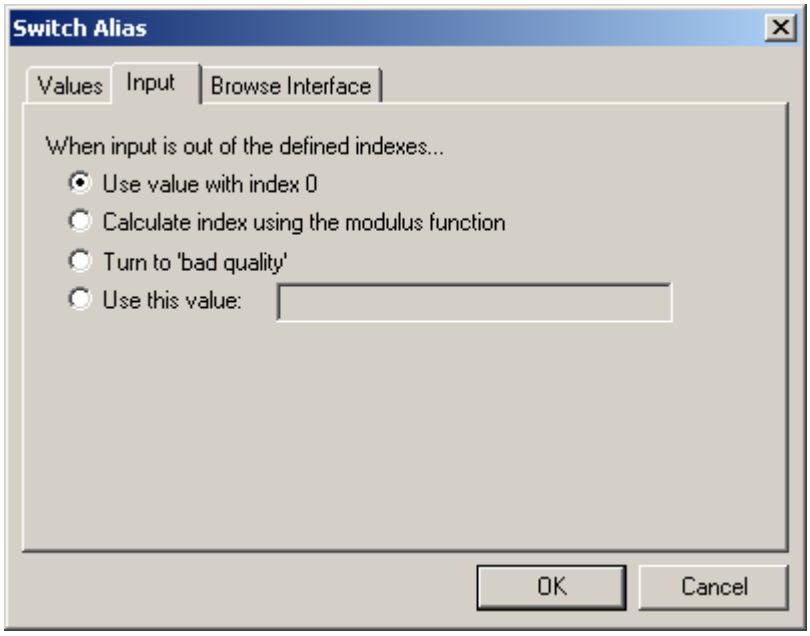
2. Use the **Add New** field and the **Add** button to add new values. Use the **Move Up, Move Down,** and **Remove** buttons to change the order of the values.

3. To edit a value, simply click it in the **Value** column. The corresponding numbers for particular values are listed in the **Index** column.

Note
The indexes cannot be changed.

Input Tab

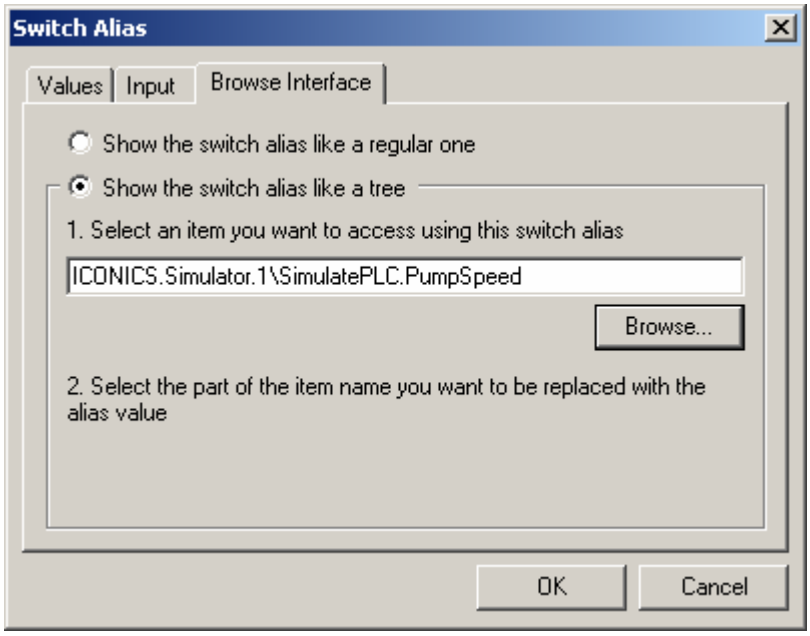
When the input is outside the range of the defined indexes, you can select an input type using the **Input** tab of the **Switch Alias** dialog box, shown below. For example, suppose you have defined four values for the switch alias; thus, their indexes are "0," "1," "2," and "3." The **Input** tab allows you to define what should happen when the switch alias's input is less than "0" or greater than "3." The value with index "0" can be used, or the MODULUS function can be used. It can indicate "bad quality." You can also specify a value.



Switch Alias Dialog Box: Input Tab

Browse Interface Tab

The **Browse Interface** tab of the **Switch Alias** dialog box, shown below, can be used to define how the switch alias should appear in the browse interface.



Switch Alias Dialog Box: Browse Interface Tab

Show the switch alias like a regular one. Selecting this option will cause the switch alias to be displayed as a leaf of the tree.

Show the switch alias like a tree. The reason for this second option is that the switch aliases are typically used to switch a subtree of items in an OPC server. Then the subtree of items will appear as a subtree of the switch alias. In other words, the hierarchy displayed in the browser will mirror the exact tree structure of the switch alias set up in the group sub-window of DataWorX.

Then simply follow the two steps indicated on the **Browse Interface** tab. If you choose to show the switch alias like a tree, first select an item you want to access using the switch alias. Then select a part of the item name you selected in Step 1. This selected part is the one you intend to replace with the alias.

Example of Using a Switch Alias

Assume that a Modbus OPC server with two devices is connected to one PC. Both devices are getting the same process data. The task is to establish redundancy on these devices.

Assume further that the connection from DataWorX to the OPC server works well. What can go wrong is the connection from the OPC server to the devices. Redundancy aliases cannot solve this problem as it works on the server level. This is a task for switch aliases and conditions.

1. Assume that a tag connected to the first device is: SMAR.ModbusOPCServer\device1.Tag1

The same data can be accessed via the second device as: SMAR.ModbusOPCServer\device2.Tag1

2. Now define a switch alias that will switch between the two devices:

Alias name: DEVICE

Set of predefined values: device1, device2

Tag1 mentioned above will be then accessed using the switch alias as:

SMAR.ModbusOPCServer\[[DEVICE]].Tag1

Depending on the input of the DEVICE alias, either device1 or device2 will be involved.

3. To set up the correct value on the DEVICE alias input a register with condition will be used. Define such a register:

- Add a new register and choose a name, such as RCOND.
- Go to the **Input** tab, select **Condition** input, and then click the **Edit** button.
- Select **First with Good Quality** as the criterion, and **Zero-Based Index** as the result.
- Switch to the **Inputs** tab of the Condition dialog box, and then add the two tags mentioned above:

SMAR.ModbusOPCServer\device1.Tag1

SMAR.ModbusOPCServer\device2.Tag1

The output of the RCOND register is a number that tells which is the first tag with the good quality.

4. Finally, connect the output of the RCOND register to the DEVICE alias.

Redundancy

The **Redundancy** feature of DataWorX allows the user to utilize other PCs as backup servers if a primary server goes offline. This means that users can designate alternative PC machines as backup servers if a designated primary server goes offline. Once a primary server does go off-line, DataWorX will default to the backup server or servers in the sequence in which the backup servers were designated.

If the appropriate option is selected in the dialog box provided for the purpose, DataWorX will default to the primary server once it returns on-line.

Main Qualifying Factors for Using Redundancy

The following are the main qualifying factors for using the Redundancy feature in DataWorX:

1. Users must designate one OPC server as the "Primary" server in each set.

2. Users may designate one or more OPC servers as the "Backup" servers in each set. (The number of servers is not restricted by DataWorX itself; rather it is limited only by system resources). If more than one backup server is specified, they should be ordered (2^{nd}, 3^{rd}, 4^{th}, etc.) The user will see a message outlining the details of the discrepancies and is allowed to either accept it as is, or permit reconfiguration.

3. The various OPC client applications request data from DataWorX, rather than from the OPC server directly. This way if a Primary OPC server failure occurs (due to any number of conditions), an automatic switchover to the Backup OPC server occurs.

4. DataWorX monitors the OPC Servers and aggregates the data to the requesting Clients.

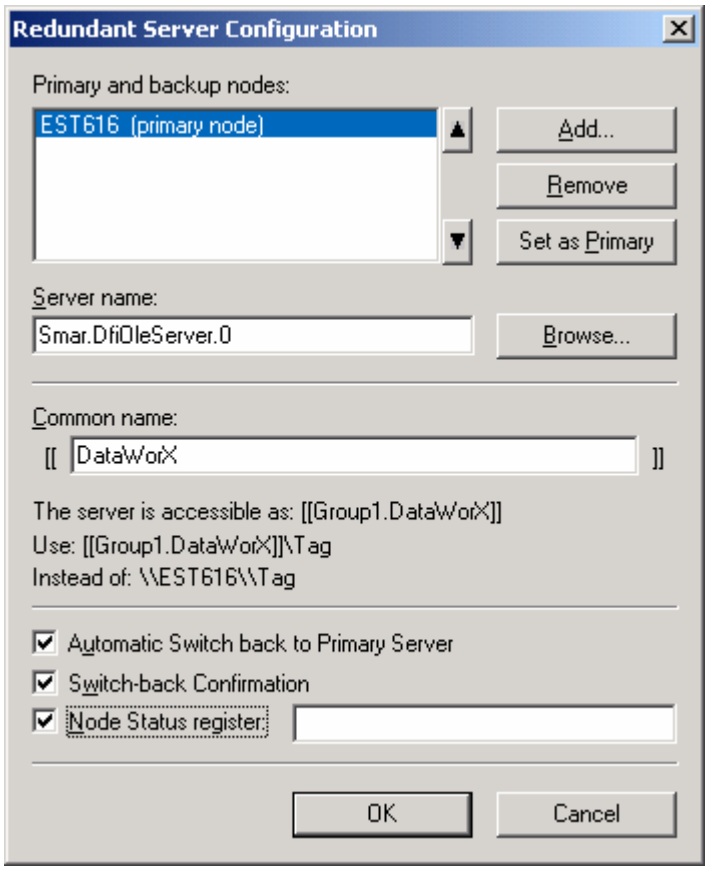
Redundancy Menu

This section describes how you can use the **Redundancy** menu to add, edit, and remove redundant nodes.

Adding Redundant Nodes

To add a redundant node:

1. Select **Add Redundant Node** from the **Redundancy** menu. This opens the **Redundant Server Configuration** dialog box, shown below.



Redundant Server Configuration Dialog Box

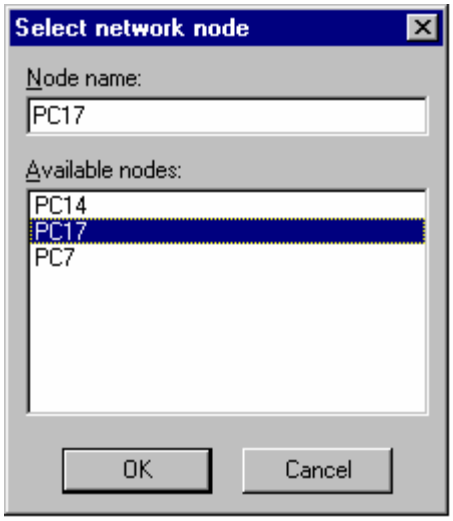
2. Configure the parameters as described in the following table.

3. Click **OK.**

Note
The "up" and "down" arrows in the **Primary and Backup Nodes** field are not for scrolling up and down but for changing the order of the nodes.

Redundant Server Configuration Parameters

Primary and Backup Nodes	This box displays the names of the designated primary and backup nodes.
Add	Allows you to add a network node to be designated as a primary or secondary node. Selecting the Add button in the Primary and Backup Nodes section displays the Select network node dialog shown in the following figure: You must enter the necessary data in the spaces provided as defined below:



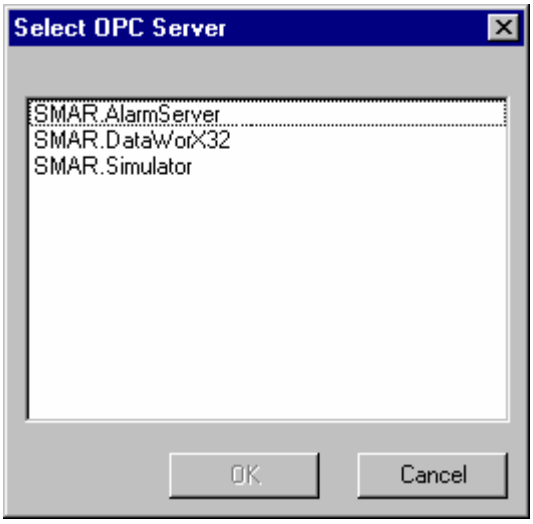
Select Network Node Dialog Box

Select Network Node Parameters

Node Name	Allows you to specify the node you wish to add. To select a node name, enter a node name and then click OK.
Available Nodes	Displays the list of nodes available on the network.

Redundant Server Configuration Parameters *(continued)*

Remove	Allows you to remove a network node from the list of primary or secondary nodes .
Set as Primary	Allows you to designate a PC on the network as a primary node. Once selected, the name of the chosen primary server will be displayed in the Primary and Backup Nodes box.
Server Name	Displays the name of the server from which the designated PC is receiving its input.
Browse	Opens the Select OPC Server dialog box, shown below, which allows you to browse the list of OPC servers in order to choose a server from which the PC that will be designated as a primary or secondary node will receive its input.



Select OPC Server Dialog

Redundant Server Configuration Parameters *(continued)*

Common Name	Allows you to specify a common tag name by which the server can be accessed. **Note:** The common name must be enclosed by double brackets [[ ]].
Automatic Switch-Over	Checking this box ensures that DataWorX will switch the system back to the designated primary server once it returns online.
Switch-Back Confirmation	When DataWorX detects that a primary server is running again (after having crashed), it switches back to the primary server from the backup server. If this option is checked, you will be asked before DataWorX switches back to the primary server. This is in case you may not want to the primary server for some reason (e.g. the server is in testing mode).
Node Status Register	Checking this box allows you to specify whether the primary node is online or offline using the integers "0" or "1." This register could be very helpful in triggering alarms and starting/stopping data logging.

Editing Redundant Nodes

To edit a redundant node, select Edit from the Redundancy menu. This opens the Redundant Server Configuration dialog box. The User Interface for the Redundant Server Configuration Edit dialog box is similar to the dialog for the Redundant Server Configuration Add dialog box.

Note
You can scroll up and down the list of nodes using the "up" and "down" arrows.

Removing Redundant Nodes

To remove a redundant node:

1. Select a redundant node.
2. Select **Remove Redundant Node** from the **Redundancy** menu.
3. This will remove the redundant node from the list of nodes designated as primary and backup nodes.

Tools

This section explains how you can use the Tools functions in your configuration of DataWorX.

The **Tools** menu contains the following commands:

- Options
- Security configuration
- Set working directory
- Event logger options

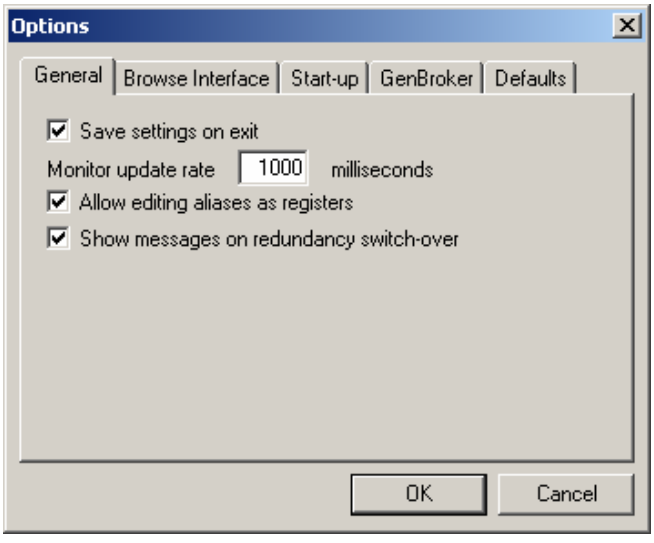
Options

Selecting the **Options** command from the **Tools** menu opens the **Options** dialog box, which allows you to define settings that determine how you can use and navigate within DataWorX. The Options dialog box contains the following tabs:

- General
- Browse Interface
- Start-up
- GenBroker
- Defaults

General Tab

The **General** tab of the **Options** dialog box, shown below, contains the following commands:

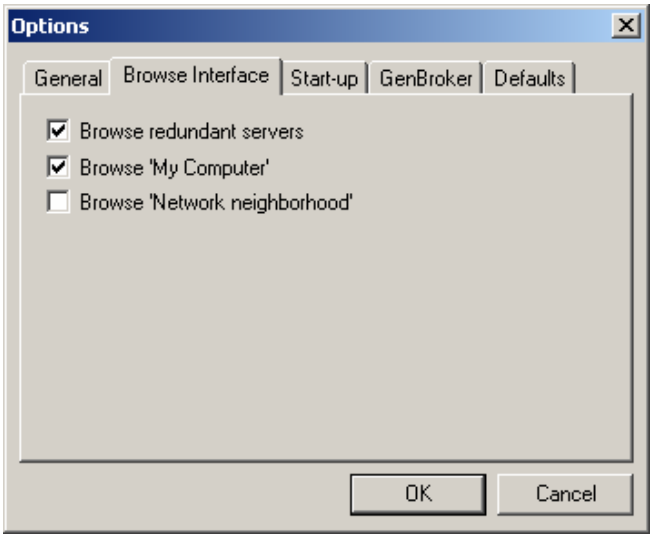


Options Dialog Box: General Tab

Save settings on exit	Checking this box saves the settings on exiting the application.
Monitoring update rate	Specifies the update rate (in milliseconds).
Allow editing aliases as registers	Allows you to edit aliases as registers in configuration mode.
Show messages on redundancy switch-over	Checking this box ensures that when one of the redundant servers goes offline or when the primary server comes back online again, a message is displayed indicating that the system has defaulted (switched over) to another PC.

Browse Interface Tab

The **Browse Interface** tab of the **Options** dialog box, shown below, contains the following commands:

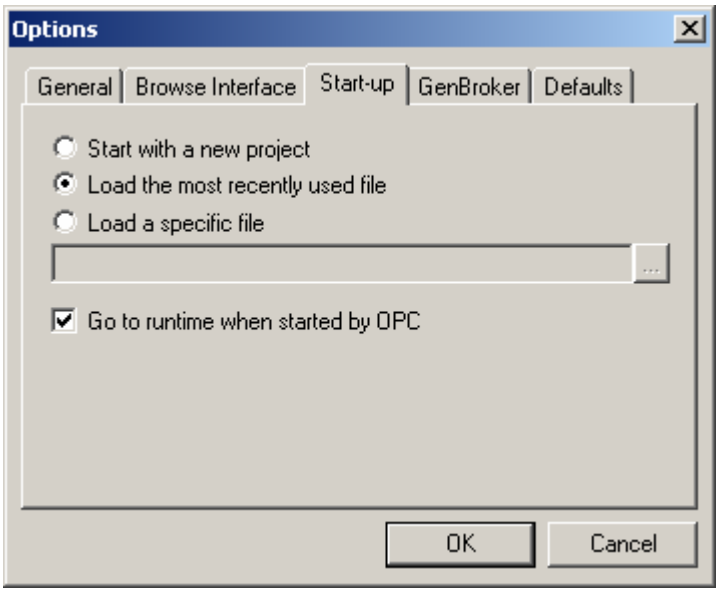


Options Dialog Box: Browse InterfaceTab

Browse redundant servers	When checked, the items of the servers defined as redundant (in the Redundancy menu) will be shown.
Browse 'My Computer'	When checked, the local OPC server and its items available through DataWorX will be shown.
Browse 'Network neighborhood'	When checked, the OPC servers located on remote nodes and their items will be shown.

Start-up Tab

The **Start-up** tab of the **Options** dialog box, shown below, contains the following commands:

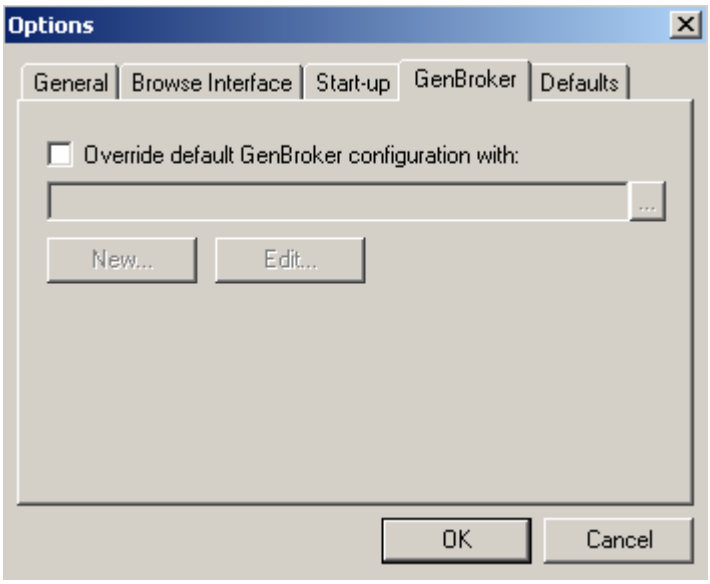


Options Dialog Box: Start-up Tab

Start with a new project	Checking this box specifies whether DataWorX will start with a new object.
Load the most recently used file	Checking this box specifies whether the most recent project file will be loaded upon starting DataWorX.
Load a specific file	Checking this box allows you to specify whether any file should be loaded on start-up. If so, you can specify which file should be loaded. You can browse for a file by clicking the button to the right.
Go to runtime when started by OPC	When checked, enters runtime when started by the OPC server.

