



Your comments and suggestions on the operation of this software are welcome. Please address them to:

ICONICS

100 Foxborough Blvd.

Foxborough, MA 02035

Tel: 508-543-8600

Fax: 508-543-1503

E-mail: support@iconics.com

Web: www.iconics.com

© ICONICS, Inc. All rights reserved.

Contents

Introduction

Using AlarmWorX32 Viewer ActiveX

Connections

Security

Language-Aliasing Support

Global-Aliasing Support

Asynchronous Downloading Features

User Interface

Runtime Operations

Introduction

The AlarmWorX32 Current Events Viewer is an OPC ActiveX client application that provides real-time alarm information within the GENESIS32 product family. It communicates to OPC Alarm and Events servers to provide excellent alarming features. It is important to note that this is a **Current Events** Viewer, which means that it only displays active alarms and current operator messages.

The structure of the AlarmWorX32 Current Events Viewer is built on a three-level system, consisting of **Default** settings (color, font, justification, wrapping method and size), **Row** types (color and font) and **Column** values (color, font, justification, wrapping method and size). Each system starts with certain default settings, which can be overwritten by row settings, which in turn can be overwritten by column settings.

Using the AlarmWorX32 Viewer ActiveX

The AlarmWorX32 Viewer ActiveX integrates easily within Microsoft Visual C++ or Microsoft Visual Basic (VB) container applications. The ICONICS GENESIS32 containers, such as GraphWorX32, provide toolbar support for inserting the AlarmWorX32 Viewer ActiveX. Once inserted into a document or a form, the component can be resized and configured.

Within the ICONICS GENESIS32 containers and in Visual Basic forms, the component is initially placed in **Configuration** mode, where its user interface is disabled and you can configure the AlarmWorX32 Viewer ActiveX through the **Properties** dialog box. Once the application is committed to **Runtime** mode, the viewer will automatically connect to alarm and event OPC servers, query for a complete update of all requested alarms and display current alarm messages. The AlarmWorX32 Viewer ActiveX has the option of saving file information as part of the container or in its own file format.

You must give a valid filename, if saving to a file, before the data are serialized and saved. The AlarmWorX32 Viewer uses the concept of a **Working Directory**, defined in the ICONICS GENESIS32 containers. This represents the working directory where project files and configuration files will be stored. The AlarmWorX32 Viewer will utilize this directory as a starting point for saving and loading its configuration files.

Connections

The AlarmWorX32 Viewer can display data from any OPC Alarm and Event server. Alarm subscription filters are made through the **Subscription** tab of the **Properties** dialog box.

Once a subscription is set up, the viewer requests a refresh of all current alarms upon animation. Alarms are sent by the OPC Alarm and Event server on an alarm-by-alarm basis. This decreases the number of alarms sent to each viewer and allows for server-side filtering.

Security

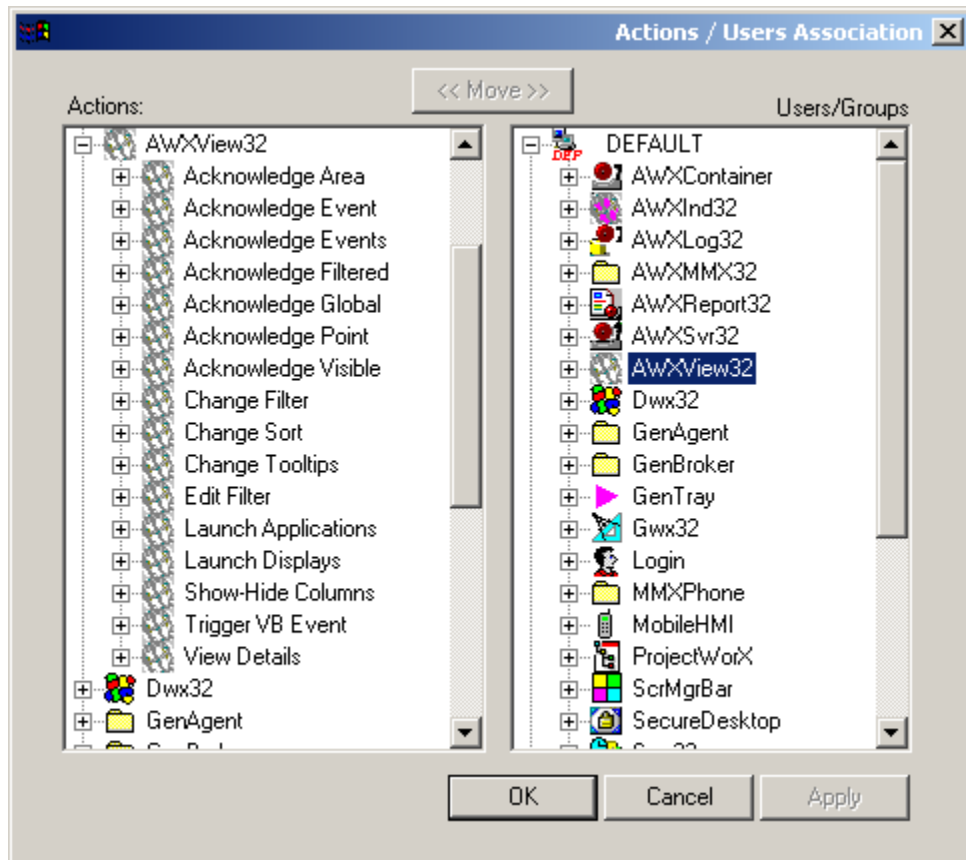
To access the ICONICS Security Server from the AlarmWorX32 Container, either select **Security > Configuration** from the **Tools** menu or click the **Security** button on the toolbar. This opens the **Security Server Administrator Login** dialog box, as shown in the figure below.