GenBroker Tab

In the Smar ProcViewBroker Configurator, you may specify whether applications should use OPC communication or ProcViewBroker communication. If ProcViewBroker communication is selected, you may specify a configuration file. This applies to all ProcessView applications. You can override this setting for DataWorX because GenBroker communication provides better redundancy support. The **ProcViewBroker** tab of the **Options** dialog box, shown below, contains the following commands:



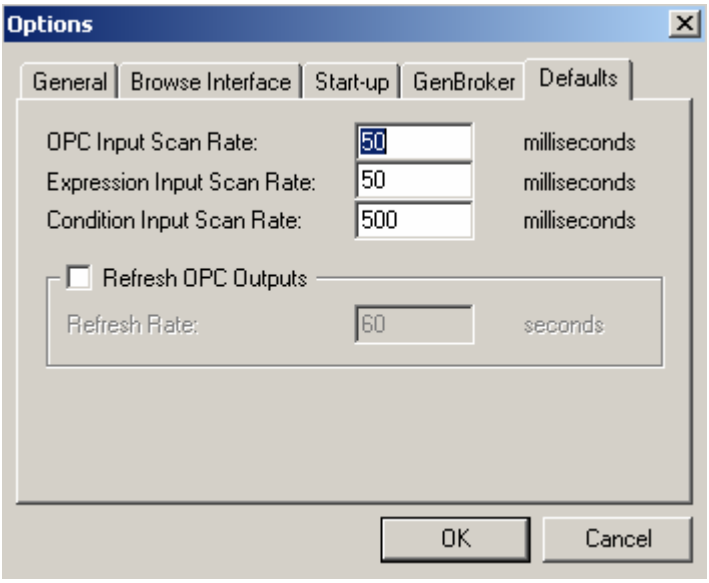
Options Dialog Box: GenBroker Tab

Override default ProcViewBroker configuration with:	Check this box if you wish to override the default ProcViewBroker configuration and specify an alternate configuration to be used in DataWorX. Clicking the button to the right allows you to browse for a different ProcViewBroker configuration.
New	Clicking this button opens a new configuration file in the Smar ProcViewBroker Configurator.
Edit	Clicking this button allows you to edit an existing configuration file in the Smar ProcViewBroker Configurator.

Defaults Tab

The **Defaults** tab of the **Options** dialog box, shown below, enables you to set the default settings for the OPC Input Scan Rate, Expression Input Scan Rate, and Condition Input Scan Rate (in milliseconds).

When bridging OPC data, values are written to the outputs only when the input value changes. This is the default behavior. To modify this behavior, you can force DataWorX to refresh outputs periodically. Then the value is written to the outputs even if the input value does not change. To switch the output refresh mechanism, check the **Refresh OPC Outputs** check box on the **Defaults** tab of the **Options** dialog box, and specify the output refresh rate (in seconds).



Options Dialog Box: Defaults Tab

Security Configuration

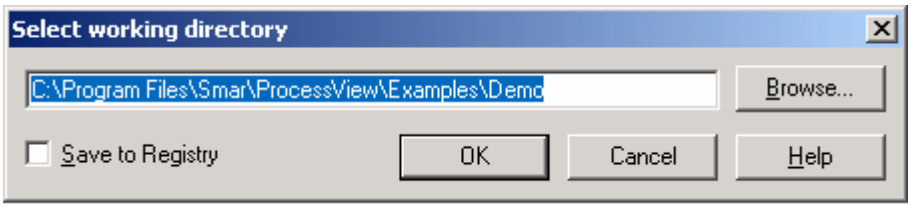
Selecting **Security Configuration** from the **Tools** menu opens the **Smar Security Server Administrator Login** dialog box, shown below. Enter your user name and password to login to the Smar Security Configurator. Click **OK** to launch the Security Configurator. You can now proceed with your security configuration.

Smar Security Server Administrator Login Dialog Box

Set Working Directory

The **Set Working Directory** command on the **Tools** menu allows you to set a working directory that can be also saved in the system registry.

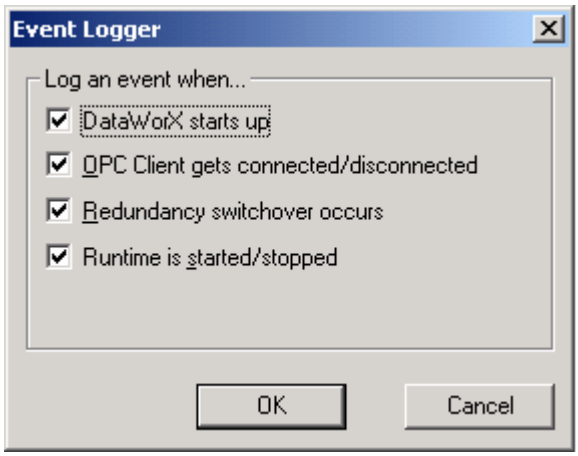
Note
This is a working directory common for all ProcessView applications.



Select Working Directory Dialog Box

Event Logger

Some events can be logged using the ProcessView Event Logger. Events are logged by the ProcessView logger system, which also sends these messages to the NT Event Logger. Selecting **Event Logger Options** from the **Tools** menu opens the **Event Logger** dialog box, shown below. Selecting all or any of these options will log the chosen events using the ProcessView Event Logger.



Event Logger Dialog Box

Action

This section describes the runtime environment of DataWorX. Runtime lets you view real-time data as they are transmitted between the servers on the network. This is the state in which you can view the live data being received according to the parameters you set in configuration mode.

The aggregation feature, which was discussed in the Getting Started section, is especially useful during runtime mode. A register with its input connected to an OPC point can be used to "prerequest" items from the OPC servers. Because registers remain in existence during the entire runtime operation, the items connected to their inputs are also requested for the entire duration of runtime. Because of the feature of OPC aggregation, clients can then request these items without a delay.

During runtime, you can perform the following tasks;

- Open a DataWorX file.
- Edit the value of a register.
- Change the value of an alias.
- Edit an alias and change its default value.

Command Line Option for Runtime

The command line option **runtime** starts runtime mode in DataWorX.

Use:

dwx32 -runtime or dwx32 -start)

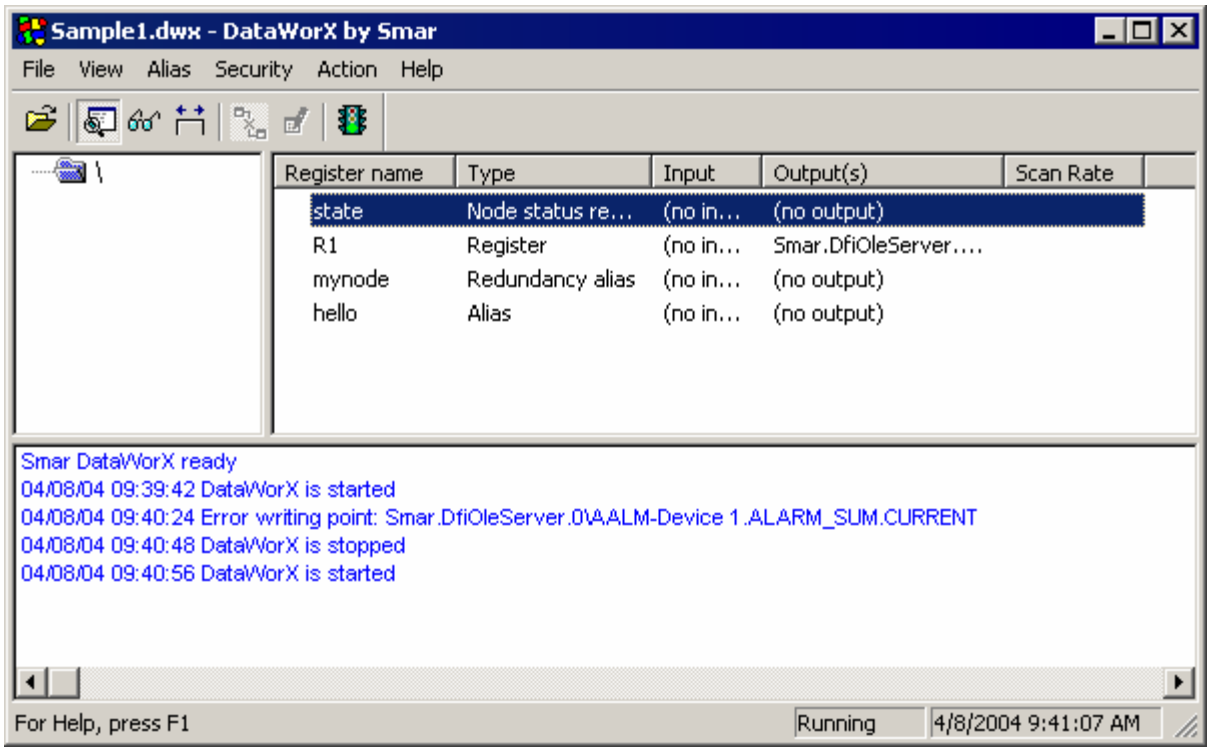
Switching Between Configuration and Runtime Mode

You can switch to runtime mode by selecting **Start DataWorX** from the **Action** menu in configuration mode.

You can stop runtime mode by selecting **Stop DataWorX** under the **Action** menu.

Clicking the traffic light button on the toolbar, shown at left, also commences or stops runtime mode.

This displays the following screen with live data for the items displayed.:



DataWorX Runtime Screen

DataWorX runtime mode displays live data in the top pane of the splitter window, as shown in the above figure.

The output (lower) pane of the DataWorX runtime screen displays the current status of DataWorX. It logs the start and stop of runtime as many times as runtime is started and stopped. It also displays messages giving the current status of the tags.

On the Runtime Toolbar the traffic light icon switches from red to green once you switch from the configuration mode to runtime mode.

Runtime Menu Bar

The runtime menu bar provides the following menus that can be used to configure and view a chosen file in DataWorX during runtime mode:

- File
- View
- Aliases
- Security
- Action
- Help

File

The **File** menu in runtime mode contains the following commands:

- Open
- Exit

To open a file in DataWorX, select **Open** from the **File** menu.

To exit DataWorX and return to Windows, select **Exit** from the **File** menu.

View

The **View** menu in runtime mode contains the following commands:

- Toolbar
- Status Bar
- Output
- Registers
- Aliases
- Monitor Values
- Adjust Column Widths
- Statistics

Toolbar

Selecting **Toolbar** from the **View** menu in runtime mode displays the runtime toolbar, shown below.

Runtime Toolbar

Status Bar
Selecting **Status Bar** from the **View** menu in runtime mode displays the status bar at the bottom of the screen showing the following items:

- Context-sensitive help
- Whether DataWorX is running or stopped
- Time
- Date

Status Bar

Output

Clicking the **Toggle Output Window** button on the runtime toolbar, shown at left, or selecting **Output** from the **View** menu in runtime mode, allows you to toggle between views of the DataWorX screen with or without the output window in the bottom pane.

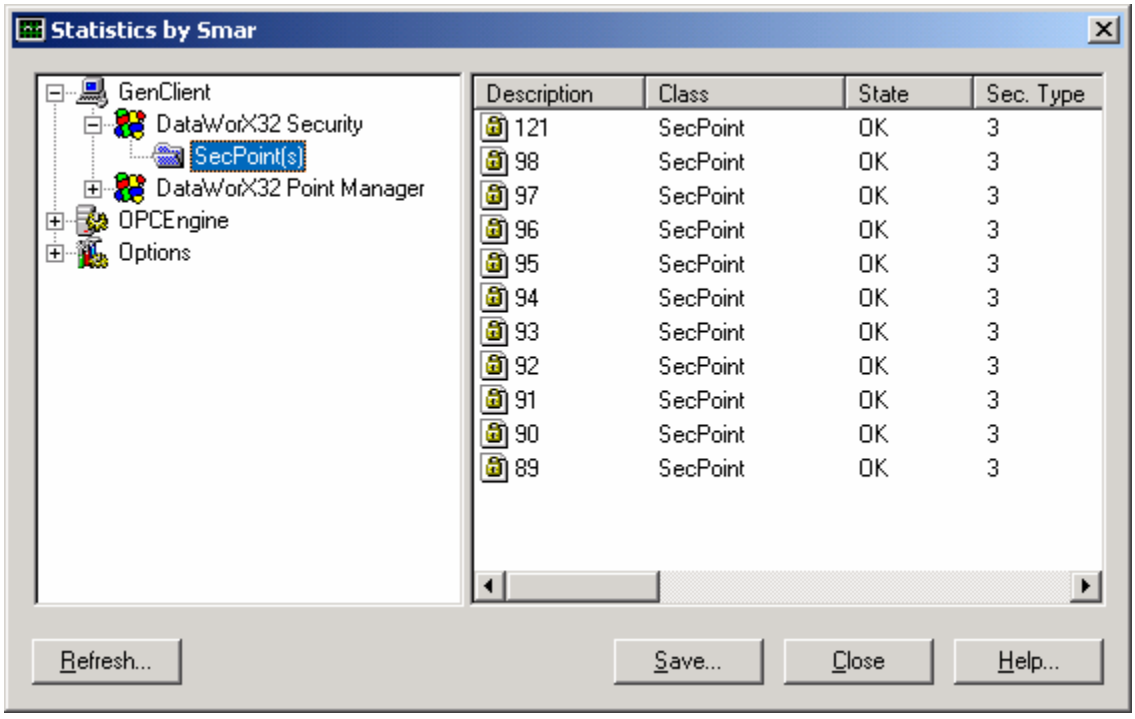
Registers
Selecting **Registers** from the **View** menu in runtime mode displays all the registers in runtime mode. This command is available in the runtime mode as well as in configuration mode.

Aliases
Selecting **Aliases** from the **View** menu in runtime mode displays all the aliases in runtime mode. This command is available in the runtime mode as well as in configuration mode.

Monitor Values

Clicking the **Display Values** button, or selecting **Monitor Values** from the **View** menu in runtime mode, displays the current values of the tags displayed in the registers and aliases views.

Statistics
In runtime mode, you can view the statistics of the live data being received. To view the current status of the servers and the statistics of the data being received, select **Statistics** from the **View** menu in runtime mode. This displays the **Runtime Statistics** dialog box, shown below. Click the **Help** button for more information.



Runtime Statistics

Aliases

In runtime mode, the **Alias** menu provides you with the option to change an alias value. Select **Change Value** from the **Alias** menu in runtime mode to open the **Change Alias Value** dialog box.

Security

In runtime mode, the **Security** menu provides the option for security configuration in runtime. Selecting **Login** from the **Security** menu in runtime mode launches the Smar Security Login utility. Refer to the Security help documentation for more information.

Action

Select **Stop DataWorX** from the **Action** menu in runtime mode to return to configuration mode.

Right-Click Menus Available in Runtime Mode

Right-clicking on a register or alias in runtime mode opens the popup menu shown below.

Selecting **Disconnect Input** disconnects the register or alias from the OPC input (if any). If the register or alias is not connected, this menu allows you to enter the register/alias input manually.

Selecting **Change Value** allows you to change the current value of the selected register or alias.

Right-clicking on a redundancy alias in runtime mode opens the popup menu shown below.

Selecting **Switch to Primary Node** switches the redundancy alias to the primary node.

Selecting **Switch to Next Backup Node** switches the redundancy alias to the next backup node

Exiting Runtime

You are provided the following two options for exiting runtime mode.

1. You can exit the DataWorX runtime environment and return to the DataWorX configuration screen.

2. You can close DataWorX and return to Windows.

To exit runtime mode and return to the configuration screen, click the traffic light button on the runtime toolbar, or select **Stop DataWorX** from the **Action** menu .

To exit runtime by closing DataWorX and returning to Windows, select **Exit** from the **File** menu. This simultaneously closes down runtime and exits DataWorX. Make sure you are certain that you want to exit runtime before you perform this action, because selecting the exit option will close DataWorX without asking for any confirmation.

Help

In both configuration mode and runtime mode the **Help** menu for DataWorX contains the following two commands:

- Help Topics
- About

Help Topics

Selecting the **Help Topics** from the **Help** menu opens the Help file for DataWorX.

About

Selecting **About Application** from the **Help** menu, or clicking the **About** button on the toolbar in configuration mode, shown at left, displays the Smar About Box, which displays information about the current release of DataWorX.

Distributed Component Object Model (DCOM)

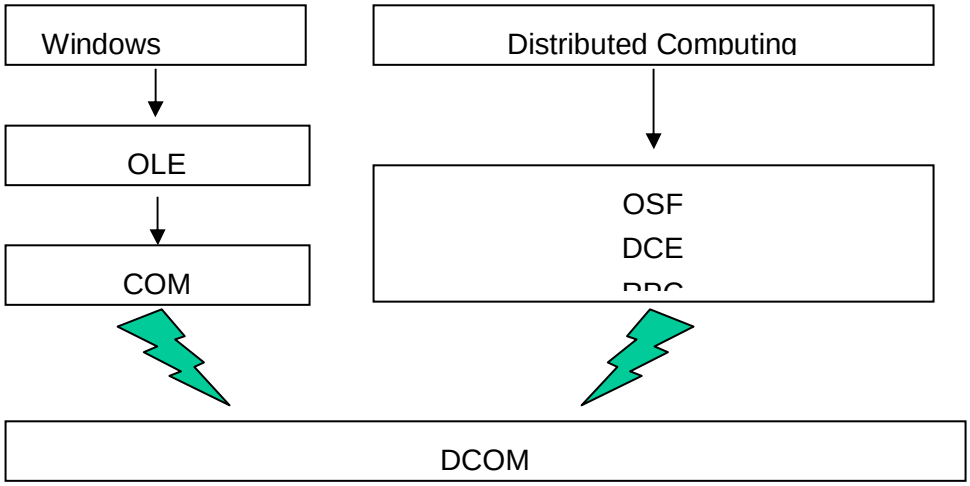
This chapter addresses DCOM configuration issues on Windows 95/98 and Windows NT 4.0. This will help the user while configuring the system in a true distributed client/server environment over DCOM. For more information about DCOM, refer to the Windows help documentation.

Introduction

Distributed Component Object Model (DCOM) is a fairly new concept based on the foundation provided by Object Linking and Embedding (OLE). DCOM is also called "COM with a long wire." This is because OLE or Component Object Model (COM) only works on a local computer, while DCOM extends the same concept over the network with extended security features and application access privileges. DCOM is the technology that extends COM to allow components objects to "live" on remote machines.

DCOM facilitates the process of connectivity and thus allows users to select various components from different vendors without loosing performance or functionality. This concept is heavily supported by Open Software Foundation (OSF).

DCOM was not invented overnight; it is a merging of two technologies, as illustrated in the figure below.



The Evolution of DCOM

Compared with MS DOS, which could only run one application at time, the evolution of Multitasking Windows allows you to use multiple applications concurrently. It is possible to share data within applications via DDE. The clipboard offered cut, copy, and paste operations, while OLE took it one step further by allowing the editing of such objects. Over the years, OLE has faded into the background while COM has taken center stage.

With the release of MS WIN NT 4.0 in 1996, COM gained the functionality necessary to invoke components that were running on remote computers connected via a network.

DCOM was also the result of attempts made by various industry groups to get enough companies to define standards and then agree to abide by those standards. Open Software Foundation (OSF) addressed the issue of distributed computing, and out of this effort grew the specifications for the Distributed Computing Environment (DCE) . The outcome from OSF and DCE was a specification for communicating between computers, known as Remote Procedure Call (RPC), that allows the applications on different computers to communicate. DCOM uses RPCs for its inter-computer communication. This indicates that DCOM evolved from RPC.

Types of Components

Components come in one of three kinds: in-process, local, or remote, depending on the

In-Process

In-process servers are loaded into the client's process space because they are implemented as DLLs. DLLs are code libraries that are loaded at runtime by the operating system on behalf of programs that want to call functions in the DLLs. DLLs are always loaded into the address space of the calling process. This is important because in Windows 95 and Windows NT, each program (process) is loaded into its own private 32-bit address space for security and stability. Since it is not normally possible to access memory locations beyond this private address space, DLLs need to be loaded in-process.

The main advantage of in-process servers is their speed. Since the objects are loaded in-process, no context switching is necessary to access their services, as is the case with DLLs. The only potential disadvantage to in-process servers is that because an in-process server is a DLL and not a complete application executable (EXE), it can be used only in the context of calling a program and cannot be run as a stand-alone application. ActiveX controls are implemented as in-process servers.

Local

A **local** server runs in a separate process on the same machine as the client. This type of server is an EXE of its own, thus qualifying as a separate process. Local servers are comparatively slower to access than in-process servers because the operating system must switch between processes and copy any data that need to be transferred between the client and the server applications. The one advantage of local servers is that since they are EXE files, they can be run as stand-alone applications by the user without an external client.

Remote

Another type of component process is a **remote** server that runs on a separate machine connected via a network. Remote servers, therefore, always run in another process. This functionality can be implemented using DCOM. The advantage of DCOM is that it does not require any special programming to enable this functionality.

Machine Setup Tips

The following section outlines some facts and requirements for the DCOM networking.

- DCOM can be installed on a Windows 95 machine along with Internet Explorer 3.0x, since Windows 95 does not come with DCOM as part of the operating system. The installation files can be downloaded from www.microsoft.com or from SMAR Distributors CD98 (/Tools/DCOM). There are two self-extracting files. Double click for installation.

- DCOM is part of Internet Explorer 4.0 (old build). Please read the installation notes on installation of Internet Explorer 4.0. If you need to uninstall DCOM or Internet Explorer 4.0 from your computer, choose the reverse order as of your installation.

- Windows NT and Windows 98 come with DCOM as part of the operating system.

- Set User Level Security on the remote client as well as on the remote server. This can only be done where you have a domain controller on the Network. Windows NT Server and Windows NT Workstation can act as domains on the network. This domain is required for user security clearance.

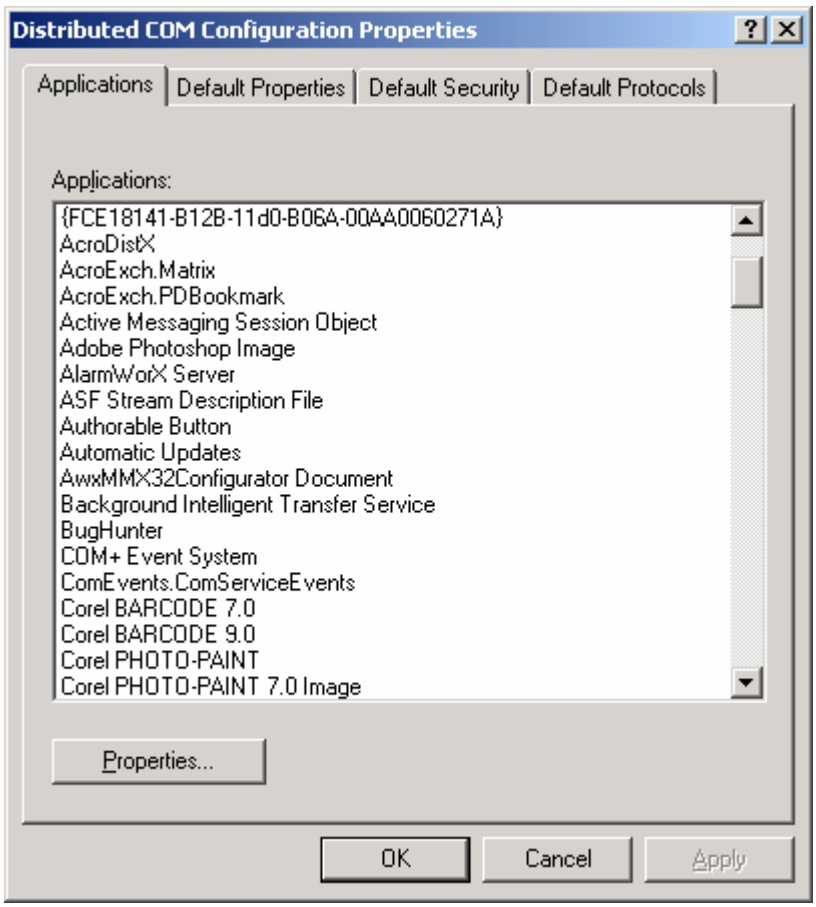
DCOM Configuration

DCOMCNFG.EXE (DCOM Config) is a utility you can use to secure DCOM Objects you have created. This section describes the DCOM Configuration interfaces, options, and settings.
Because security is much more limited on Windows 95, the interface and options may differ on Windows 95/98 systems as compared with Windows NT 4.0.

DCOM for Windows 95 and Windows 98

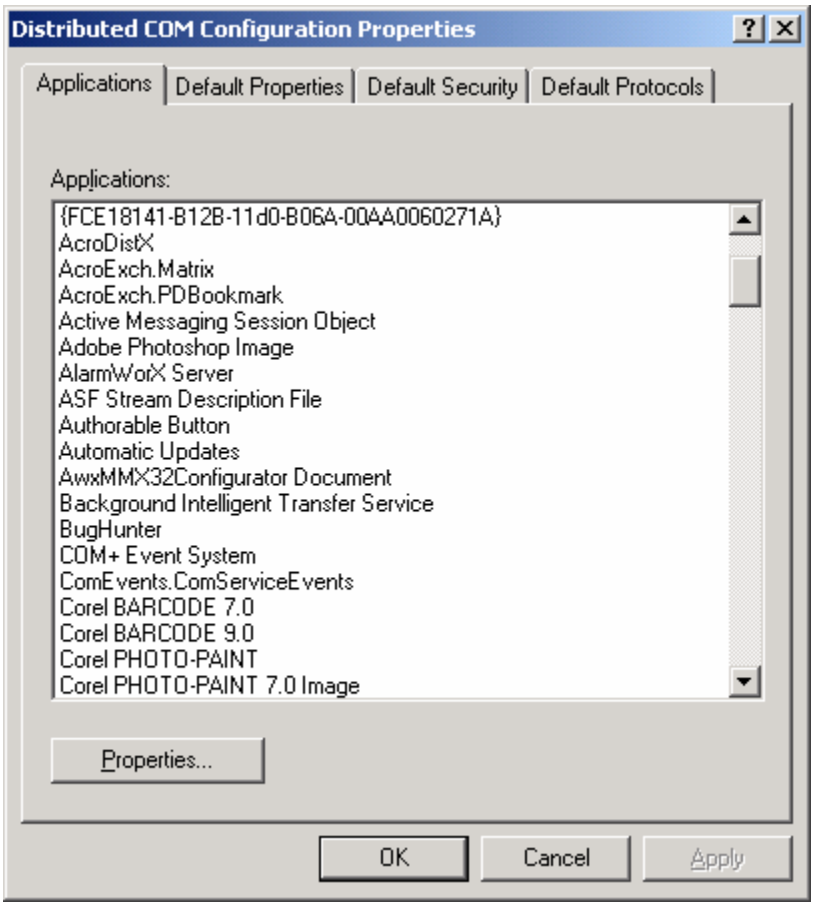
When you start DCOMCNFG.EXE from **Start - Run,** you will see the main interface, which contains the following tabs.

- Applications
- Default Properties
- Default Security



Distributed COM Configuration Properties Dialog Box

Applications Tab



Distributed COM Configuration Properties: Applications Tab

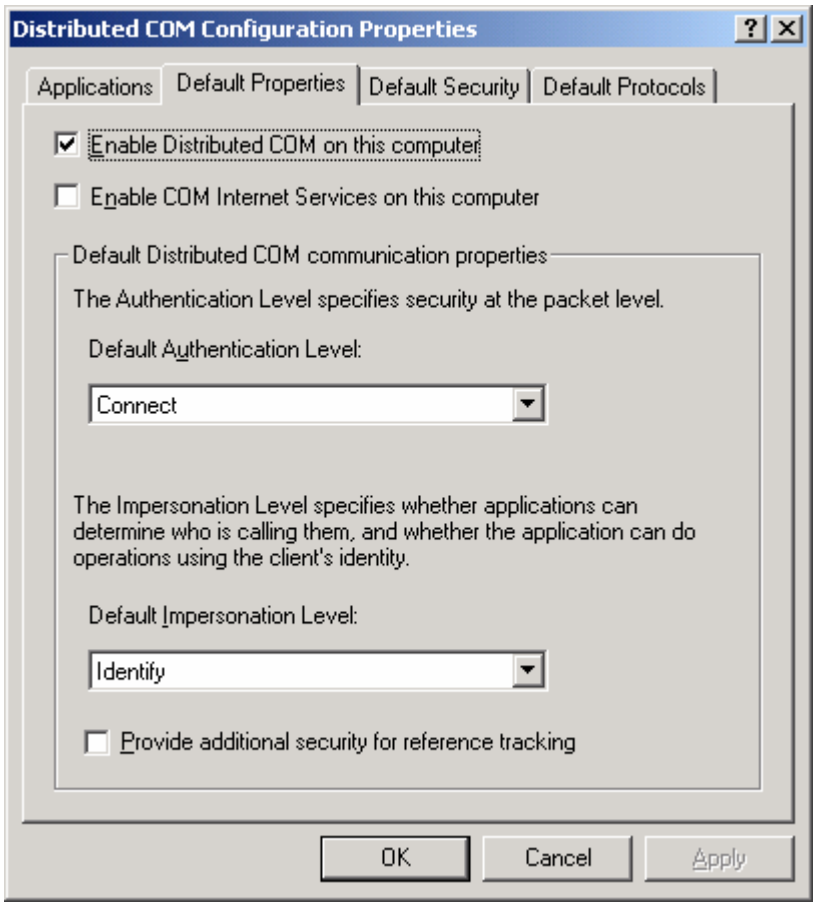
The **Applications** tab shows each of the items registered under the following registry key:

HKEY_CLASSES_ROOT\AppId\

Beneath this key are all the objects that can be launched on a remote machine. The **DCOM Configuration Properties** interface displays just the ProgIDs (friendly names) of each object, such as "Smar ProcessView OPC Server", or "Smar Modbus OPC Server." Some objects may register without registering a ProgID; in these cases, the GUID of the object will be displayed, such as "{4E6B942A-01B0-11D1-A9CB-00AA00B7B36F}."

For each item listed in the **Applications** tab, properties for each application can be viewed by selecting an item and clicking the **Properties** button or by double-clicking an application name.

Default Properties



Distributed COM Configuration Properties: Default Properties Tab

Each of the values displayed under the **Default Properties** tab may be found under the following key in the registry:

HKEY_LOCAL_MACHINE\Software\Microsoft\OLE

Enable Distributed COM on This Computer

The **Enable Distributed COM on This Computer** check box is a global setting for the entire machine. When this option is checked, the machine allows the creation of DCOM objects. If it is not checked, objects cannot be created via DCOM.

Note
You must reboot the system in order for a change in this setting to take effect.

Default Distributed COM Communication Properties

The Default Distributed COM Communication Properties section has two levels:

1. Default Authentication Level
2. Default Impersonation Level

These two options can only be modified if DCOM is enabled on this system.

1. Default Authentication Level (Packet Level)
Authentication Levels are as follows:

Name	Description
None	No authentication.
Connect	Authentication occurs when a connection is made to the server. Connectionless protocols do not use this.

Note that "Connect" is used for connectionless protocols only. Windows 95 uses TCP, which is connection-based.

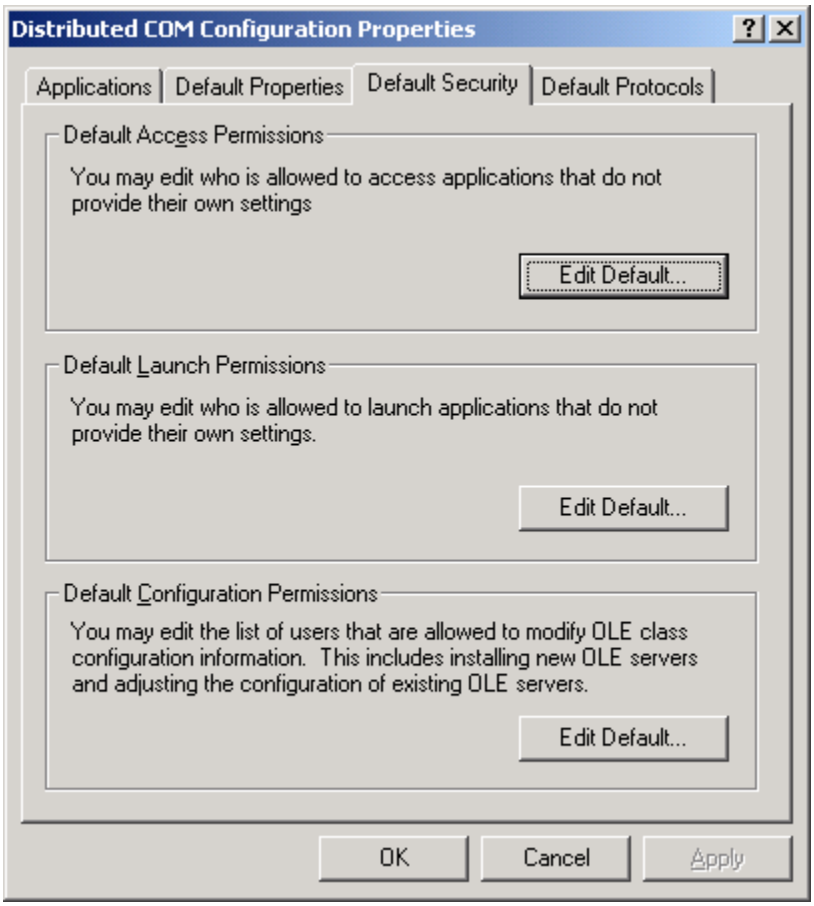
2. Default Impersonation Level
If no security is set at the object level, the server uses the security setting specified here as the default. The possible values are:

Name	Description
Identify	The server can impersonate the client to check permissions in the ACL (Access Control List) but cannot access system objects.
Impersonate	The server can impersonate the client and access system objects on the client's behalf.

Provide Additional Security for Reference Tracking

The last item on the **Default Properties** tab is a check box called **Provide additional security for reference tracking.** This tells the server to track connected client applications by keeping an additional reference count. Checking this box uses more memory and may cause COM to slow down, but it ensures that a client application cannot kill a server process by artificially forcing a reference count to zero.

Default Security



Distributed COM Configuration Properties: Default Security Tab

The **Default Security** tab is used set up default security permissions for all applications. The values stored here can be found in the Windows registry at the following location:

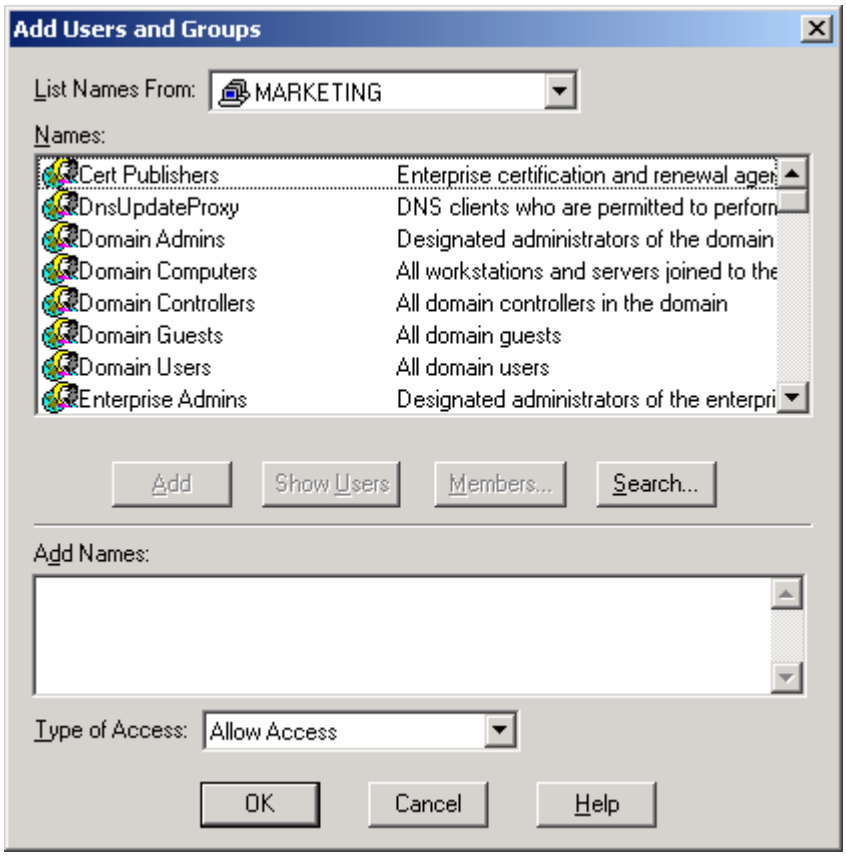
HKEY_LOCAL_MACHINE\Software\Microsoft\OLE

Default Access Permissions

This value determines the users and groups that can access an object when no other access permissions are provided.

Edit Default

Click **Edit Default** to add or remove the users from the list. You can grant or deny users access permissions on the machine. This list of users is obtained from the domain where the user is logged in.



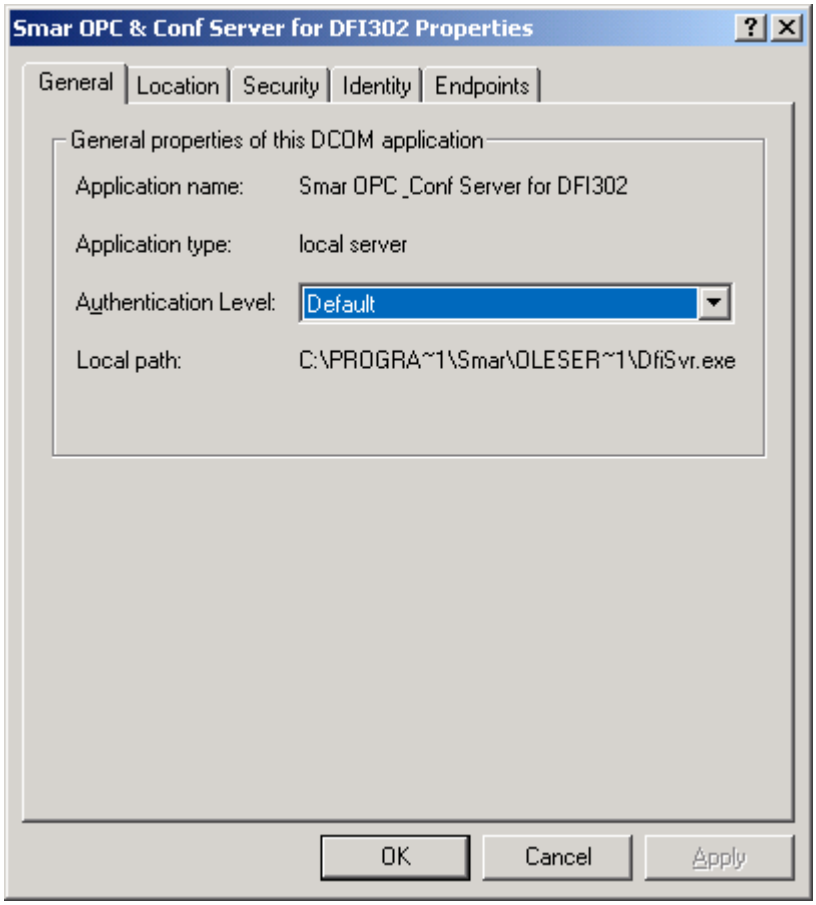
Add Access Permissions Dialog

Application Properties

On the **Applications** tab of the **Distributed DCOM Configuration Properties** dialog box, if you select a particular application and then click the **Properties** button, it will open interface for setting properties of that application. There are three tabs: General, Location, and Security.

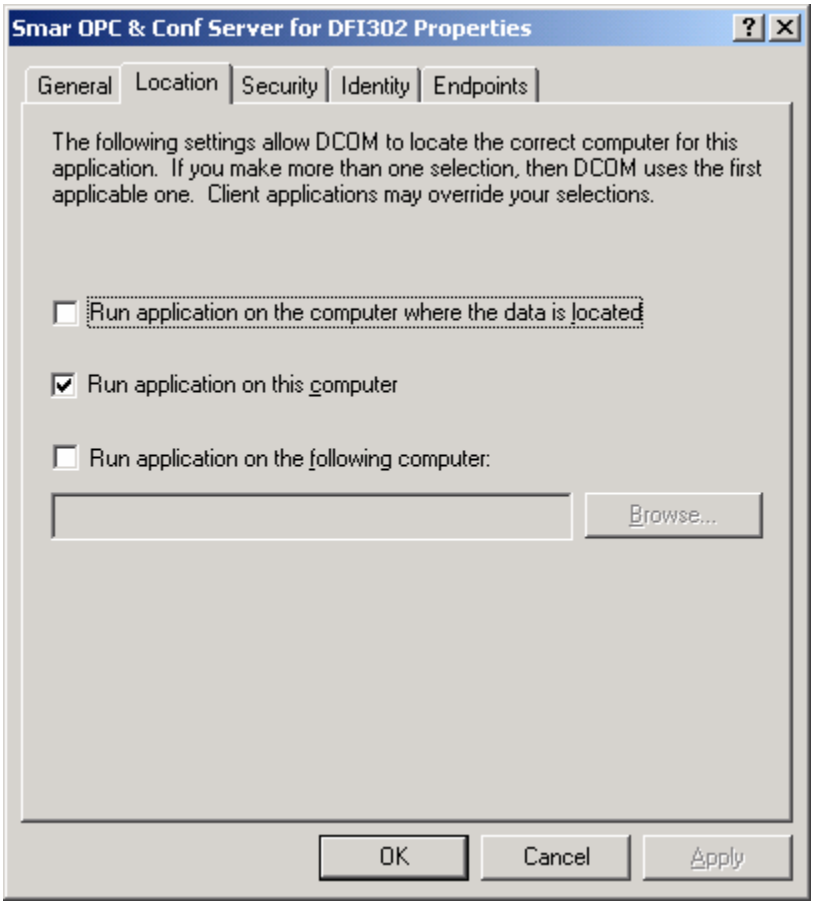
General

The **General** tab of the DCOM Application Properties only displays general information about the application.



DCOM Application Properties: General Tab

Location



DCOM Application Properties: Location Tab

The **Location** tab of the DCOM Application Properties is used to determine where DCOM will execute the application. Here you can specify where to run your application. There are three possible choices:

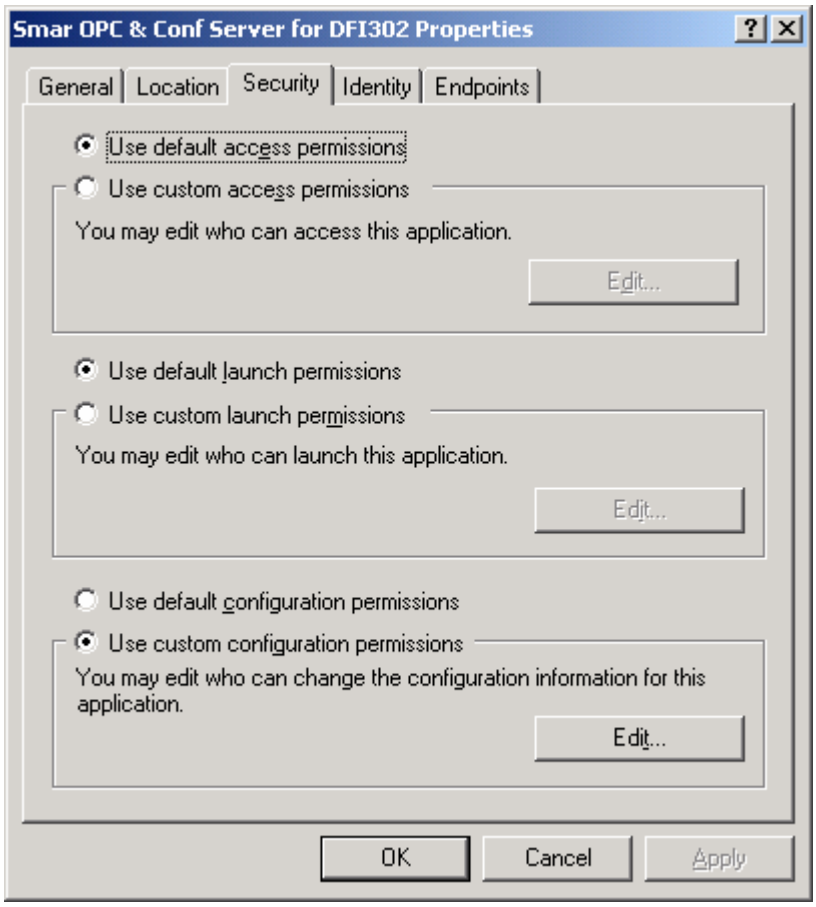
1. **Run application on the computer where the data are located.** If selected, DCOM will execute the application where the data are located. This is useful only if the application provides a data file for the server application.