The image shows a Windows-style dialog box titled "ICONICS Security Server Administrator Login". It has a blue title bar with a close button (X) on the right. The main area is light gray. On the left, there is a small icon of a padlock. To the right of the icon, the text reads: "Enter an Administrator user name and password, or leave the user name blank and enter the default Administrator password:". Below this text are three input fields. The first is labeled "User Name:" and is empty. The second is labeled "Password:" and is empty. The third is labeled "Challenge:" and contains the text "18303". At the bottom of the dialog box, there are two buttons: "OK" and "Cancel".

Security Server Login

Enter your **User Name** and **Password**, and then click **OK**. This opens the ICONICS Security Server, where you can access the **AWXView32** Security Server. Depending on your configured level of security, you may or may not be able to edit components of the **AWXView32** Security Server. Choose **Application Actions from the Edit menu**. The **Actions/Users Association** dialog box appears, as shown in the figure below.

Note: For more information about security, please see the Security Configurator help documentation.



AlarmWorX32 Viewer ActiveX Actions Protected by Security

The Security Server can be set up to allow access to several different actions from this dialog. To set the options for the AlarmWorX32 Viewer, select **AWXView32** from the list, as shown in the figure above. This opens the list in the left pane, which includes the following options:

- Acknowledge Area
- Acknowledge Event
- Acknowledge Events
- Acknowledge Filtered
- Acknowledge Global
- Acknowledge Point
- Acknowledge Visible
- Change Filter
- Change Language
- Change Sort
- Change ToolTips
- Edit Filter
- Configure Property Pages
- Launch Applications
- Launch Displays
- Show - Hide Columns
- Trigger VB Event
- View Details

Language-Aliasing Support

The AlarmWorX32 Viewer ActiveX component has all of its strings saved in the resource file, which can be modified to provide custom versions. In addition, it provides real-time support for loading a resource-only .dll to support an international language. This .dll file is compiled as a standard Win32 .dll, and it is made up of all the resources and the .odl file of the AlarmWorX32 Viewer module. Any time the ICONICS applications settings are switched to a different language, AlarmWorX32 Viewer will load the corresponding resource .dll.

The resource-only .dll file name uses the AlarmWorX32 Viewer component, followed by the three characters used in Microsoft Windows NT as the abbreviation for a language name (e.g., the abbreviation "deu" stands for German, and the .dll should be named "AWXView32_deu.dll"). In addition, using the OLE Automation property language ID, you can control this functionality through OLE Automation.