2. **Run application on this computer.** This indicates that the DCOM application should run on the local machine.

3. **Run application on the following computer.** This allows you to specify a computer on which to execute. It requires the node name of the computer where the application is located. Some applications do not support network browsing of tags; in these cases, you can register the server locally and have the option of running the application on another machine. When this option is selected, the server will start on a remote location at the client's call.

If more than one of the above options is selected, DCOM will use the first applicable option. Client applications may also override this setting.

Security



DCOM Application Properties: Security Tab

The **Security** tab of the DCOM Application Properties provides two options:

Use the Default Security Option
In this case, you do not need to add any users. Default security access permissions are applied to this application.

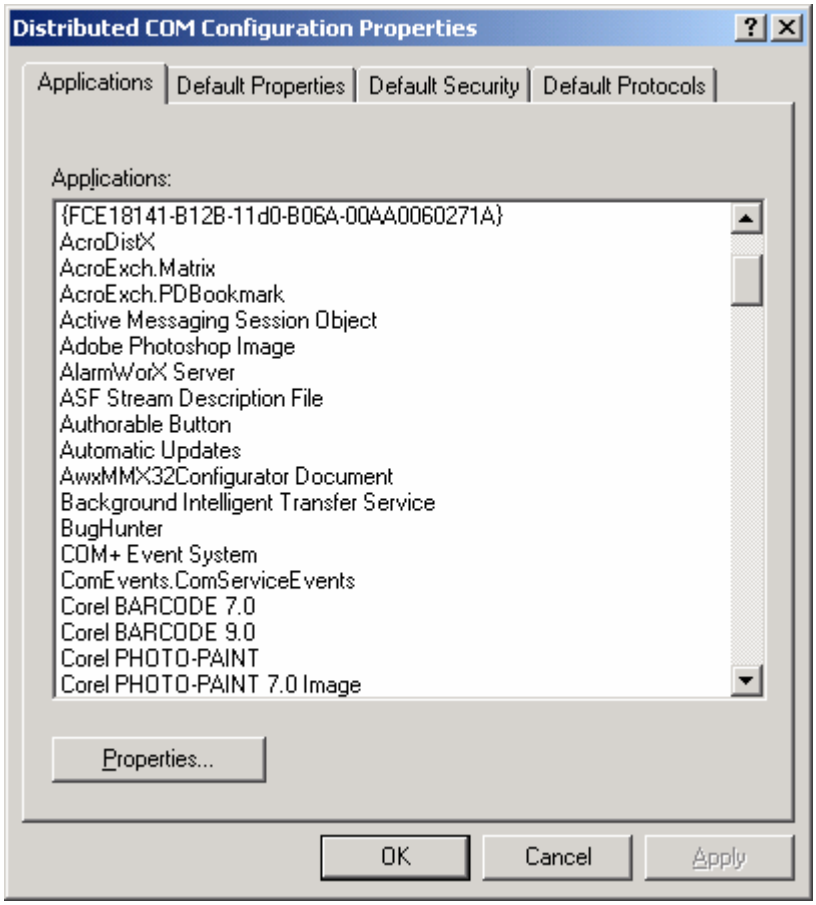
Use Custom Access Permissions
You can grant access to users ***only*** for this application instead for all applications from default security.

DCOM for Windows NT 4.0

When you start DCOMCNFG.EXE from **Start - Run,** you will see the main interface, which contains the following tabs:

1. Applications
2. Default Properties
3. Default Security

Applications Tab



Distributed COM Configuration Properties (Windows NT 4.0): Applications Tab

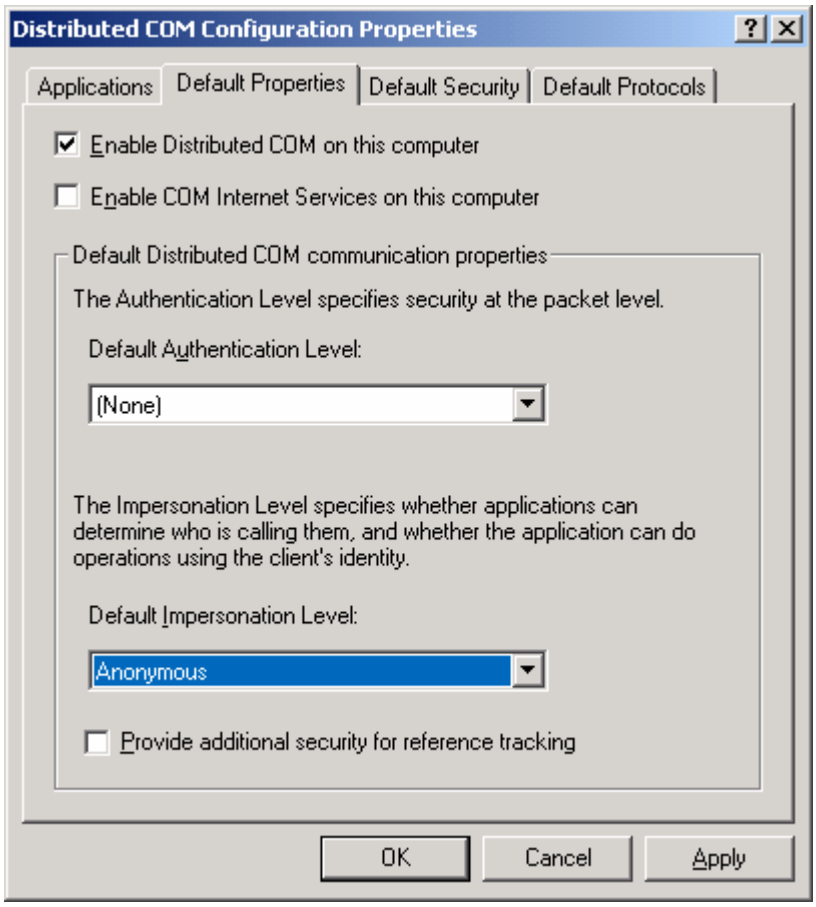
The **Applications** tab shows each of the items registered under the following registry key:

HKEY_CLASSES_ROOT\AppId\

Beneath this key are all the objects that can be launched on a remote machine. The DCOM Configuration Properties interface displays just the ProgIDs (friendly names) of each object, such as "SMAR Genesis OPC Server" or "SMAR Modbus OPC Server". Some objects may register without registering a ProgID; in these cases, the GUID of the object will be displayed, such as "{4E6B942A-01B0-11D1-A9CB-00AA00B7B36F}."

For each item listed in the **Applications** tab, properties for each application can be viewed by selecting an item and choosing the Properties" button or by double-clicking an application name.

Default Properties



Distributed COM Configuration Properties (NT 4.0): Default Properties Tab

Each of the values displayed under the **Default Properties** tab may be found under the following key in the registry:

HKEY_LOCAL_MACHINE\Software\Microsoft\OLE

Enable Distributed COM on This Computer

The **Enable Distributed COM on This Computer** check box is a global setting for the entire machine. When this option is checked, the machine allows the creation of DCOM objects. If it is not checked, objects cannot be created via DCOM.

Note
You must reboot the system in order for a change in this setting to take effect.

Default Distributed COM Communication Properties

The Default Distributed COM Communication Properties section has two levels:

1. Default Authentication Level
2. Default Impersonation Level

These two options can only be modified if DCOM is enabled on this system.

1. Default Authentication Level (Packet Level)
Authentication Levels are as follows:

Name	Description
None	No authentication.
Connect	Authentication occurs when a connection is made to the server. Connectionless protocols do not use this.
Call	The authentication occurs when a RPC call is accepted by the server. Connectionless protocols do not use this.
Packet	This authenticates the data on a per-packet basis. All data are authenticated.
Packet Integrity	This authenticates that the data have come from the client, and checks that the data have not been modified.
Packet Privacy	In addition to the checks made by the other authentication techniques, this encrypts the packet.
Default	This may vary depending upon operating system.

Note that "Connect" and "Call" are used for connectionless protocols only. Windows NT uses a connectionless protocol, UDP, by default. However, Windows 95 uses TCP, which is connection-based. Windows 95 machines can only accept calls on the "None" or "Connect" levels.

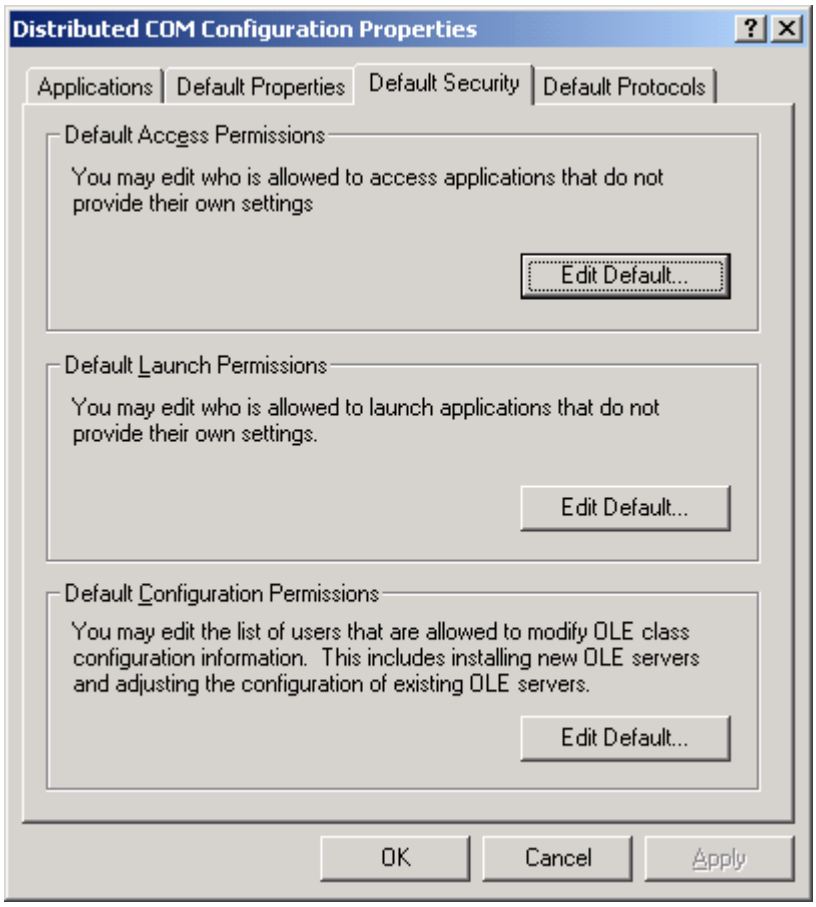
Default Impersonation Level
If no security is set at the object level, the server uses the security setting specified here as the default. The possible values are:

Name	Description
Anonymous	The client is anonymous. This setting is not currently supported by DCOM.
Identify	The server can impersonate the client to check permissions in the ACL (Access Control List) but cannot access system objects.
Impersonate	The server can impersonate the client and access system objects on the client's behalf.
Delegate	In addition to the Impersonate level, this level can impersonate the client on calls to other servers. This is not supported in the current release of DCOM.

Provide Additional Security for Reference Tracking

The last item on the **Default Properties** tab is a check box called **Provide additional security for reference tracking.** This tells the server to track connected client applications by keeping an additional reference count. Checking this box uses more memory and may cause COM to slow down, but it ensures that a client application cannot kill a server process by artificially forcing a reference count to zero.

Default Security



Distributed COM Configuration Properties: Default Security Tab

There are three options under the **Default Security** tab. Each of the values stored here can be found in the Windows registry at the following location:

HKEY_LOCAL_MACHINE\Software\Microsoft\OLE

Default Access Permissions

This value determines the users and groups that can access an object when no other access permissions are provided. For information on how to give individual access permissions to specific DCOM objects, see the "Application Properties" section later in this document. By default, access is provided to the "System" and "Interactive" groups.

Default Launch Permissions

This value determines the users and groups that can launch an object when no other access permissions are provided. For more information on how to give individual launch permissions to specific DCOM objects, see the "Application Properties" section later in this document.

Default Configuration Permissions

This value determines the users and groups that may read or modify configuration information for DCOM applications. This also determines which users and groups will have permission to install new DCOM servers.

System Groups

When you configure users and groups, you will find several group accounts. The following list is a summary of which user belongs to each group:

Group	Description
Interactive	Includes all users who log on to a Windows NT system locally (at the console). It does not include users who connect to NT resources across a network or are started as a server.
Network	Includes all users who connect to Windows NT resources across a network. It does not include those who connect through an interactive logon.
Creator/Owner	The Creator/Owner group is created for each sharable resource in the Windows NT system. Its membership is the set of users who either create resources (such as a file) and those who take ownership of them.
Everyone	All users accessing the system, whether locally, remotely, or across the network.
System	The local operating system.

The list above includes the group accounts that are intrinsic to Windows NT systems. Your particular network may include more groups from which you may choose. In order to determine the membership of each custom group account, you must contact your network administrator.

Application Properties

You can specify custom settings for individual DCOM applications by clicking the **Properties** button on the **Applications** tab in the DCOM Configuration Properties user interface.

The following sections describe each tab **(General, Location, Security,** and **Identity)** and setting found within the application properties.

General

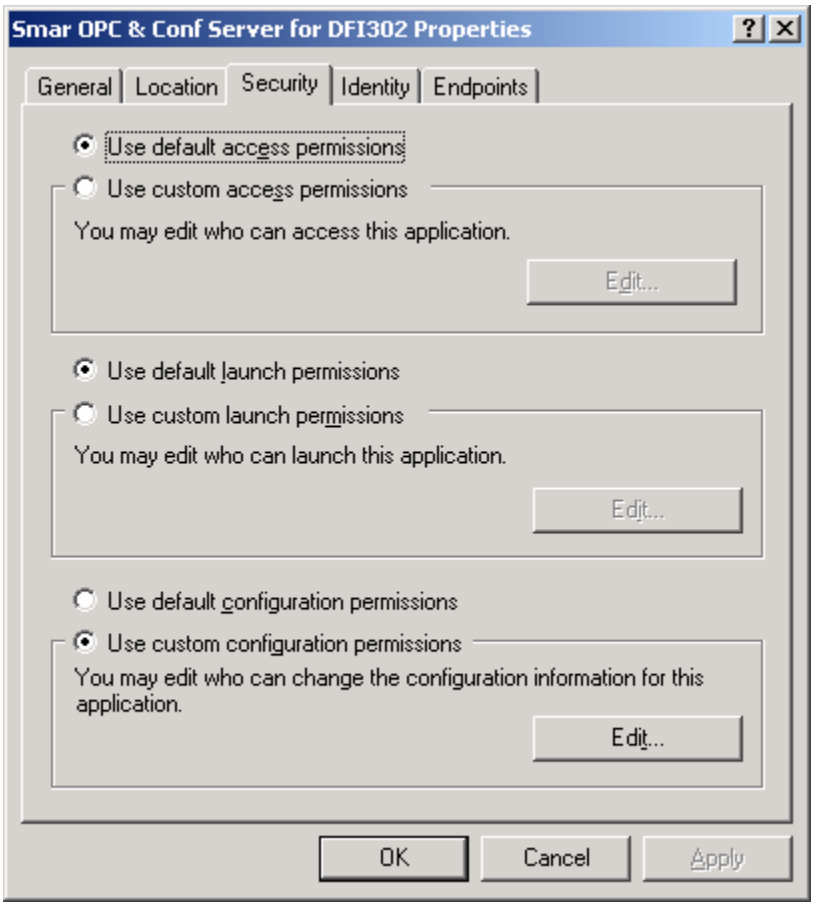
The **General** tab of the DCOM Application Properties provides general information about the application, displaying the Application name, type (local server or remote server), and location (local path or remote computer). These settings are not modifiable through the DCOM Configuration interface.

The **General** tab retrieves all of its information from subkeys of the following registry key:

HKEY_CLASSES_ROOT\CLSID\{...CLSID...}

where {...CLSID...} is the unique CLSID for the object server currently being viewed.

Location



DCOM Application Properties: Location Tab

The **Location** tab of the DCOM Application Properties is used to determine where DCOM will execute the application. Here you can specify where to run your application. There are three possible choices:

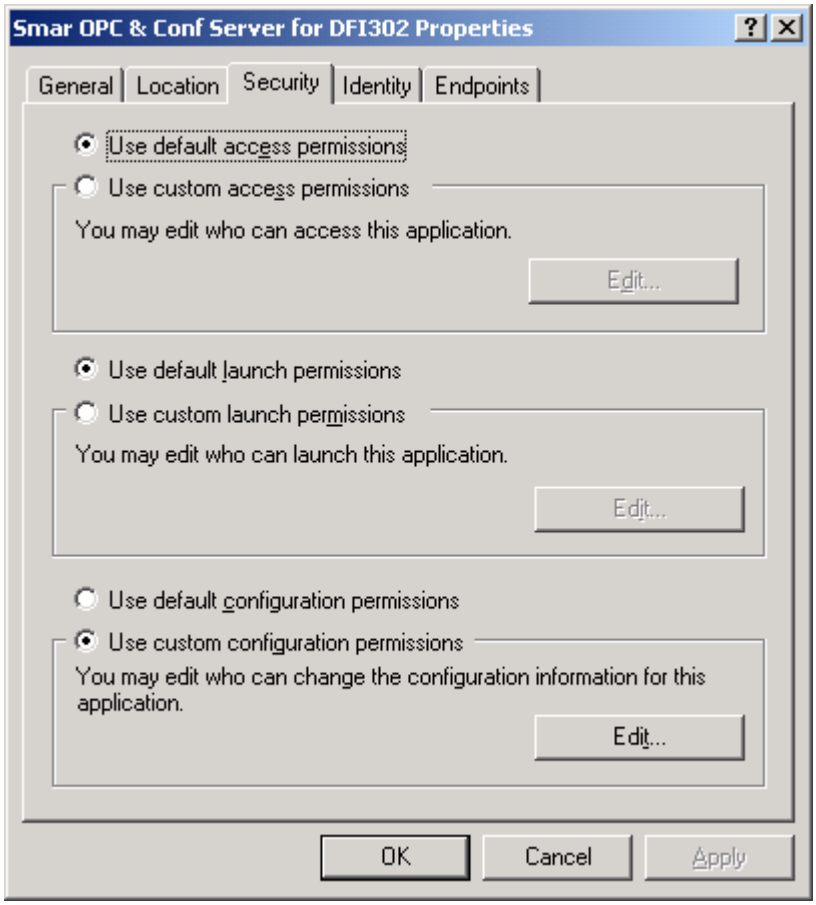
1. **Run application on the computer where the data are located.** If selected, DCOM will execute the application where the data are located. This is useful only if the application provides a data file for the server application.

2. **Run application on this computer.** This indicates that the DCOM application should run on the local machine.

3. **Run application on the following computer.** This allows you to specify a computer on which to execute. (This feature is currently unavailable on Windows NT 4.0 systems; Windows NT 4.0 does not support full security delegation.)

If more than one of the above options is selected, DCOM will use the first applicable option. Client applications may also override this setting.

Security



DCOM Application Properties: Security Tab

On the **Security** tab of the DCOM Application Properties, you can customize settings for the following individual application permissions:

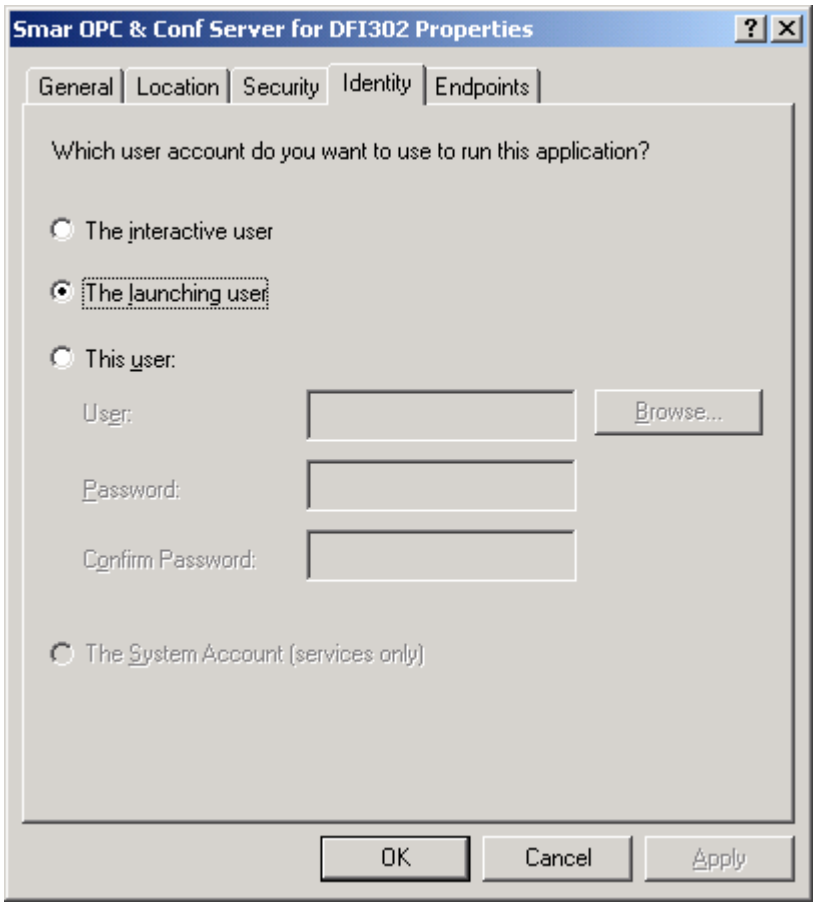
1. Access Permissions

2. Launch Permissions

3. Configuration Permissions

If you do not customize these settings, the default security settings are used. For more information about the **Security** tab, refer to the "Default Security" section.

Identity



DCOM Application Properties: Identity Tab

The **Identity** tab of the DCOM Application Properties is used to determine which account you want to use to run the application. There are four choices by which the system determines under which account your DCOM object will run.

1. **The Interactive User.** The application will run using the security context of the user currently logged onto the computer. If this option is selected and the user is not logged on, then the application will not start.

2. **The Launching User.** The application will run using the security context of the user who started the application. The launching user and the interactive user may be the same.

3. **This User.** You may specify the user whose security context will be used to run the application.

4. **The System Account.** This is available only for NT services that use DCOM.

Common Questions About DCOM

Question:
What is the difference in DCOM security configuration between machines located in a workgroup (stand-alone) and a domain environment?

Background:
Certain DCOM security configurations that are suitable for OPC client/server operation on a domain (e.g. vendor development environment) may not be suitable for workgroup or stand-alone (e.g. customer installations) operation.

Resolution:
The issue here is authentication. In a domain environment, the domain controller holds domain accounts that are valid on all machines that are part of the domain (as opposed to local accounts, which are valid only on the local machine) - the domain, while the standalone machine receives authentication from it's own Security Accounts Manager (SAM). So, if authentication between the client and server is enabled, a caller's credentials must be able to be authenticated by either the domain or the local machine. For example, if a domain "OPC" is set up to run OPC components, and the machines used are logged in this domain, the desired users from the domain must be added to the components' Access Permissions and/or Launch Permissions lists.

Question:
Can you suggest any general guidelines for setting up a secure environment?

Background:
Many process control applications require a secure environment.

Resolution:
Here are some general guidelines:

- Set the authentication level to "connect" (or higher) on the server machines.
- Create a group (in the domain) that will be allowed to launch and access the OPC objects.
- Use DCOMCNFG.EXE to include this group in ACLs for launch and access permissions for each OPC object.
- Ensure that your operators are members of this group.

This should allow different users to be logged onto various client machines. It should also allow callbacks to function as the server machine can be authenticated by the client machine.

Object Linking and Embedding (OLE)

OLE Automation in DataWorX

This appendix describes the OLE Automation features available in the DataWorX, and provides an example of how to use these features with Visual Basic (VB). The example shows how to build a form too. It also provides a way to change the type of ToolTips that are shown.

DataWorX exposes three objects via the automation interface. They can be used in VB.

Four DataWorX variable types are available:

- Application
- Point
- Register
- Redundancy Alias

These four data types have their own functions

Application Object

Methods

CreatePoint(strName As String, nScanRate As Long) As Object
- Creates a point object with the specified name and scan rate. The point object can be subsequently used to read and/or write values from/to OPC servers.

The strName parameter should meet SMAR point name convention:

i.e. [\\node\]OPCServer\TagName.

The nScanRate is the requested scan rate in milliseconds.

GetRegister(strName As String) As Object
- Returns reference to a register defined in DataWorX. The register object has the same methods and properties as the point object.

GetRedundancyAlias(strName As String) As Object
- Returns reference to a redundancy alias object defined in DataWorX.

FileOpen(strFileName As String) As Boolean
- Forces DataWorX to open the given file. Returns TRUE on success; otherwise returns FALSE.

StartRuntime()

StopRuntime()
- Starts and stops DataWorX runtime.

The following methods can be used for configuring DataWorX from VB.

FileSave() As Boolean
- Saves the current file. Returns TRUE on success.

FileSaveAs(strFileName As String) As Boolean
- Saves the current file with the name specified. Returns TRUE on success.

CreateRegister(strName As String, vtDataType As REGDATATYPE, bWriteable As Boolean, strParentGroup As String) As Object

- Creates a new register and returns reference to the newly created register object.

- The register is created with the name specified in strName, and of the type given in vtDataType (see the enumerated type REGDATATYPE).

- The bWriteable parameter tells whether the register should be writeable.

- The register is created in the group named strParent.

Note
A group named strParent must exist for successful creation of a register.

- Use a backslash to refer to the root group.

Refresh OPC()
- Everything that is connected to the OPC server is disconnected and then reconnected. A typical use for this is that when a user creates a register during runtime mode, the register does not get connected when it is created and therefore does not receive any data. When RefreshOPC() is called, the register will be connected to its OPC input.

CreateAlias(BSTR strName As String, strDefault As String, strParentGroup As String) As Object
- This is similar to CreateRegister, but this function creates a new alias.

- The strDefault parameter is the default value of the alias; other parameters see CreateRegister function.

CreateGroup(strName As String, strParentGroup As String) As Boolean
- Creates a group in another group. Returns TRUE on success.

FileNew() As Boolean
- Opens a new file; i.e. cleans up the contents of the current file.

boolean GetUnderlayingRegistersAndGroups(BSTR strParentGroup, LPVARIANT RegisterNames, LPVARIANT GroupNames);
- Given the name of the parent group (or "\" for root), returns two arrays of strings. The first contains the names of the underlying registers. The second contains the names of the underlying groups. Returns true on success.

void Shutdown();
- Causes unconditional shutdown on DataWorX. Clients should not call this function. Its primary use is with ProcViewTray.

Property

IsRuntime As Boolean
-This is a read-only property that tells whether DataWorX is in runtime mode.

Point Object

A point object can represent either an OPC point or a DataWorX register created previously by a call to Application.CreatePoint or Application.GetRegister.

Properties

Name As String
- Read-only property; name of the point or register.

Value As Variant
- Property for reading and writing; represents value of the point or register.

Quality As OPCQUALITY
- Read-only property; contains the quality of the point value. Can equal to one of QUAL_BAD, QUAL_UNCERTAIN, QUAL_GOOD

SubQuality As OPCSUBQUALITY
- Read-only property; contains extended quality information.

Timestamp As DATE
- Read-only property; contains date and time of the last value change.

Register Object

The Register object is basically an extension of the Point object. It represents a register of any type (i.e. aliases, switch aliases, etc.) defined in DataWorX.

Properties

Properties common with Point object are:

- Name As String
- Value As Variant
- Quality As OPCQUALITY
- SubQuality As OPCSUBQUALITY
- Timestamp As DATE

Other properties of the Register object are:

HiRange As Double

LoRange As Double
- Read/write properties containing ranges of the register.

UseRanges As Boolean
- Read/write property telling the user whether the ranges should be used.

ParentGroup As String
- Read-only property containing name of the group to which the register belongs.

boolean InputEnable;
- Read/write property enables/disables updates incoming from the register input. Used primarily in runtime, this property can be also accessed by right-clicking on a register in runtime and selecting **Disconnect Input.**

REGINPUTTYPE InputType;

- read only - type of the register's input, one of the REGINPUTTYPE values:

```
enum {      REGINPUT_NONE   = 0,
            REGINPUT_OPC,
            REGINPUT_REGISTER,
            REGINPUT_EXPRESSION,
            REGINPUT_CONDITION,
} REGINPUTTYPE;
```

REGTYPE RegisterType;

- read only - type of the register, on of the REGTYPE values:
enum { REGTYPE_REGISTER = 0,
REGTYPE_ALIAS,
REGTYPE_REDUNDANCY_ALIAS,
REGTYPE_REDUNDANCY_FLAG,
REGTYPE_SWITCH_ALIAS
} REGTYPE;

BSTR InputOPC(long *dwScanRate);
- Read only. Returns the name of the register's OPC input tag, and is only valid when the register's input type = REGINPUT_OPC. Also returns input scan rate.

BSTR InputRegister;
- Read only. Returns the name of the register's input register, and is only valid when the register's input type = REGINPUT_REGISTER.

BSTR InputExpression(long *dwScanRate)
- Read only. Returns the name of the register's input expression, and is only valid when the register's input type = REGINPUT_EXPRESSION. Also returns input scan rate.

VARIANT InputInitValue(BOOL *bInitEnabled);

- Read only. Returns the value to which the register should be initialized, and is only valid when the register's input type = REGINPUT_NONE. Also returns a boolean value, whether the register is set to be initialized or not.

Methods

The following methods can be used to connect the register to other registers and OPC points:

SetInputOPC(strPointName As String, nScanRate As Long) As Boolean
- Connects the register input to the specified OPC Point.

nScanRate
- Specifies the scan rate in milliseconds.

SetInputRegister(strRegisterName As String) As Boolean
- Connects the register input to another register.

SetInputExpression(strExpression As String, nScanRate As Long) As Boolean
- Connects the register input to an expression input.

nScanRate
-Specifies the scan rate in milliseconds.

SetInputNone(vInitValue As Variant) As Boolean
- Disconnects the register input. May be used to set an initial value for the register. The initial value is not set when the vInitValue parameter is empty.

AddOutputOPC(strPointName As String) As Boolean
- Adds a new register output connected to a given OPC point.

AddOutputRegister(strRegisterName As String) As Boolean
- Adds a new register output connected to a given register.

Redundancy Alias Object

Object representing a redundancy alias defined in DataWorX was previously obtained by a call to Application.GetRedundancyAlias. It can be used to force a redundancy switchover.

Methods

SwitchToPrimary
- Forces redundancy switchover to the node defined as primary.

SwitchToNextBackup
- Forces redundancy switchover to the backup node following the node that is currently used.

Property

IsPrimary As Boolean
- Returns TRUE if redundancy currently uses the node defined as primary; otherwise returns FALSE.

Example Using VB With OLE Automation

Getting Started

Open a new standard .exe project in Visual Basic. This will open a new blank Visual Basic form.

Note
Before adding anything to this form, it is necessary to add the DataWorX ActiveX component to your Toolbox. To do this, right-click anywhere in the Toolbox and select **Component** from the drop-down menu. This will open the following dialog listing all the available components.

Referencing DataWorX Functions in VB

To reference DataWorX functions in VB:

1. In the Visual Basic Editor, go to Project References.
2. In the References Dialog select DWX32.
3. Also select Smar AutoProcViewRegistrar Wrapper 2.0 type Library. Click **OK.**

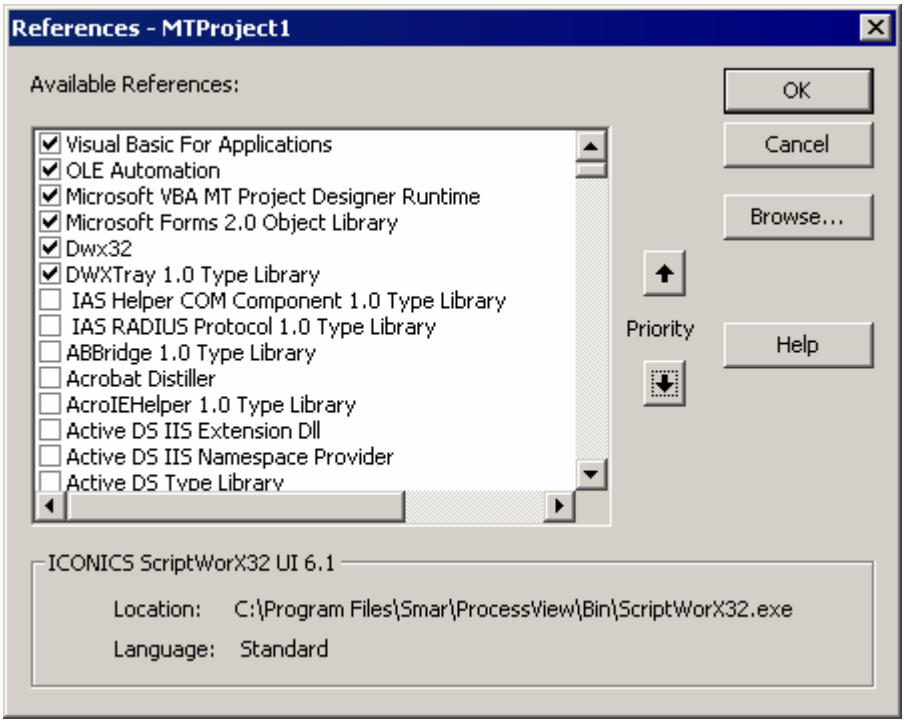
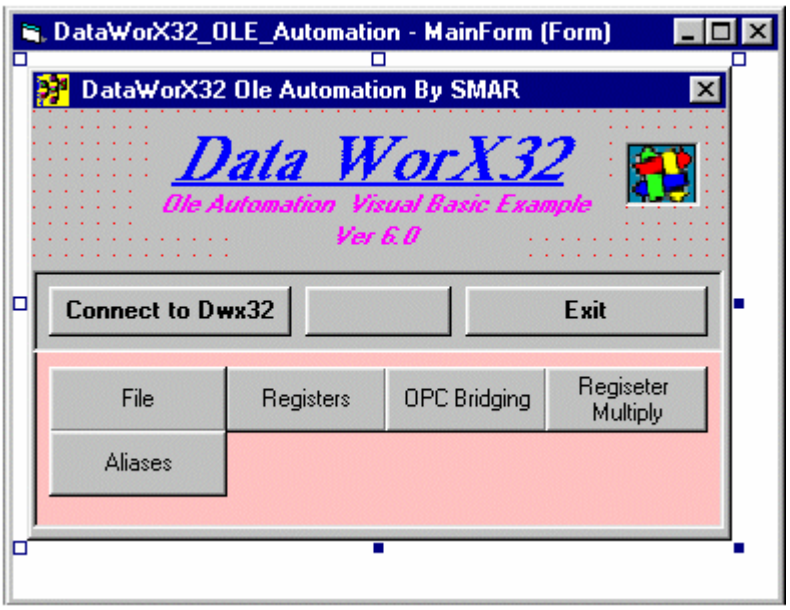


Fig. A.1. References Dialog

You are now ready to begin configuring the interface and set the code for the functions in the new VB form.

Visual Basic Form Configuration

Now we need to enter a series of shapes on the blank form using the Shape component from the Toolbox, so that your form will look like the following. This form is available under SMAR/PROCESSVIEW/Examples/DWX32 OLE. Open the file called "DWX32OLE.exe."



To configure each rectangular shape, select it and then enter the appropriate data in the **Properties window.**

Once the shapes have been configured with regard to appearance, background color, background style, border color, border style, etc. using the property window, enter the code behind each function.

Examples for the OLE Automation Interface

DataWorX OLE Automation Main Form

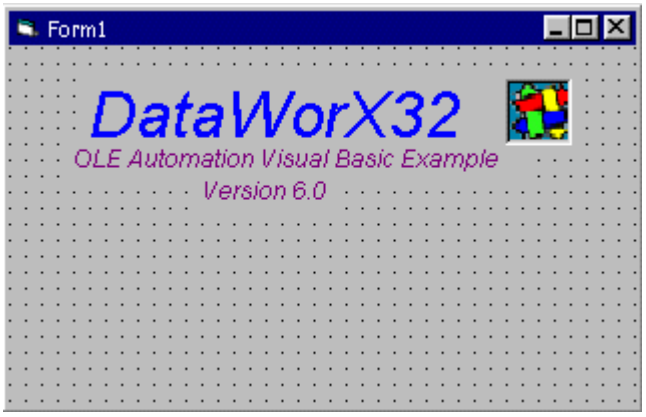
The first form that we will configure is the Main Form that will be used to access all other parts of the DataWorX example. This form allows the user to connect to DataWorX, toggle between runtime and configuration mode, open a new file, create a register, perform OPC bridging, multiply a register, create an alias, and exit the example.
Once you have opened a new Standard .exe in Visual Basic, a blank form should appear. The first step in turning this blank form into our main form is to add the top label and picture fields.

Note
We will first configure the layout of each form and will add the code behind each element in a later section.

Using the two icons on the left, insert three labels and one picture box. Using the Properties Page, in Visual Basic enter the following Data for each element added.

	Label1	Label2	Label3	Label4
Caption	DataWorX	OLE Automation in Visual Basic Example	Version 6.0	N/A
Font	Arial, Italic, 24pt	Arial, Italic, 9pt	Arial, Italic, 9pt	N/A
ForeColor	Blue	Dark Purple	Dark Purple	N/A
Picture	N/A	N/A	N/A	(Bitmap) use search dialog to choose icon.

Your form should now look like the following figure:

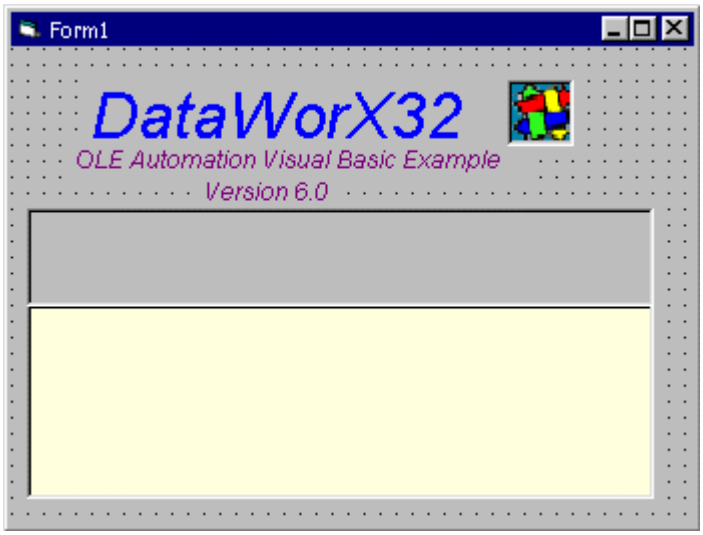


The next step in configuring our Main Form is adding two label areas in which we will place the command buttons. Using the Label button again, draw two rectangular areas below the labels you just entered. This time the labels will be used as display areas instead of text displays, so the

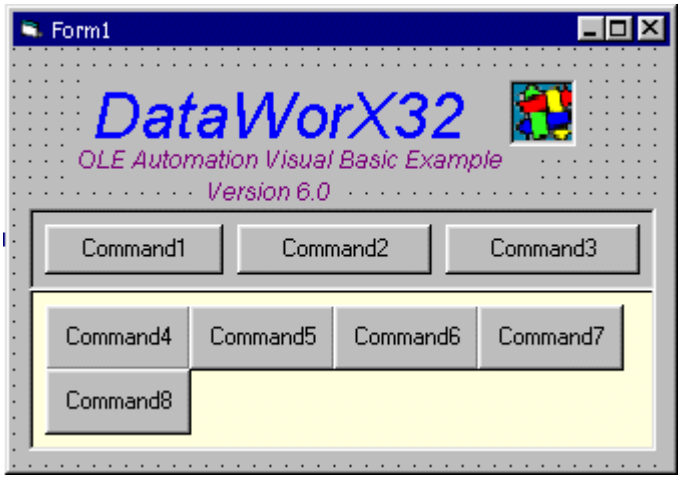
properties that must be changed are the BackColor and BorderStyle. Change these properties to the following:

	Label4	Label5
BackColor	Default gray	Light yellow or orange
BorderStyle	1 - Fixed Style	1 - Fixed Style

After fixing these properties, your form should look like the following figure.



Next insert eight command buttons and arrange them as shown below:

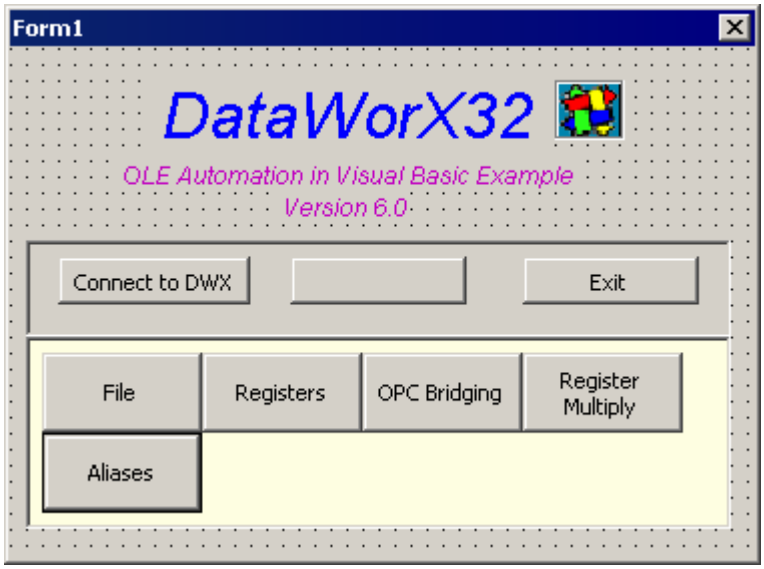


It is now necessary to enter the proper button names and captions. Enter the following properties on the Properties Page for each button.

	(Name)	Caption
Command1	cmdConnectToDWX32	Connect to DWX32
Command2	cmdStartStop	
Command3	cmdCancelExit	Exit
Command4	cmd_File	File

	(Name)	Caption
Command5	cmdRegisters	Registers
Command6	cmd_OPC_Bridging	OPC Bridging
Command7	cmd_Register_Mult	Register Multiply
Command8	cmd_Aliases	Aliases

After entering these data, your form is complete and should look like the following:



To add the code behind this form, double click on each object. Performing this action will launch the code window where the user can enter the code that will define the functionality of the object.