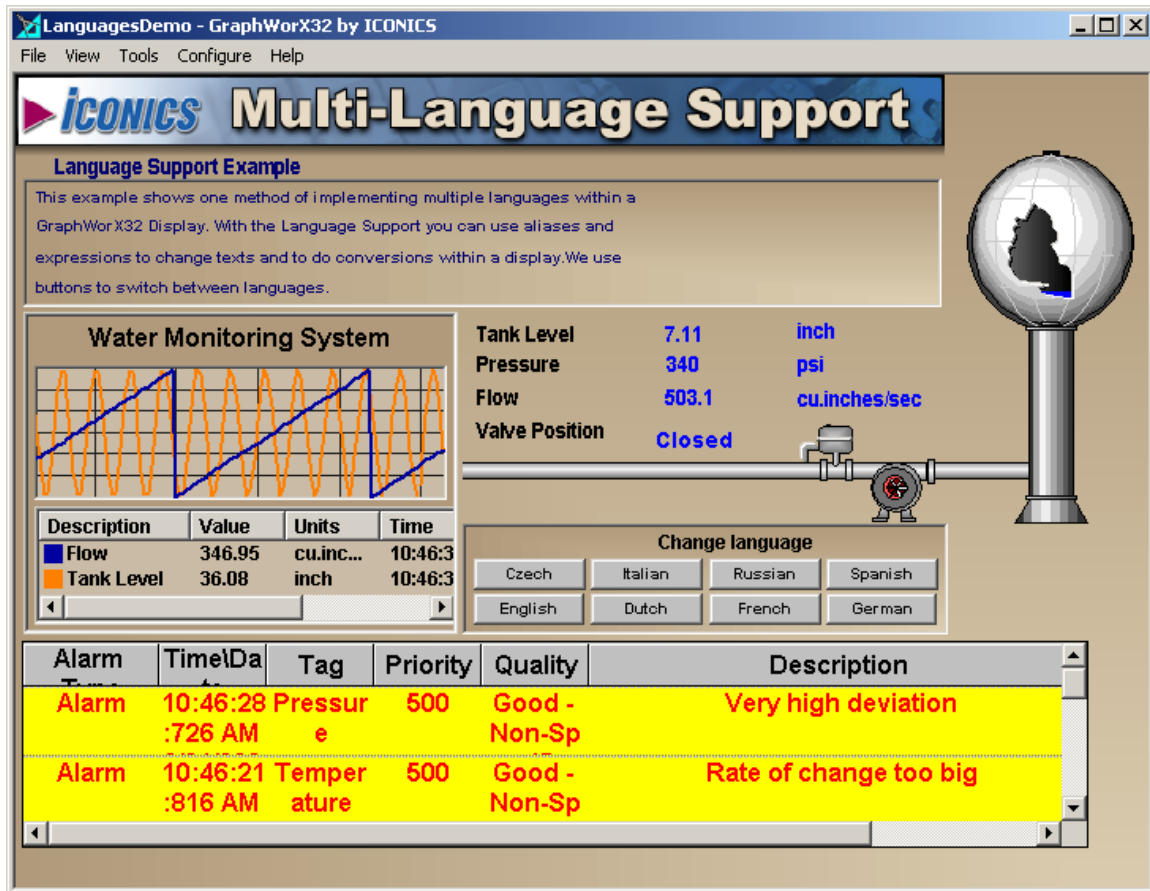
The Unicode version of the AlarmWorX32 Viewer ActiveX also supports language aliasing with automatic range and value scaling, as well as unit conversions, formatting, and font selection. For more information about configuring Unicode version language-aliasing support, refer to the GENESIS32 **Language Configurator** documentation.

Note: The AlarmWorX32 Viewer requires the following additional language-aliasing configuration work. If any expression is to be used for range, alarm limits, and or unit scaling, you need to define both a "Read Expression" and a "Write Expression" in the Language Configurator. By doing so, the Viewer will be able to function properly between language switches.

Note: To enhance language-aliasing performance, select the Microsoft Arial Unicode font, which contains all Unicode characters. The Arial Unicode MS font must be selected separately within the **Properties** dialog boxes of each ActiveX component.

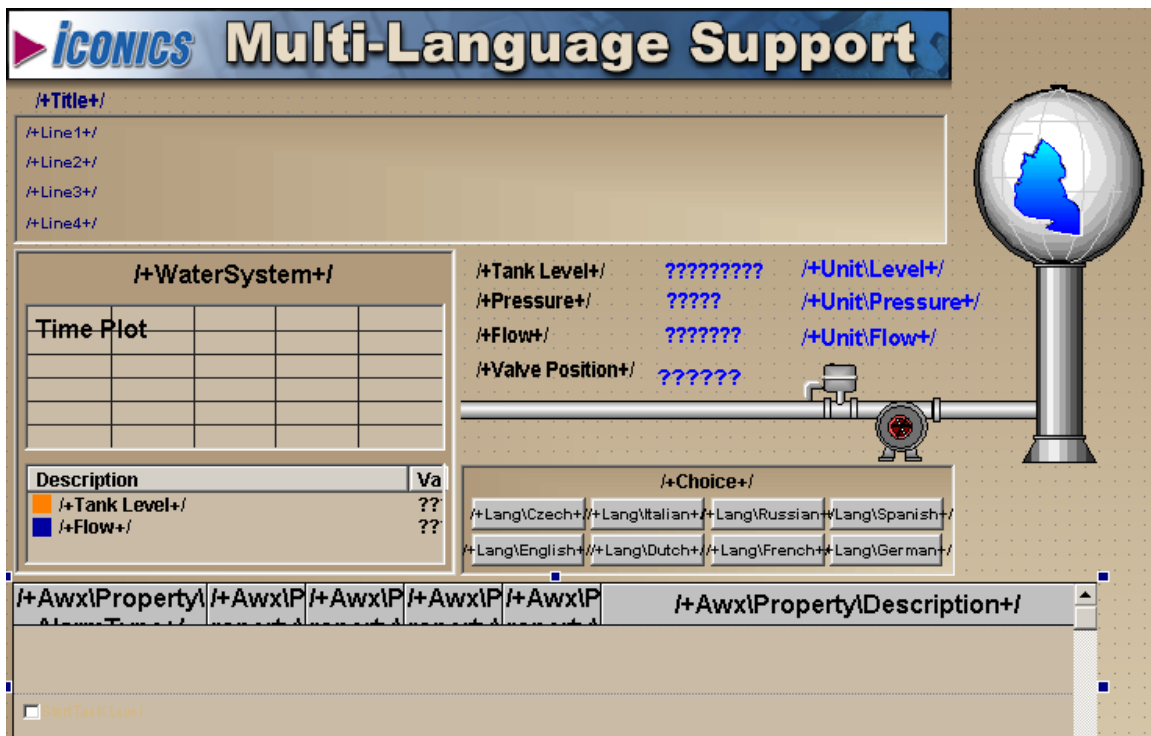
Sample Language Configuration

The Unicode Installation of GENESIS32 provides a sample Language Server configuration under the "Languages" folder of the GENESIS32 product installation "tree." In addition, a sample display "languagesDemo.gdf," which includes a sample Viewer configuration, is provided under the GEN32DEMO folder. The figure below shows this sample during runtime mode. Notice that you can switch between languages in this display by simply clicking the buttons in the Change Language field.



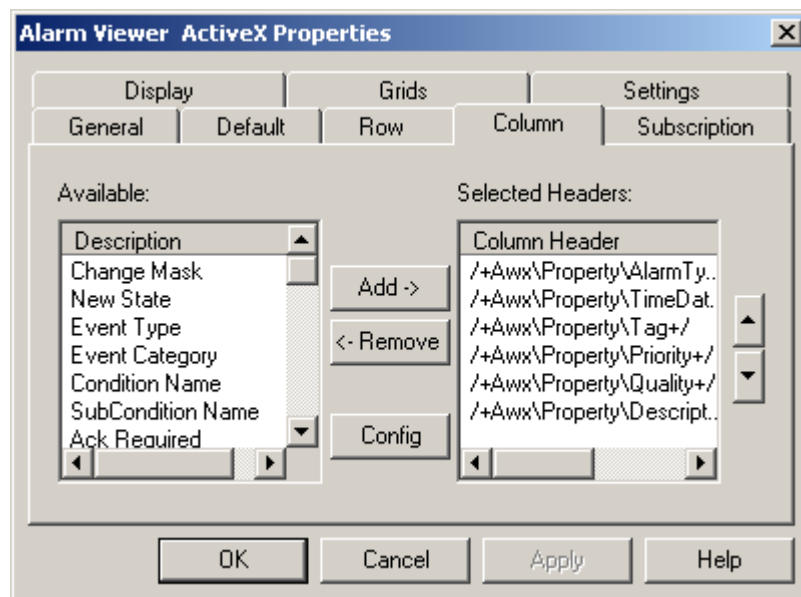
Language-Aliasing Demo in Runtime Mode

The figure below shows this same language-aliasing demo in configuration mode. This display incorporates ActiveX objects from GraphWorX32, TrendWorX32, and AlarmWorX32.



Language-Aliasing Demo in Configuration Mode

For example, if you click on the Alarm Viewer ActiveX in the language-aliasing configuration shown above, you can view the properties of the Alarm Viewer Active X, as shown in the figure below. The **Column** tab of the **Alarm Viewer ActiveX Properties** dialog box, shown below, displays the language-aliasing configuration of the headers in the Alarm Viewer.