Connect to DataWorX button

```
Private Sub cmdConnectToDWX32_Click()

    Check_GenRegistrar

    Dispatch_Dwx32

    Dwx_RunTime_Support

End Sub
```

Stop/Start

```
Private Sub cmdStartStop_Click()

    'Catch any Automation Errors

    On Error Resume Next

    cmdStartStop.FontItalic = True

    cmdStartStop.FontBold = True

    If dwx.IsRuntime Then

        Screen.MousePointer = vbHourglass

        dwx.StopRuntime
```

```
        Screen.MousePointer = VBDEFULT
        cmdStartStop.Caption = "Start"
    Else
        Screen.MousePointer = vbHourglass
        dwx.StartRuntime
        cmdStartStop.Caption = "Stop"
        Screen.MousePointer = VBDEFULT
    End If
End Sub
```

Exit

```
Private Sub cmdCancelExit_Click()
    ' Release Gen Registrar
    Set dwx = Nothing
    ' Release GWX32 dispatch
    Set GenReg = Nothing
    End
    Unload Me 'close the Main Form
End Sub
```

File

```
Private Sub cmd_File_Click()
    File_Form.Show
End Sub
```

Registers

```
Private Sub cmdRegisters_Click()
    RegistersForm.Show
End Sub
```

OPC Bridging

```
Private Sub cmd_OPC_Bridging_Click()
    OPC_Bridging_Form.Show
End Sub
```

Register Multiply

```
Private Sub cmd_Register_Mult_Click()

    Register_Mult_Form.Show

End Sub
```

Aliases

```
Private Sub cmd_Aliases_Click()

End Sub
```

File Form

The next form that we will configure is the File form. This form will launch when the **File** button is pushed on the Main form. To add a new form, either click the **Add Form** button from the main toolbar or select **Project - Add Form** from the main menu. A blank form will appear, and as we did with the previous form, we must add elements to create our desired form.

First we must add a frame and a picture box. The frame will contain all of the elements that make up the form, and the picture box will contain a picture file that will act as the header for the form. Configuring these elements is actually quite easy. To set the frame properly, delete the default text for the caption so that the **Caption** field is blank. As for the picture box, insert a bitmap that will be useful as a header to this form. For our example, we have selected a bitmap that contains the DataWorX icon along with a DataWorX label. After these few changes have been made, the form should look like the following.

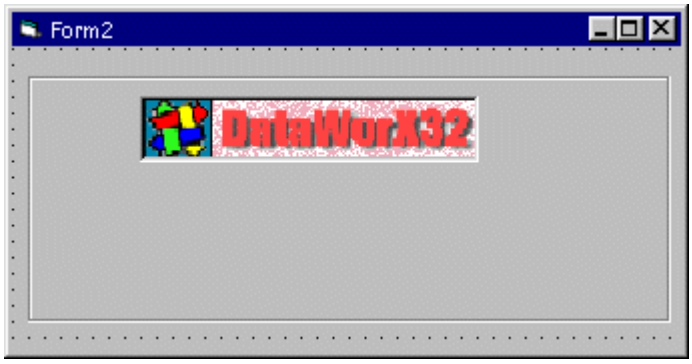


Fig. A.1. File Form in the early stages…

Next it is necessary to enter a label and five Command buttons. The form should now look like the following:

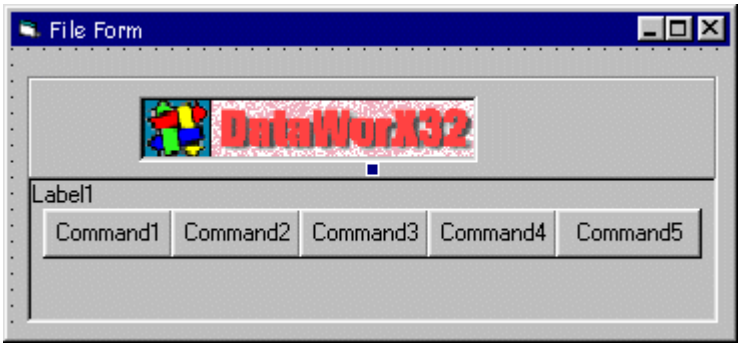


Fig. A.2. File Form

The area in which the buttons are located is simply a label. It is now time to configure these newly inserted objects. Using the Properties page, configure those items in the following way:

	(Name)	Caption
Label1	Label1	
Command1	cmd_Open	Open
Command2	cmd_New	New
Command3	cmd_Save	Save
Command4	cmd_Save_As	Save As
Command5	cmd_Exit	Exit

After such configuration has occurred it is necessary to enter the final elements of the form. These elements are a label and a text box. Once you have entered these elements, your form should look like the following figure.

Fig. A.3.

To configure the new elements, enter the following data in the properties page.

	(Name)	Text
Label2	Txt_Lbl_File_Name	File Path & Name:
Text1	Txt_Dwx32_File_Name	Dwx

The final File Form should appear as shown below:

Fig. A.4. File Form

Open

```
Private Sub cmd_Open_Click()

Dim b As Boolean

    Dwx32_File_Name = Txt_Dwx32_File_Name.Text & "." & Dwx32_File_Ext

    Txt_Lbl_File_Name.Text = "File Path & Name: "


    If Not b = dwx.FileOpen(Dwx32_File_Name) Then

         MsgBox "Failed to Open " & Dwx32_File_Name & " File."

    End If

End Sub
```

New

```
Private Sub cmd_New_Click()

    dwx.FileNew

End Sub
```

Save

```
Private Sub cmd_Save_Click()

  If Not dwx.IsRuntime Then

    dwx.FileSave

  Else

    MsgBox "Make sure DataWorX is NOT in Run-time."

  End If

End Sub
```

Registers Form

Next we will create the **Registers** form. This form will allow the user to connect a register to a point, create a register, read/write a register, or edit a register. A register acts like a point. The only difference is that a register has been "'enhanced" by DataWorX. In fact, a register is actually a point from an OPC server, which has been "taken over" by DataWorX.

To begin creating the form, insert a new form into your DataWorX Visual Basic Project. Once you have a blank form, use the Label component to add two labels to the form so that it looks like the following figure.

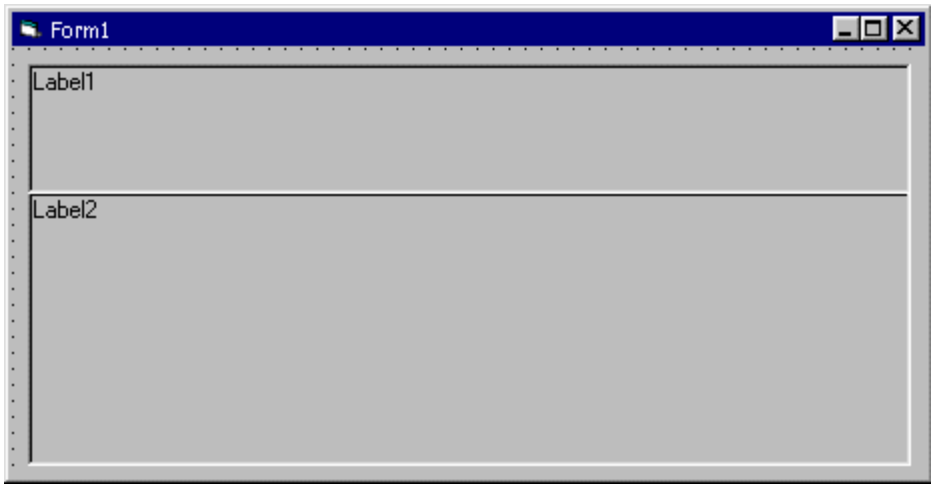


Fig. A.5.

Using the Properties page, delete the text for the Caption element so that two blank areas are present. Next we will add the five command buttons, which will perform most of the action on this form. To do so, select the Command Button icon and insert five buttons so that your form looks like the figure below.

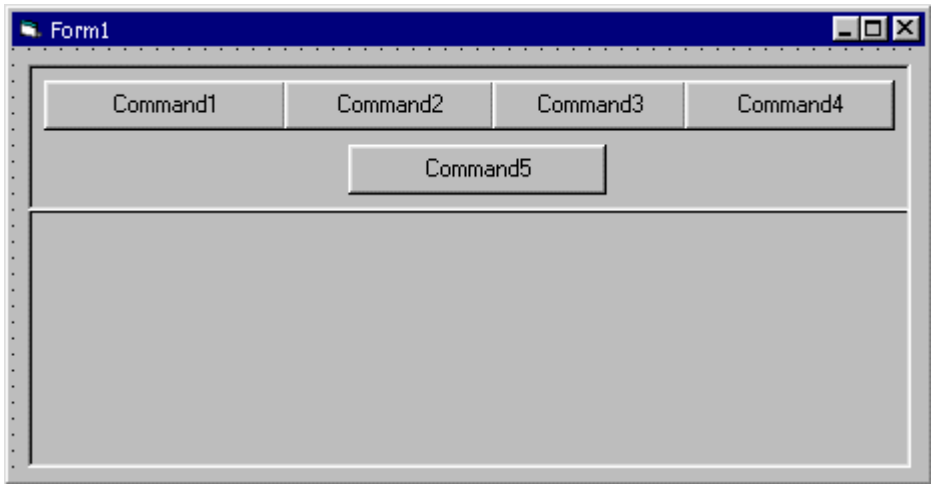


Fig. A.6.

Next, using the Properties pages enter the following information for each button:

	(Name)	Caption
Command1	cmdConnect_Point	Connect to Point
Command2	cmdCreateRegister	Create Register
Command3	cmdRW_Register	Read/Write Register
Command4	cmd_Edit_Register	Edit Register
Command5	cmdBack_ToMain	Main Window

After you have entered the proper information in the Properties window, enter three frames in the Label2 area of the form.

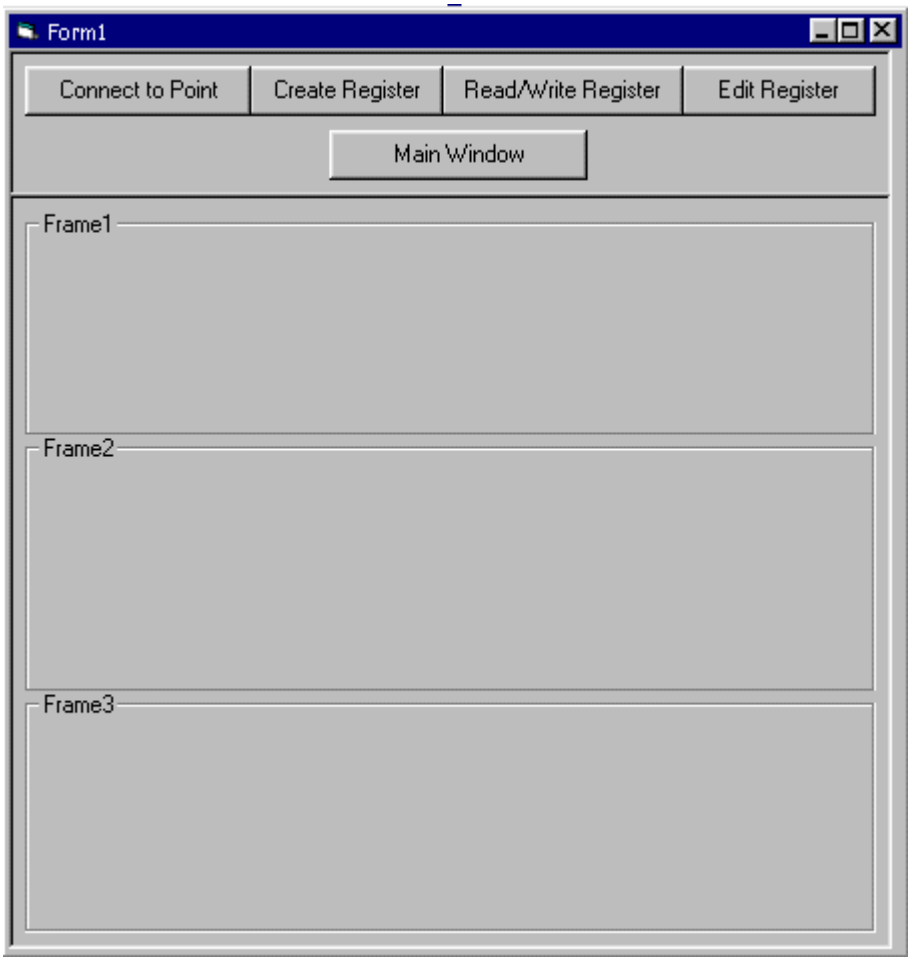


Fig. A.7.

Note that there are three blank frames. Each of these frames relates to one of the command buttons from the first section of this form. Depending on which button is pushed, only one frame will be visible.

The first frame will display when the **Connect to Point** button is pushed. For this frame, change the Caption of the frame to 'Type the Point/Tag Information'; change the Font to Arial, Italic, 8pt; and change the ForeColor to blue. Now insert the following items into the frame and arrange them as shown in the following figure: six labels, five text boxes, and two command buttons.

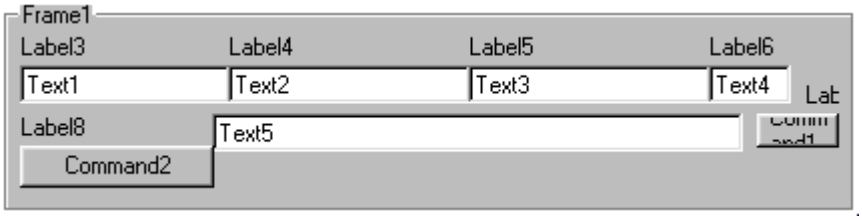


Fig. A.8.

Change the properties for each of these newly inserted objects according to the following table.

	(Name)	Caption/Text
Label3	lbl_Node_Name	Node Name:
Label4	lbl_Server_Name	Server Name:
Label5	lbl_Tag_Name	Tag Name:
Label6	lbl_Interval	Interval:
Label7	lbl_Ms	Ms.
Label8	lbl_Point_Value	Point Value:
Text1	TxtNode_Name	
Text2	TxtOPC_Server_Name	SMAR.Simulator
Text3	TxtTag_Name	SimulatePLC.Random
Text4	TxtInterval	500
Text5	TxtPoint_Value	
Command1	cmdGo	Go
Command2	cmdDisConnect_Point	Disconnect

After you have entered the above properties your form should look like the following figure.

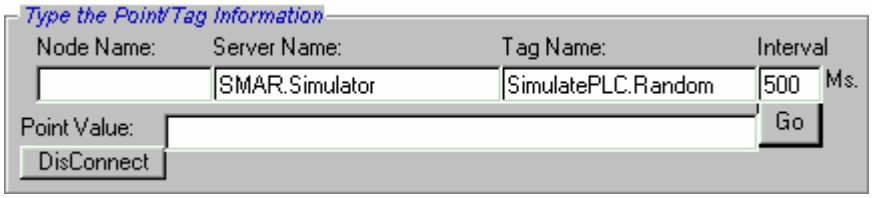


Fig. A.9.

The second frame will display when the **Create Register** button is clicked. For this frame, change the Caption of the frame to 'Type the Register Information'; change the Font to Arial, Italic, 8pt; and change the ForeColor to blue. Now insert the following items into the frame, and arrange them as shown in the following figure: four labels, two text boxes, two combo boxes, and two command buttons.

Fig. A.10.

Change the properties for each of these newly inserted objects according to the following table.

	(Name)	Caption/Text
Label3	lbl_Register_Name	Register Name:
Label4	lbl_Type	Type:
Label5	lbl_Writeable	Writeable:
Label6	lbl_Group_Name	Group Name:
Text1	TxtRegister_Name	Reg_Name
Text2	TxtGroup_Name	\
Combo1	Cmb_Register_Type	Bool
Combo2	Cmb_Writeable	True
Command1	cmdAdd_Register	Add Register
Command2	cmdExit	Exit

After you have entered the above properties, your form should look like the following figure.

Fig. A.11.

The third frame will display when the **Read/Write Register** button is clicked. For this frame, change the Caption of the frame to 'Type the Register Information'; change the Font to Arial, Italic, 8pt; and change the ForeColor to blue. Now insert the following items into the frame, and arrange them as shown in the following figure: four labels, two text boxes, one frame, two option buttons, and two command buttons.

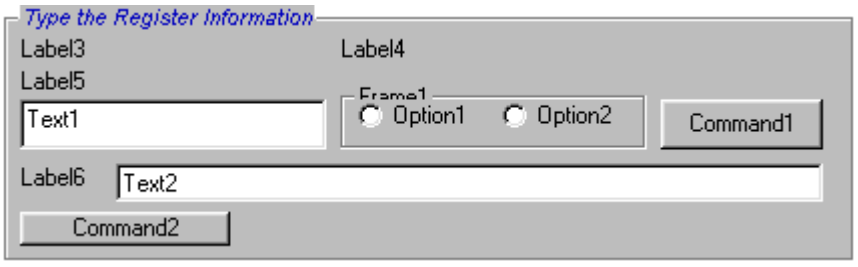


Fig. A.12.

Change the properties for each of these newly inserted objects according to the following table.

	(Name)	Caption/Text
Label3	LblRegister_Name	Register Name:
Label4	Lbl_RW_Register	Read/Write Register
Label5	Lbl_IE_Reg_Name	Ie, GroupA.SubGroupA.Reg 1
Label6	Lbl_Register_Value	Register Value:
Text1	Txt_Register_Name	Reg_Name
Text2	Txt_Register_Value	
Frame2	Frame2	
Option1	Opt_Read	Read
Option2	Opt_Write	Write
Command1	cmd_RW_Register	
Command2	cmd_DisConnect_Reg	DisConnect

After you have entered the above properties, your form should look like the following figure.

Fig. A.13.

Connect to Point

```
Private Sub cmdConnect_Point_Click()

    Show_Connect_Point_support

End Sub
```

Create Register

```
Private Sub cmdCreatRegister_Click()

    Show_CreateRegister_support

End Sub


Private Sub Show_CreateRegister_support()

'to Show all what you need"functions" for Create a Register
```

```
    CreateRegister_Frame.Enabled = True
    CreateRegister_Frame.Visible = True
    cmdCreatRegister.Enabled = False
    set_Register_Type_ComboBox
    set_Writeable_ComboBox
End Sub
Private Sub Hide_CreateRegister_support()
'to Hide all functions for Create a Register
    CreateRegister_Frame.Enabled = False
    CreateRegister_Frame.Visible = False
    cmdCreatRegister.Enabled = True
End Sub
```

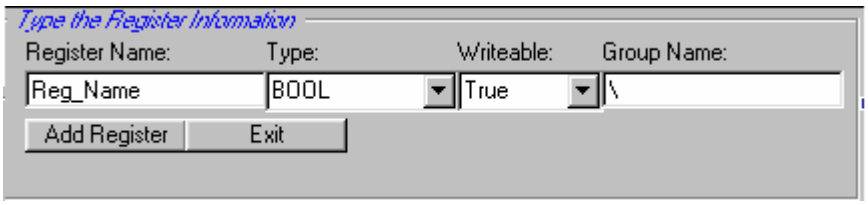


Fig. A.14.

Type the Register Information:

Register Name

```
Private Sub lbl_Register_Name_Change()
End Sub
```

Register Name Text Box

```
Private Sub TxtRegister_Name_Change()
End Sub
```

Type

```
Private Sub lbl_Type_Change()
End Sub
```

Register Type Combo Box

```
Private Sub Cmb_Register_Type_Change()
End Sub
```

Writeable

```
Private Sub lbl_Writeable_Change()

End Sub
```

Writeable Combo Box

```
Private Sub Cmb_Writeable_Change()

End Sub
```

Group Name

```
Private Sub lbl_Group_Name_Change()

End Sub
```

Group Name Text Box

```
Private Sub TxtGroup_Name_Change()

End Sub
```

Add Register

```
Private Sub cmdAdd_Register_Click()

Dim RegName, GroupName As String

Dim RegType As REGDATATYPE

Dim RegWriteable, b As Boolean

Dim Dwx32_Register11 As Boolean

   RegName = TxtRegister_Name.Text

   RegType = Get_Reg_Type(Cmb_Register_Type)

   RegWriteable = Get_Reg_Writeable(Cmb_Writeable)

   GroupName = TxtGroup_Name.Text

      'make sure Dwx32 is Running.

    If dwx Is Nothing Then

        b = MsgBox("Make sure DataWorX is in Run-time Mode" & Chr(13) &
"Try Again..", vbOKOnly, "Dwx32 Ole")

        Show_CreateRegister_support ' go back and change the Information.

        Exit Sub  'do not do anything more exit the sub now.

    Else

        Screen.MousePointer = vbHourglass              'change the mouse
to busy..

    Set Dwx32_Register = dwx.CreateRegister(RegName, RegType,
RegWriteable, GroupName)

        Screen.MousePointer = VBDEFULT                 'change the mouse
to default..

    End If
```

```
        If (Dwx32_Register Is Nothing) Then

        b = MsgBox("Make sure you have a Valid Group Name, " & Chr(13) &
_

        "Register Type or/and Writeable Option", vbOKOnly, "Dwx32 Ole")

    Else

         Show_CreateRegister_support ' go back and change the
Information.

        Exit Sub           'do not do anything more exit the sub now.

    End If

    Set Dwx32_Register = Nothing

End Sub
```

Exit

```
Private Sub cmdExit_Click()

    Set Dwx32_Register = Nothing    'release the unused memory

    Hide_CreateRegister_support      'hide support functions to Create a
Register

    cmdCreateRegister.Enabled = True 'reEnable the Create Register Button

End Sub
```

Read/Write Register

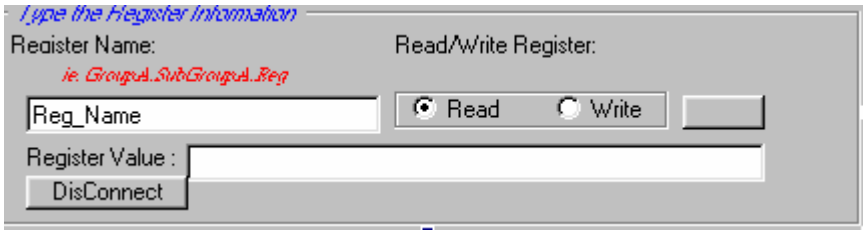


Fig. A.15.