Alarm Viewer ActiveX Properties: Column Tab

For the example above, the Alarm Viewer configuration appears as shown in the figure below. Note that the strings within a "/" and "+" delimiter pair define a language-aliased string. As you can see, each header in the Alarm Viewer configuration is a language-aliased string that contains these delimiters.

/+AwxlProperty/AlarmType	/+AwxlProperty/TimeDate	/+AwxlProperty/Tag	/+AwxlP	/+AwxlPrope

Language-Aliased Alarm Viewer Display in Configuration Mode

During animation (runtime) mode, the Viewer ActiveX will interface to the Language Server, and it will try to resolve the language-aliased strings. If successful, it will display the translated strings in the Alarm Viewer display, as shown below. In addition, if the **Units** language alias string has an equation defined in the Language Configurator, it will autoscale the values based on the scaling equation supplied by the Language Server.

AwxlProperty/AlarmType	AwxlProperty/TimeDate	AwxlProperty/Tag	AwxlPro
Alarm	08:59:54:923 19.03.2001	Pressure	500
Alarm	09:00:00:251 19.03.2001	Temperature	500

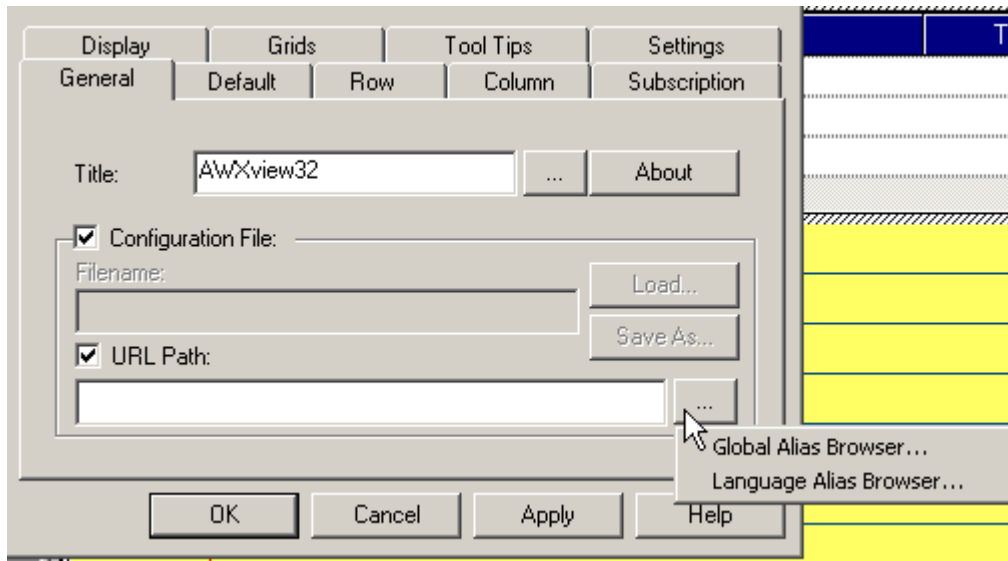
Language-Aliased Alarm Viewer Display in Runtime Mode

Note: The Unicode version language aliasing is independent of the resource .dll selection, which relates mostly to the text in the dialog boxes and other user interface elements.

Language Alias Browser

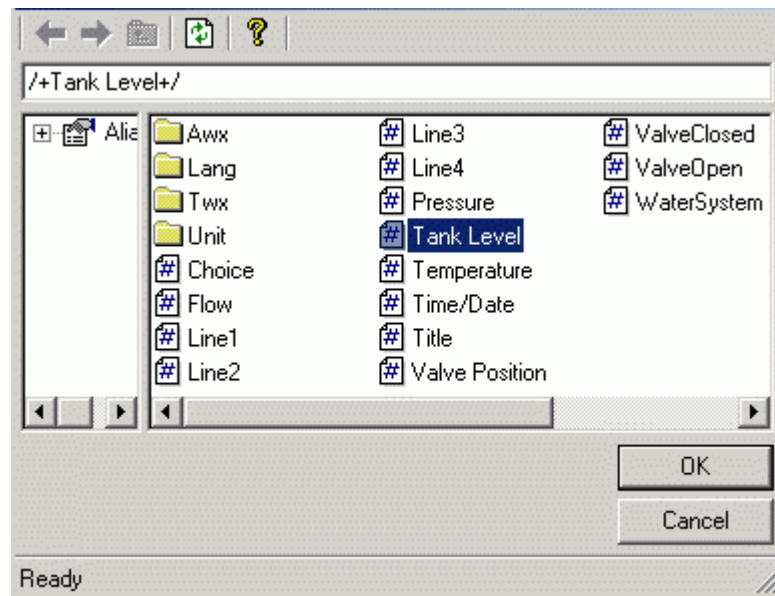
When specifying a language alias in the **Alarm Viewer ActiveX Properties** dialog box, you can also select a language alias from the **Language Alias Browser**, which includes all language aliases in the language database. This eliminates the need to manually type in the alias name. Clicking the ... button and selecting **Language Alias Browser** from the pop-up menu opens the Language Alias Browser, as shown in the figure below. The Language Alias Browser is available for the following:

- **Configuration file** and **URL Path** sections of the **Alarm Viewer ActiveX Properties** dialog box **General** tab
- Alarm filtering via the Expression Editor
- **Event Subscriptions** dialog box: **Server**, **Areas**, and **Sources**



Opening the Language Alias Browser in the Viewer Properties

The browser, shown in the figure below, includes all languages aliases in the language database. All language aliases that are configured in the Language Configurator are conveniently available to choose from inside the browser. The tree control of the Language Configurator is mimicked in the tree control of the Language Alias Browser. Select a language alias by double-clicking the alias name. The alias name appears at the top of the browser, which automatically adds the /+ and +/ delimiters to the alias name. Click the **OK** button.



Selecting an Alias From the Language Alias Browser

Global-Aliasing Support

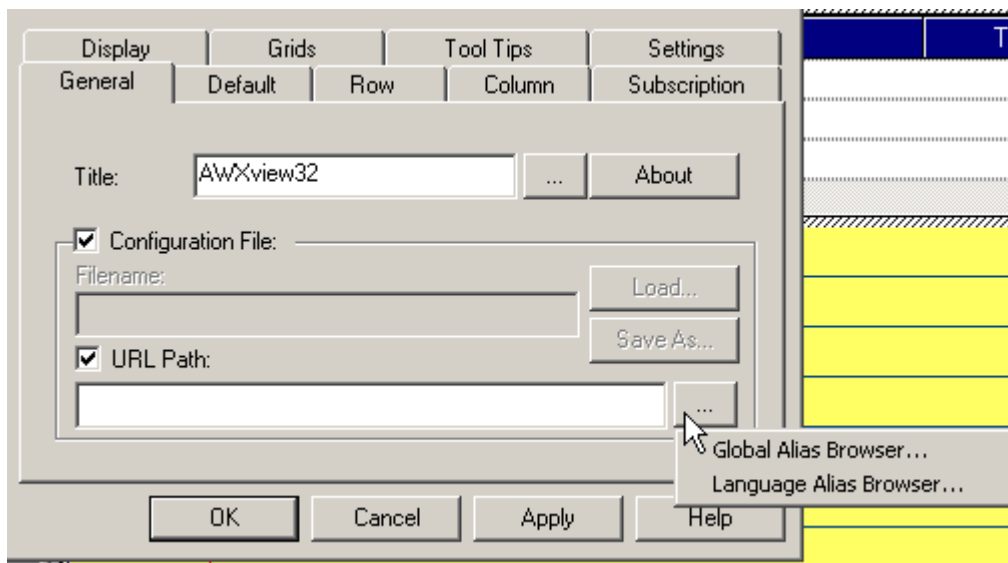
The AlarmWorX32 Viewer ActiveX supports Global Aliasing by interfacing to the ICONICS Global Aliasing System (GAS) Engine Server.

Currently the AlarmWorX32 Viewer supports machine-level and process-level aliasing. All fields that support aliasing have been interfaced to the Global Alias Browser dialog, which is evoked by clicking on the ... button in the configuration.

In addition, the AlarmWorX32 Viewer now has an integrated Global Alias Browser.

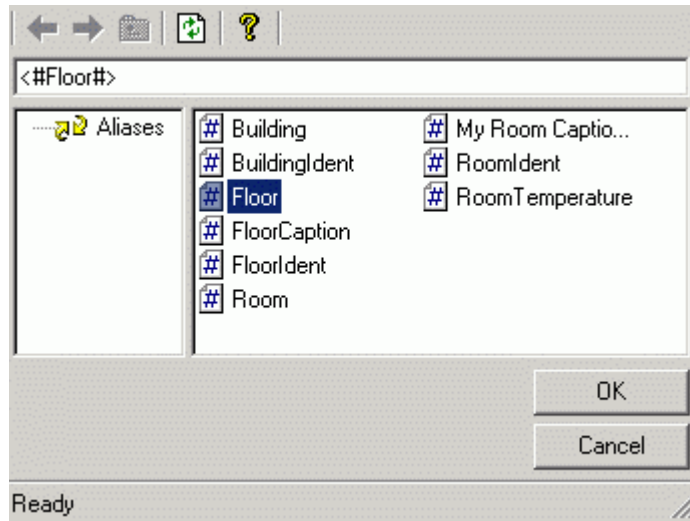
When specifying a global alias in the **Alarm Viewer ActiveX Properties** dialog box, you can also select a global alias from the **Global Alias Browser**, which includes all global aliases in the Global Aliasing System database. This eliminates the need to manually type in the alias name. Clicking the ... button and selecting **Global Alias Browser** from the pop-up menu opens the Global Alias Browser, as shown in the figure below. The Global Alias Browser is available for the following:

- **Title, Configuration file** and **URL Path** sections of the **Alarm Viewer ActiveX Properties** dialog box **General** tab
- Alarm filtering via the Expression Editor
- **Event Subscriptions** dialog box: **Server, Areas, and Sources**



Opening the Global Alias Browser in the Viewer Properties

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.

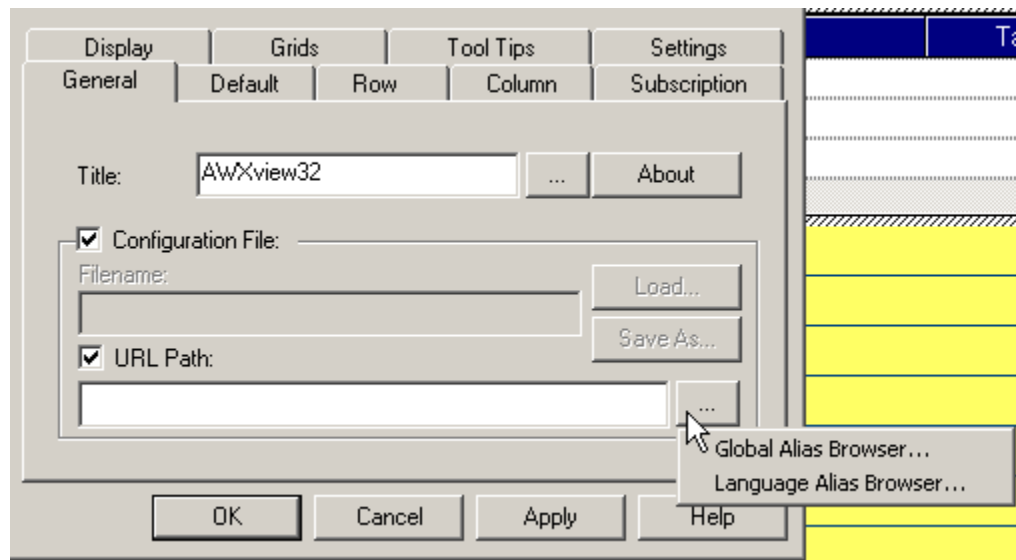


Selecting an Alias From the Global Alias Browser

Asynchronous Downloading features

The AlarmWorX32 Viewer supports asynchronous downloading of configuration data when used inside ICONICS ActiveX document servers, such as GraphWorX32, to produce documents, which can be viewed using Microsoft Internet Explorer.

To enable this feature, select **URL Path** from the **General** tab of the **Alarm Viewer ActiveX Properties** dialog box, shown below, and supply a URL path. In the case of viewing ActiveX documents within an Intranet setup, you can supply a server name and directory specifying the location of the AlarmWorX32 Viewer configuration data (must be a shared directory), by using the convention "file:\\Server Name\\Directory\\File Name.awv."