Fig.: Read/Write Register Frame Enabled

```
Private Sub cmdRW_Register_Click()

    If Opt_Read Then

        cmd_RW_Register.Caption = "Read"

    Else

        cmd_RW_Register.Caption = "Write"

    End If

    Show_Read_Write_Register_support

End Sub
```

Read Write Register Support Functions

```
Private Sub Show_Read_Write_Register_support()
'to Show all what you need"functions" for Read/Write a Register
    Txt_Register_Value.Text = ""
    RW_Register_Frame.Enabled = True
    RW_Register_Frame.Visible = True
    cmdRW_Register.Enabled = False
End Sub

Private Sub Hide_Read_Write_Register_support()
'to Hide all the functions for Read/Write a Register
    RW_Register_Frame.Enabled = False
    RW_Register_Frame.Visible = False
    cmdRW_Register.Enabled = True
End Sub

Private Sub cmdRW_Register_Click()
    If Opt_Read Then
        cmd_RW_Register.Caption = "Read"
    Else
        cmd_RW_Register.Caption = "Write"
    End If
    Show_Read_Write_Register_support
End Sub

Private Sub Label1_Click()

End Sub

Private Sub Opt_Read_Click()
    cmd_RW_Register.Caption = "Read"
End Sub

Private Sub cmd_DisConnect_Reg_Click()
    Hide_Read_Write_Register_support
End Sub
```

```
Private Sub Opt_Write_Click()

    cmd_RW_Register.Caption = "Write"

End Sub


Private Sub cmd_RW_Register_Click()

Dim Reg_Name As String

Dim b As Boolean


    Reg_Name = Txt_Reg_Name.Text


    'make sure Dwx32 is Running.

    If dwx Is Nothing Then

        b = MsgBox("Make sure DataWorX is in Run-time Mode" & Chr(13) &
"Try Again..", vbOKOnly, "Dwx32 Ole")

        Show_Read_Write_Register_support ' go back and change the
Information.

        Exit Sub           'do not do anything more exit the sub now.

    Else

        Screen.MousePointer = vbHourglass               'change the mouse
to busy..

        Set Dwx32_Register = dwx.GetRegister(Reg_Name)

        Screen.MousePointer = VBDEFULT                  'change the mouse
to default..

    End If


    If Dwx32_Register Is Nothing Then

        b = MsgBox("Make sure you have a Valid Register Name.." & Chr(13)
& _

        "NOTE: Name is a Case Sensitive.", vbOKOnly, "Dwx32 Ole")

        Show_Read_Write_Register_support  '  go  back  and  change  the
Information.

        Exit Sub           'do not do anything more exit the sub now.

    End If


    Screen.MousePointer = vbHourglass               'change the mouse to
busy..

    On Error Resume Next

    If Opt_Read Then

       Txt_Register_Value.Text = Dwx32_Register.Value
```

```
    Else

        Dwx32_Register.Value = Txt_Register_Value.Text

    End If

    Screen.MousePointer = VBDEFULT


    Set Dwx32_Register = Nothing


End Sub
```

Register Name

```
Private Sub LblRegister_Name_Change()

End Sub
```

Register Name Text Box

```
Private Sub Txt_Reg_Name_Change()

End Sub
```

Read/Write Register

```
Private Sub Lbl_RW_Register_Change()

End Sub
```

Read radio button

```
Private Sub Opt_Read_Click()

    cmd_RW_Register.Caption = "Read"

End Sub
```

Write radio button

```
Private Sub Opt_Write_Click()

    cmd_RW_Register.Caption = "Write"

End Sub
```

Read/Write button

```
Private Sub cmd_RW_Register_Click()

Dim Reg_Name As String

Dim b As Boolean

    Reg_Name = Txt_Reg_Name.Text

    'make sure Dwx32 is Running.

    If dwx Is Nothing Then

        b = MsgBox("Make sure DataWorX is in Run-time Mode" & Chr(13) &
"Try Again..", vbOKOnly, "Dwx32 Ole")

        Show_Read_Write_Register_support ' go back and change the
Information.
```

```
        Exit Sub           'do not do anything more exit the sub now.

    Else

        Screen.MousePointer = vbHourglass               'change the mouse
to busy..

        Set Dwx32_Register = dwx.GetRegister(Reg_Name)

        Screen.MousePointer = VBDEFULT                  'change the mouse
to default..

    End If


    If Dwx32_Register Is Nothing Then

        b = MsgBox("Make sure you have a Valid Register Name.." & Chr(13)
& _

        "NOTE: Name is a Case Sensitive.", vbOKOnly, "Dwx32 Ole")

        Show_Read_Write_Register_support  '  go  back  and  change  the
Information.

        Exit Sub           'do not do anything more exit the sub now.

    End If


    Screen.MousePointer = vbHourglass               'change the mouse to
busy..

    On Error Resume Next

    If Opt_Read Then

       Txt_Register_Value.Text = Dwx32_Register.Value

    Else

        Dwx32_Register.Value = Txt_Register_Value.Text

    End If

    Screen.MousePointer = VBDEFULT

    Set Dwx32_Register = Nothing

End Sub
```

Register Value

```
Private Sub Lbl_Register_Value_Change()

End Sub
```

Register Value Text Box

```
Private Sub Txt_Register_Value_Change()

End Sub
```

Disconnect

```
Private Sub cmd_DisConnect_Reg_Click()

    Hide_Read_Write_Register_support

End Sub
```

Edit Register

```
Private Sub cmd_Edit_Register_Click()

    Edit_Register_Form.Show

End Sub
```

Main Window

```
Private Sub cmdBack_ToMain_Click()

    Timer1.Enabled = False   'make sure to stop the Timer

    MainForm.Show            'Show the main window

    Unload Me                'close this window down

End Sub
```

Register Form Support Functions:

```
Private Sub Form_Load()

'this function will start every time the form is loaded.

    Timer1.Enabled = False      'make sure the timer is off

    Timer1.Interval = 50        'set the timer interval to a 50Ms

    set_RegisterType_ComboBox

    set_Writeable_ComboBox

    Hide_ConnectToPoint_support 'make sure the Connect to a point support
function is hidden

    Hide_CreateRegister_support

    Hide_Read_Write_Register_support

End Sub
```

Edit Register Form

When the **Edit Register** button is clicked, the Edit Register form will appear. We will now configure this form, and as noted earlier, we will enter the code behind the form at a later point.

First it is necessary to insert a new form. Do so by clicking the form icon from the toolbar or selecting **Project - Add Form.** Once the form is added,* insert three labels and change their BorderStyle to '1 - Fixed Single'.

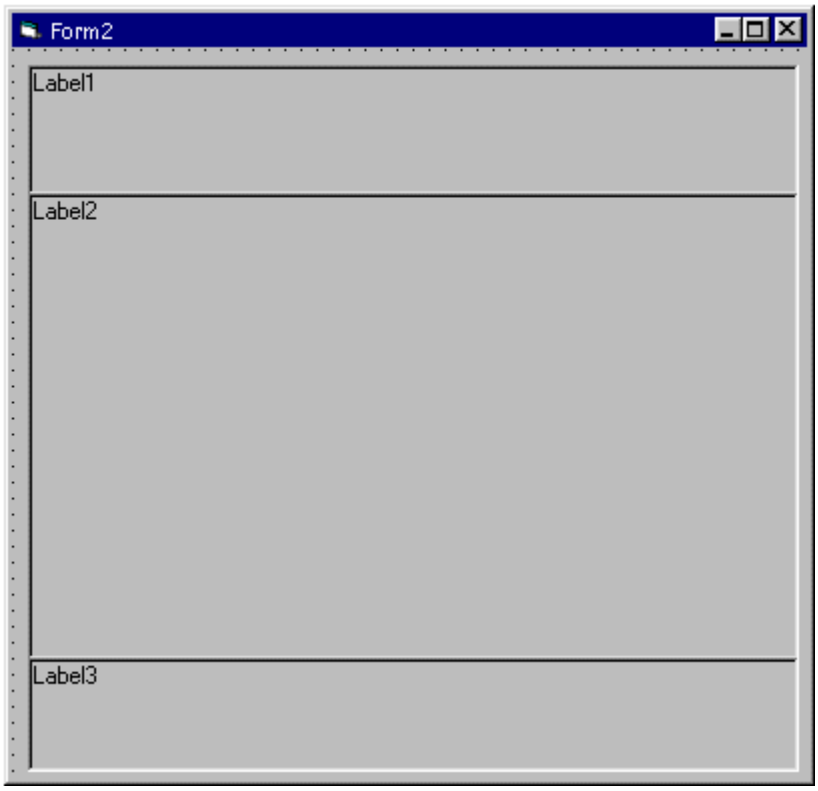


Fig. A.16.

In the first label section, add two labels, two command buttons, and one text box with the following properties:

	(Name)	Caption
Label1	Label1	
Lable2	lbl_Register_Name	Register Name:
Label3	lbl_IE_Reg_Name	Ie, GroupA.SubGroupA.Reg1
Command1	cmd_Connect_Register	Connect to Register
Command2	cmd_Cancel	Cancel
Text1	Txt_Reg_Name	Reg_Name

Once the appropriate components have been added and their properties edited to reflect the changes the user made, the form should look like the following:

Fig. A.17.

In the second label section, add 12 labels, one command button, two frames, and 10 text boxes with the following properties:

	(Name)	Caption
Label4	lbl_Ranges	Ranges:
Lable5	lbl_Register_Value	Register New Value
Label6	lbl_Lo	Lo:
Label7	lbl_Hi	Hi:
Label8	lbl_Node_Name	Node Name:
Label9	lbl_Server_Name	Server Name:
Label10	lbl_Tag_Name	Tag Name:
Label11	lbl_Scan_Rate	Scan Rate:
Label12	lbl_Ms	Ms.
Label13	lbl_Out_Node_Name	Node Name:
Label14	lbl_Out_Server_Name	Server Name:
Label15	lbl_Out_Tag_Name	Tag Name:
Command1	cmd_Update	Update
Frame1	Connect_toPoint_Frame	Type the New Point/Tag Information
Frame2	Output_point_Frame	Type the New Output Point/Tag Information
Text1	Txt_Lo	0
Text2	Txt_Hi	100
Text3	Txt_Register_New_Value	99
Text4	Txt_Node_Name	
Text5	TxtOPC_Server_Name	SMAR.Simulator
Text6	TxtTag_Name	SimulatePLC.Random
Text7	Txt_Scan_Rate	50
Text8	Txt_Output_Node_Name	
Text9	Txt_Output_OPC_Server	SMAR.Simulate
Text10	Txt_Output_Tag_Name	SimulatePLC.OUTPUTS.IN

Once the appropriate components have been added and their properties edited to reflect the changes the user made, the label2 section of the form should look like the following figure.

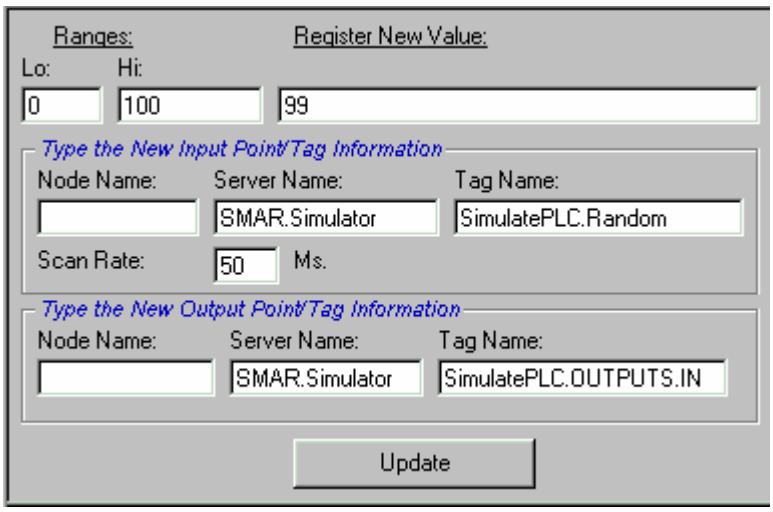


Fig. A.18.

In the third label section, add two command buttons with the following properties:

	(Name)	Caption
Command1	cmd_Registers_Window	Registers Window
Command2	cmd_Main_Window	Main Window

Once the appropriate components have been added and their properties edited to reflect the changes the user made, the label3 section of the form should look like the following figure.

Fig. A.19.

Once all of these changes have been made, it is necessary to save the form and give it the appropriate form name. To save the form, click **File -Save Form As.** A **Save As** dialog box will appear. Type in the file name **EditRegisterForm,** and then click **OK.** To change the name of the form and the caption of the form, use the Properties window and type in the following properties for those components:

	(Name)	Caption
Form4	Edit_Register_Form	Edit Register

After completing the properties changes, your finished Edit Register form should look like the following figure.

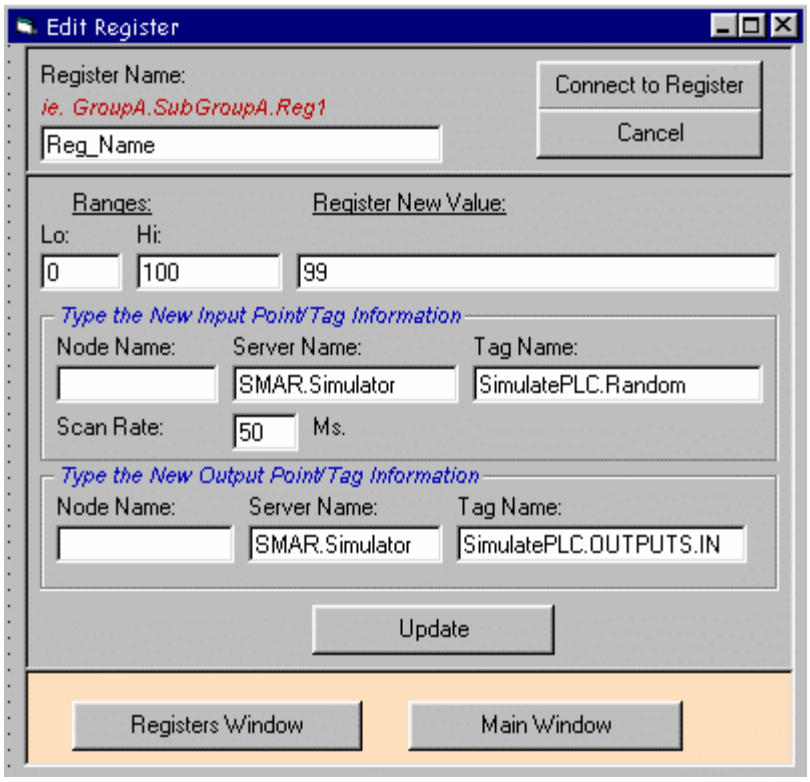


Fig. A.20. Edit Form

Register Name

```
Private Sub lbl_Register_Name_Change()

End Sub
```

Register Name Text Box

```
Private Sub Txt_Reg_Name_Change()

End Sub
```

Connect to Register

```
Private Sub cmd_Connect_Register_Click()

Dim Reg_Name As String

Dim b As Boolean

    Reg_Name = Txt_Reg_Name.Text

    'make sure Dwx32 is Running.

    If dwx Is Nothing Then

        b = MsgBox("Make sure DataWorX is in Run-time Mode" & Chr(13) &
"Try Agin..", vbOKOnly, "Dwx32 Ole")

        Form_Load      ' go back and change the Information.

        Exit Sub       'do not do anything more exit the sub now.

    Else
```

```
        Screen.MousePointer = vbHourglass               'change the mouse
to busy..

        Set Dwx32_Register = dwx.GetRegister(Reg_Name)

        Screen.MousePointer = VBDEFULT                  'change the mouse
to default..

    End If

    'make sure the Register dose exist.

    If Dwx32_Register Is Nothing Then

        b = MsgBox("Make sure you have a Valid Register Name.." & Chr(13)
& _

        "NOTE: Name is a Case Sensitive.", vbOKOnly, "Dwx32 Ole")

        Form_Load          ' go back and change the Information.

        Exit Sub           'do not do anything more exit the sub now.

    End If

    Edit_Register_Form.Height = 6060

    Edit_Register_Form.Width = 5925

End Sub
```

Cancel

```
Private Sub cmd_Cancel_Click()

    CleanUp

    RegistersForm.Show

End Sub
```

Ranges

```
Private Sub lbl_Ranges_Change()

End Sub
```

Register New Value

```
Private Sub lbl_Register_Value_Change()

End Sub
```

Lo

```
Private Sub lbl_Lo_Change()

End Sub
```

Lo Text Box

```
Private Sub Txt_Lo_Change()

End Sub
```

Hi

```
Private Sub lbl_Hi_Click()

End Sub
```

Hi Text Box

```
Private Sub Txt_Hi_Change()

End Sub
```

Register New Value

```
Private Sub lbl_Register_Value_Change()

End Sub
```

Register New Value Text Box

```
Private Sub Txt_Register_New_Value_Change()

End Sub
```

Type the New Input Point/Tag information

```
Private Sub Connect_toPoint_Frame_DragDrop(Source As Control, X As
Single, Y As Single)

End Sub
```

Type the New Output Point Tag Information

```
Private Sub Get_Ouput_Reg_Point()

Dim b As Boolean
```

Node Name

```
Private Sub lbl_Node_Name_Change()

End Sub
```

Node Name Text box

```
Private Sub Txt_Node_Name_Change()

End Sub
```

Server Name

```
Private Sub lbl_Server_Name_Change()

End Sub
```

Server Name Text Box

```
Private Sub TxtOPC_Server_Name_Change()

End Sub
```

Tag Name

```
Private Sub lbl_Tag_Name_Change()

End Sub
```

Tag Name Text Box

```
Private Sub TxtTag_Name_Change()

End Sub
```

Scan Rate

```
Private Sub lbl_Scan_Rate_Change()

End Sub
```

Scan Rate Text Box

```
Private Sub Txt_Scan_Rate_Change()

End Sub
```

Type the new Output Point/Tag Information Frame

```
Private Sub Output_point_Frame_DragDrop(Source As Control, X As Single, Y
As Single)

End Sub
```

Node Name Text box

```
Private Sub Txt_Out_Node_Name_Change()

End Sub
```

Server Name

```
Private Sub lbl_Out_Server_Name_Change()

End Sub
```

Server Name Text Box

```
Private Sub Txt_Output_OPC_Server_Change()

End Sub
```

Tag Name

```
Private Sub lbl_Out_Tag_Name_Change()

End Sub
```

Tag Name Text Box

```
Private Sub Txt_Output_Tag_Name_Change()

End Sub
```

Update

```
Private Sub cmd_Update_Click()

    'make sure Dwx32 is Running.

    If dwx Is Nothing Then

        b = MsgBox("Make sure DataWorX is in Run-time Mode" & Chr(13) &
"Try Again..", vbOKOnly, "Dwx32 Ole")

        Form_Load      ' go back and change the Information.

        Exit Sub       'do not do anything more exit the sub now.

    Else

        Screen.MousePointer = vbHourglass               'change the mouse
to busy..

        Set Dwx32_Register = dwx.GetRegister(Txt_Reg_Name.Text)

        Screen.MousePointer = VBDEFULT                  'change the mouse
to default..

    End If


    'make sure the Register dose exist.

    If Dwx32_Register Is Nothing Then

        b = MsgBox("Make sure you have a Valid Register Name.." & Chr(13)
& _

        "NOTE: Name is a Case Sensitive.", vbOKOnly, "Dwx32 Ole")

        Form_Load          ' go back and change the Information.

        Exit Sub           'do not do anything more exit the sub now.

    End If


    If Not (Txt_Hi.Text = "") Then

        Dwx32_Register.HiRange = Txt_Hi.Text

    End If


    If Not (Txt_Lo.Text = "") Then

        Dwx32_Register.LoRange = Txt_Lo.Text

    End If


    If Not (Txt_Register_New_Value.Text = "") Then

        Dwx32_Register.Value = Txt_Register_New_Value.Text
```

```
    End If

    Get_Input_Reg_Point

    Get_Ouput_Reg_Point

End Sub
```

Get Input Register Point

```
Private Sub Get_Input_Reg_Point()

Dim b As Boolean

Dim Scan_Rate As Integer

    'get all the information you need to connect to a Point

    Node_Name = TxtNode_Name.Text

    OPCServer_Name = TxtOPC_Server_Name.Text

    Tag_Name = TxtTag_Name.Text


    If (OPCServer_Name <> "") And (Tag_Name <> "") And
(Txt_Scan_Rate.Text <> "") Then


        Scan_Rate = Txt_Scan_Rate.Text


        'Make sure what you are using a local or remote OPC-Server,

        'by checking the name of the Node

        If Node_Name = "" Then

            'then go local..

            PointName = OPCServer_Name & "\" & Tag_Name

        Else

            'else use remote..

            PointName = "\\" & Node_Name & "\" & OPCServer_Name & "\" &
Tag_Name

        End If

        b = Dwx32_Register.SetInputOPC(PointName, Scan_Rate)

    End If


End Sub
```

Get Output Register Point

```
Private Sub Get_Ouput_Reg_Point()

Dim b As Boolean


    'get all the information you need to connect to a Point
```

```
    Node_Name = Txt_Output_Node_Name.Text
    OPCServer_Name = Txt_Output_OPC_Server_Name.Text
    Tag_Name = Txt_Output_Tag_Name.Text

    If (OPCServer_Name <> "") And (Tag_Name <> "") Then

        'Make sure what you are using a local or remote OPC-Server,
        'by checking the name of the Node
        If Node_Name = "" Then
            'then go local..
            PointName = OPCServer_Name & "\" & Tag_Name
        Else
            'else use remote..
            PointName = "\\" & Node_Name & "\" & OPCServer_Name & "\" &
Tag_Name
        End If
        b = Dwx32_Register.AddOutputOPC(PointName)
    End If

End Sub
```

Registers Window

```
Private Sub cmd_Registers_Window_Click()
    CleanUp
    RegistersForm.Show
End Sub
```

Main Window

```
Private Sub cmd_Main_Window_Click()
    CleanUp
    MainForm.Show
End Sub
```

OPC Bridging Form

To start the configuration of the OPC Bridging form, it is necessary to insert a new form into your Visual Basic project. Once you have inserted the form, you are ready to begin the configuration. First insert four labels and use the Properties window to configure all of the elements so that they appear as you want them to appear.

	(Name)	Caption	Font
Label1	Label1	OPC Bridging	Arial, Bold Italic, 24pt
Label2	Label2	Using DataWorX OLE Automation	Arial, Bold Italic, 10pt
Label3	Label3		N/A
Label4	Label4		N/A

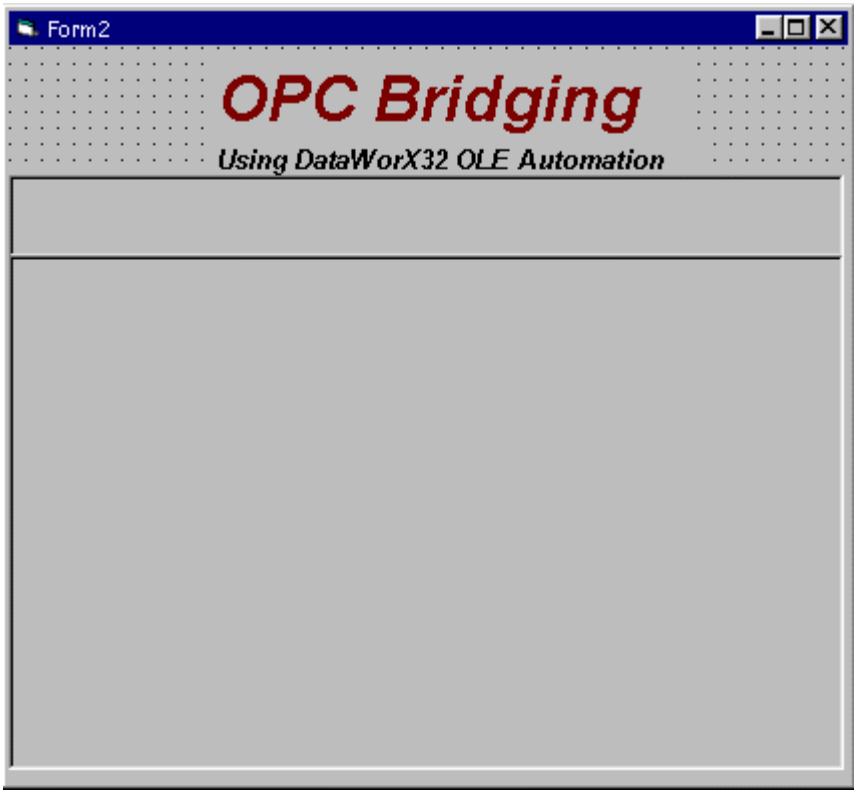


Fig. A.21.

After you have added these label areas, add two command buttons to the first blank label area with the following properties.

	(Name)	Caption
Command1	cmd_Main_Window	Main Window
Command2	cmdExit	Exit

Fig. A.22.

Now it is time to configure the second label area. Add three command buttons, 14 labels, four frames, 12 text boxes, and two combo boxes. Using the Properties window, enter the following data for each component that you just entered.

	(Name)	Caption
Frame1	CreateRegister_Frame	Add New Register/Information
Frame2	Frame2	
Frame3	Connect_toPoint_Frame	Type the Register Input Point/Tag Information
Frame4	Output_Point_Frame	Type the Register Output Point/Tag Information
Label5	Lbl_Register_Name	Register Name:
Label6	Lbl_Type	Type:
Lable7	Lbl_Writeable	Writeable:
Label8	Lbl_Group_Name	Group Name:
Lable9	Lbl_Ranges	Ranges:
Label10	Lbl_Register_New_Value	Register New Value:
Lable11	Lbl_Lo	Lo:
Label12	Lbl_Hi	Hi:
Label13	Lbl_Node_Name	Node Name:
Label14	Lbl_Server_name	Server Name:
Label15	Lbl_Tag_Name	Tag Name:
Label16	Lbl_Scan_Rate	Scan Rate:
Lable17	Lbl_Ms	Ms.
Label18	Label18	You can Add more than one Output Point for the Register. For every Point, just type the Point information and then click on the Add Output Point Button.
Command3	cmd_Add_Register	Add Register
Command4	cmd_Update	Update
Command5	cmd_Add_Output_Point	Add Output Point
Text1	Txt_Register_Name	Reg_Name
Text2	TxtGroup_Name	\

	(Name)	Caption
Text3	Txt_Lo	0
Text4	Txt_Hi	100
Text5	Txt_Register_New_Value	99
Text6	TxtNode_Name	
Text7	TxtOPC_Server_Name	SMAR.Simulator
Text8	TxtTag_Name	SimulatePLC.Random
Text9	Txt_Scan_Rate	50
Text10	Txt_Output_Node_Name	
Text11	Txt_Output_OPC_Server	SMAR.Simulator
Text12	Txt_Output_Tag_Name	SimulatePLC.OUTPUTS.INT
Combo1	Cmb_Register_Type	BOOL
Combo2	cmd_Writeable	True

After you have entered all of these elements, this label should look like the following figure:

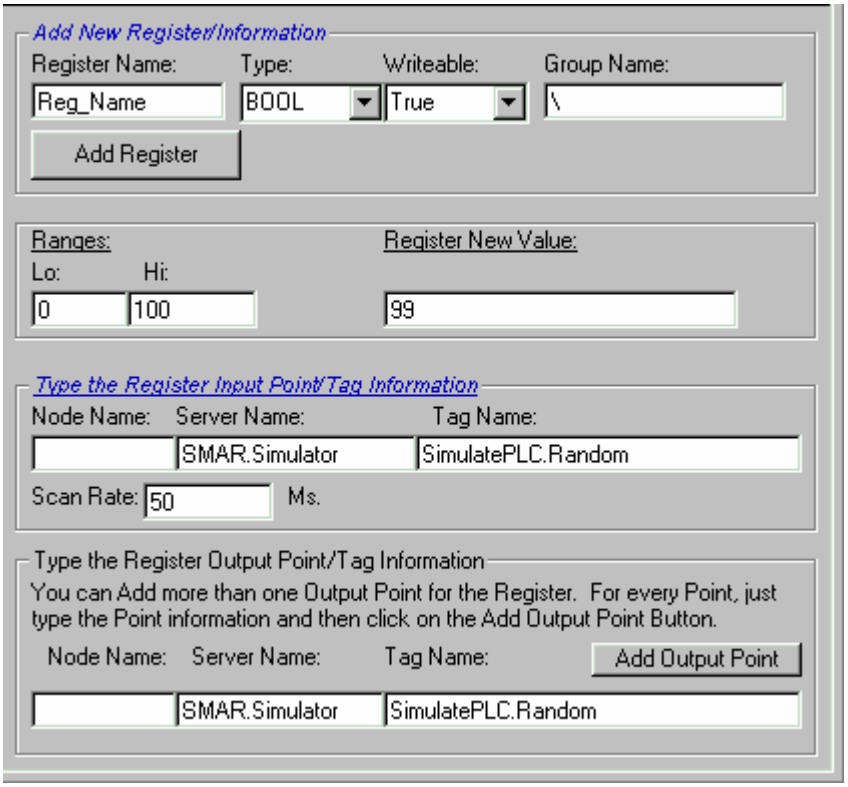


Fig. A.23.

Main Window

```
Private Sub cmd_Main_Window_Click()

    MainForm.Show

End Sub
```

Exit

```
Private Sub cmdExit_Click()

    Set Dwx32_Register = Nothing    'release the unused memory

    Unload Me

End Sub
```

Add New Register/Information

```
Private Sub CreateRegister_Frame_DragDrop(Source As Control, X As Single,
Y As Single)

End Sub
```

Register Name

```
Private Sub lbl_Register_Name_Change()

End Sub
```

Register Name Text Box

```
Private Sub Txt_Register_Name_Change()

End Sub
```

Register Type

```
Private Sub lbl_Type_Change()

End Sub
```

Register Type Combo Box

```
Private Sub Cmb_Register_Type_Change()

End Sub
```

Writeable

```
Private Sub lbl_Writeable_Change()

End Sub
```

Writeable Combo box

```
Private Sub Cmb_Writeable_Change()

End Sub
```

Group Name

```
Private Sub lbl_Group_Name_Change()

End Sub
```

Group Name text Box

```
Private Sub TxtGroup_Name_Change()

End Sub
```

Add Register

```
Private Sub cmdAdd_Register_Click()

Dim RegName, GroupName As String

Dim RegType As REGDATATYPE

Dim RegWriteable, b As Boolean

   RegName = TxtRegister_Name.Text

   RegType = Get_Reg_Type(Cmb_Register_Type)

   RegWriteable = Get_Reg_Writeable(Cmb_Writeable)

   GroupName = TxtGroup_Name.Text

   'make sure Dwx32 is Running.

    If dwx Is Nothing Then

        b = MsgBox("Make sure DataWorX is in Run-time Mode" & Chr(13) &
"Try Again..", vbOKOnly, "Dwx32 Ole")

       '''' Show_CreateRegister_support ' go back and change the
Information.

        Exit Sub           'do not do anything more exit the sub now.

    Else

        Screen.MousePointer = vbHourglass               'change the mouse
to busy..

    Set    Dwx32_Register    =    dwx.CreateRegister(RegName,    RegType,
RegWriteable, GroupName)

        Screen.MousePointer = VBDEFULT                  'change the mouse
to default..

    End If

    If (Dwx32_Register Is Nothing) Then

        b = MsgBox("Make sure you have a Valid Group Name, " & Chr(13) &
_

        "Register Type or/and Writeable Option" & Chr(13) & _

        "Or the Register Name already  exists in DataWorX.", vbOKOnly,
"Dwx32 Ole")

        Form_Load
```

```
    Else

        OPC_Bridging_Form.Height = 5490

        Exit Sub           'do not do anything more exit the sub now.

    End If

End Sub
```

Ranges

```
Private Sub lbl_Ranges_Change()

End Sub
```

Lo

```
Private Sub lbl_Lo_Change()

End Sub
```

Lo Text Box

```
Private Sub Txt_Lo_Change()

End Sub
```

Hi

```
Private Sub lbl_Hi_Change()

End Sub
```

Hi Text Box

```
Private Sub Txt_Hi_Change()

End Sub
```

Register New Value

```
Private Sub lbl_Register_New_Value_Change()

End Sub
```

Register New Value Text Box

```
Private Sub Txt_Register_New_Value_Change()

End Sub
```

Type the Register/Input Point Tag Information (Frame)

```
Private Sub Connect_toPoint_Frame_DragDrop(Source As Control, X As
Single, Y As Single)

End Sub
```

Node Name

```
Private Sub lbl_Node_Name_Change()

End Sub
```

Node Name Text Box

```
Private Sub TxtNode_Name_Change()

End Sub
```

Server Name

```
Private Sub lbl_Server_Name_Change()

End Sub
```

Server Name Text Box

```
Private Sub TxtOPC_Server_Name_Change()

End Sub
```

Tag Name

```
Private Sub lbl_Tag_Name_Change()

End Sub
```

Tag Name Text Box

```
Private Sub TxtTag_Name_Change()

End Sub
```

Scan Rate

```
Private Sub lbl_Scan_Rate_Change()

End Sub
```

Scan Rate Text Box

```
Private Sub Txt_Scan_Rate_Change()

End Sub
```

Update

```
Private Sub cmd_Update_Click()

    On Error GoTo Error_handler

    'make sure Dwx32 is Running.

    If dwx Is Nothing Then

        b = MsgBox("Make sure DataWorX is in Run-time Mode" & Chr(13) &
"Try Again..", vbOKOnly, "Dwx32 Ole")

        Form_Load      ' go back and change the Information.

        Exit Sub       'do not do anything more exit the sub now.

    Else

        Screen.MousePointer = vbHourglass               'change the mouse
to busy..

        Set Dwx32_Register = dwx.GetRegister(TxtRegister_Name.Text)
```

```
        Screen.MousePointer = VBDEFULT                  'change the mouse
to default..

    End If


    'make sure the Register dose exist.

    If Dwx32_Register Is Nothing Then

        b = MsgBox("Make sure you have a Valid Register Name.." & Chr(13)
& _

        "NOTE: Name is a Case Sensitive.", vbOKOnly, "Dwx32 Ole")

        Form_Load          ' go back and change the Information.

        Exit Sub           'do not do anything more exit the sub now.

    End If


    If Not (Txt_Hi.Text = "") Then

        Dwx32_Register.HiRange = Txt_Hi.Text

    End If


    If Not (Txt_Lo.Text = "") Then

        Dwx32_Register.LoRange = Txt_Lo.Text

    End If


    If Not (Txt_Register_New_Value.Text = "") Then

        Dwx32_Register.Value = Txt_Register_New_Value.Text

    End If


    Get_Input_Reg_Point


    OPC_Bridging_Form.Height = 7155

    Exit Sub


Error_handler:

    MsgBox (Err.Number & Err.Description)

End Sub
```

Type the Register/Output Point Tag Information (Frame)

```
Private Sub Output_Point_Frame_DragDrop(Source As Control, X As Single, Y
As Single)

End Sub
```

Node Name

```
Private Sub Text7_Change()

End Sub
```

Node Name Text Box

```
Private Sub Txt_Output_Node_Name_Change()

End Sub
```

Server Name

```
Private Sub Text8_Change()

End Sub
```

Server Name Text Box

```
Private Sub Txt_Output_OPC_Server_Change()

End Sub
```

Tag Name

```
Private Sub Text9_Change()

End Sub
```

Tag Name Text Box

```
Private Sub Txt_Output_Tag_Name_Change()

End Sub
```

Add Output Point

```
Private Sub cmd_Add_Output_Point_Click()

    Get_Ouput_Reg_Point

End Sub
```

Registers Multiply Form

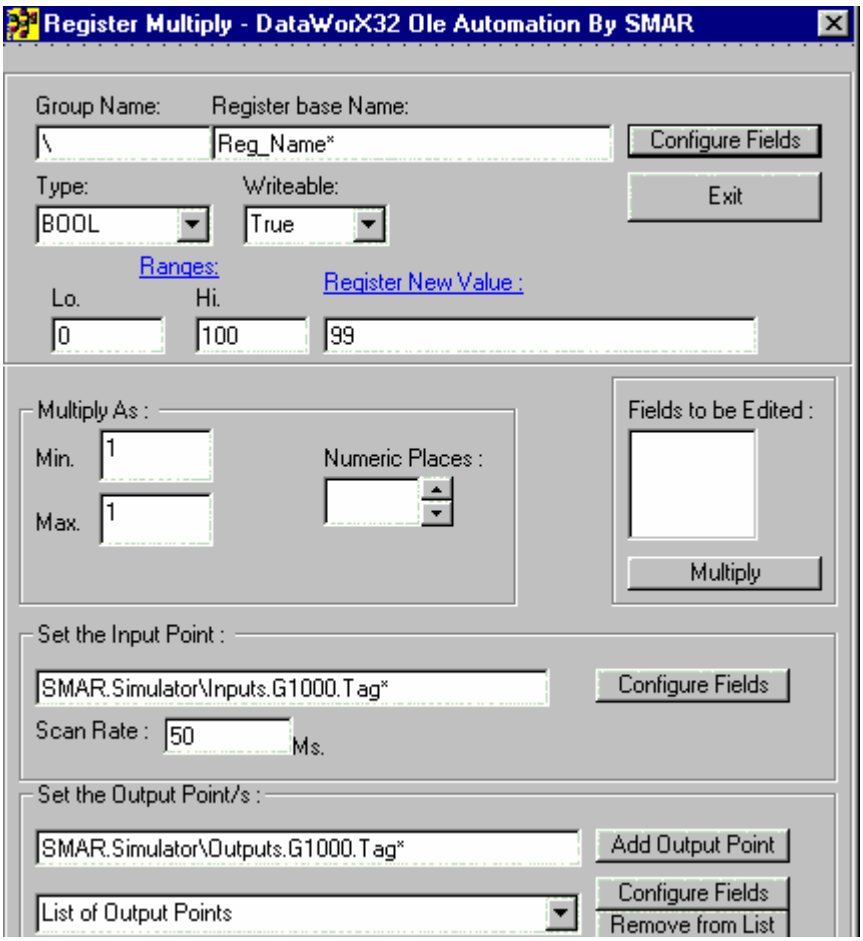


Fig. A.24. Register Multiply Form

Group Name

```
Private Sub lbl_Group_Name_Change()

End Sub
```

Group Name Text Box

```
Private Sub TxtGroup_Name_Change()

End Sub
```

Register Name

```
Private Sub lbl_Register_Name_Change()

End Sub
```

Register Name Text

```
Private Sub Txt_Reg_Name_Change()

    Multiply_As_Frame.Enabled = True

End Sub
```

Configure Fields

```
Private Sub cmd_Configure_Fields_Register_Click()

    Multiply_As_Frame.Enabled = True

    Min = txtmin.Text

    Max = txtmax.Text

    Num_Places = txtnumplace.Text


        IO = "Create_Registers"

        lstfields.Clear        'Clear any old data in the list box

        Base_Reg_Name_Text = Txt_Reg_Name 'Remember base text before it
is modified

        Setup_Fields (Base_Reg_Name_Text)

        cmd_Multiply.Visible = True

        cmd_Multiply.Caption = "Multiply Register"


        Multiply_As_Frame.Enabled = False


End Sub
```

Type

```
Private Sub lbl_Type_Change()

End Sub
```

Type Combo Box

```
Private Sub Cmb_Register_Type_Change()

End Sub
```

Writeable

```
Private Sub lbl_Writeable_Change()

End Sub
```

Writeable Combo Box

```
Private Sub Cmb_Writeable_Change()

End Sub
```

Ranges

```
Private Sub lbl_Ranges_Change()

End Sub
```

Lo

```
Private Sub lbl_Lo_Change()

End Sub
```

Lo Text Box

```
Private Sub Txt_Lo_Change()

End Sub
```

Hi

```
Private Sub lbl_Hi_Change()

End Sub
```

Hi Text Box

```
Private Sub Txt_Lo_Change()

End Sub
```

Register New Value

```
Private Sub lbl_Register_Value_Change()

End Sub
```

Register New Value Text Box

```
Private Sub Txt_Register_New_Value_Change()

End Sub
```

Multiply as Frame

```
Private Sub Multiply_As_Frame_DragDrop(Source As Control, X As Single, Y
As Single)

End Sub
```

Minimum

```
Private Sub Text7_Change()

End Sub
```

Min. Text Box

```
Private Sub txtmin_Change()

End Sub
```

Maximum

```
Private Sub Text8_Change()

End Sub
```

Max. Text Box

```
Private Sub txtmax_Change()

End Sub
```

Numeric Places

```
Private Sub Text10_Change()

End Sub
```

Numeric Place Scroll Box

```
Private Sub txtnumplace_Change()

End Sub
```

Fields to be Edited

```
Private Sub Frame4_DragDrop(Source As Control, X As Single, Y As Single)

End Sub
```

Fields to be Edited Text Box

```
Private Sub lstfields_Click()

End Sub
```

Multiply

```
Private Sub cmd_Multiply_Click()

    Select Case IO

    Case "In_put"

            Multiply_Input_Point

    Case "Out_put"

            Multiply_Output_Point

    Case "Create_Registers"

            Create_Registers

    End Select

End Sub
```

Set the Input Point Frame

```
Private Sub Frame3_DragDrop(Source As Control, X As Single, Y As Single)

End Sub
```

Set the Input Point Text Box

```
Private Sub Txt_base_Input_Point_Change()

End Sub
```

Scan Rate

```
Private Sub lbl_Scan_Rat_Change()

End Sub
```

Scan Rate Text Box

```
Private Sub Txt_Scan_Rate_Change()

End Sub
```

Set the Output Point/s Frame

```
Private Sub Frame2_DragDrop(Source As Control, X As Single, Y As Single)

End Sub
```

Set the Output Point Text Box

```
Private Sub Txt_base_Output_Point_Change()

End Sub
```

Set the Output Point Combo box

```
Private Sub Cmb_Output_Points_Change()

End Sub
```

Add Output Point

```
Private Sub cmd_Output_Points_Click()

End Sub
```

Configure Fields

```
Private Sub cmd_Configure_Output_Fields_Click()

  IO = "Out_put"

  lstfields.Clear 'Clear any old data in the list box

  Base_Reg_Output_Text = Cmb_Output_Points 'Remember base text before it
is modified

  Setup_Fields (Base_Reg_Output_Text)

  cmd_Multiply.Visible = True

  cmd_Multiply.Caption = "Multiply Outputs"

End Sub
```

Remove from List

```
Private Sub cmd_Remove_from_List_Click()

    For i = Cmb_Output_Points.ListCount - 1 To 0 Step -1 'find i index
number, then remove from the list box

        If Cmb_Output_Points = Cmb_Output_Points.List(i) Then

               Cmb_Output_Points.RemoveItem (i)

        End If

        Next

End Sub
```