Alarm Viewer ActiveX Properties: General Tab

User Interface

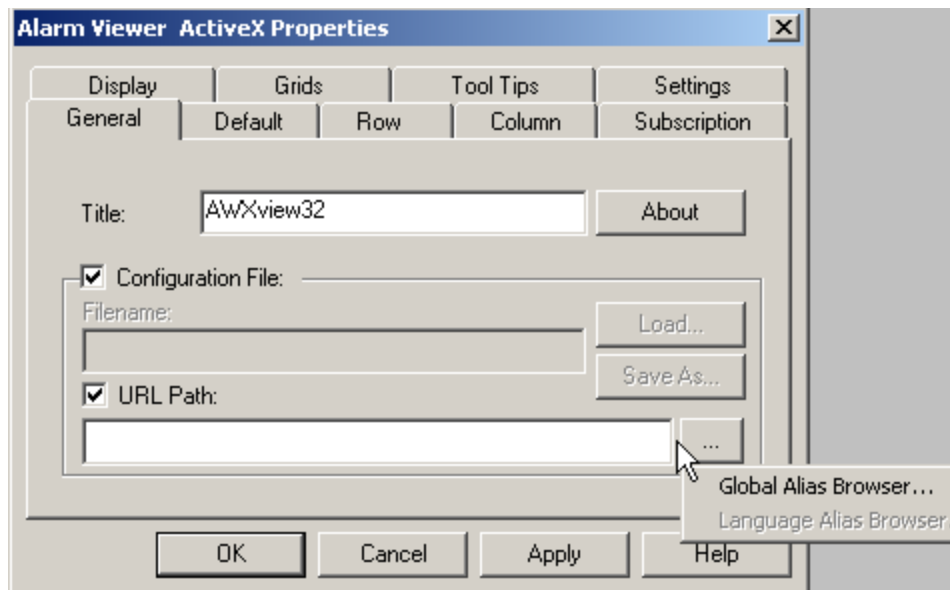
This section describes the user interface for the AlarmWorX32 Viewer ActiveX.

Properties Dialog Client Side Filtering

Properties Dialog

The **Alarm Viewer ActiveX Properties** dialog box, shown below, is the user configuration interface for the ActiveX. When embedded in an ActiveX container, double-clicking anywhere in the ActiveX client opens the **Properties** dialog box.

Note: Right-clicking on the Viewer ActiveX and selecting Properties also opens the **Alarm Viewer ActiveX Properties** dialog box.



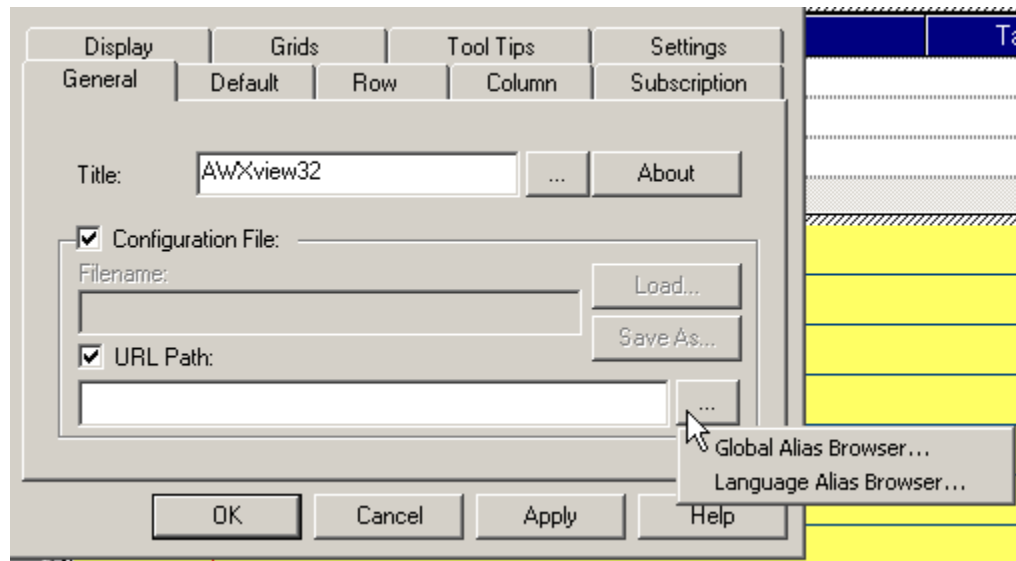
Alarm Viewer ActiveX Properties Dialog Box

General Default

Row
Column
Display
Subscription

General

The **General** tab of the **Alarm Viewer ActiveX Properties** dialog box, shown below, allows you to specify the name of the viewer file. By default, information is stored as part of the container application's document file.



Alarm Viewer ActiveX Properties: General Tab

Configure the following **General** tab settings:

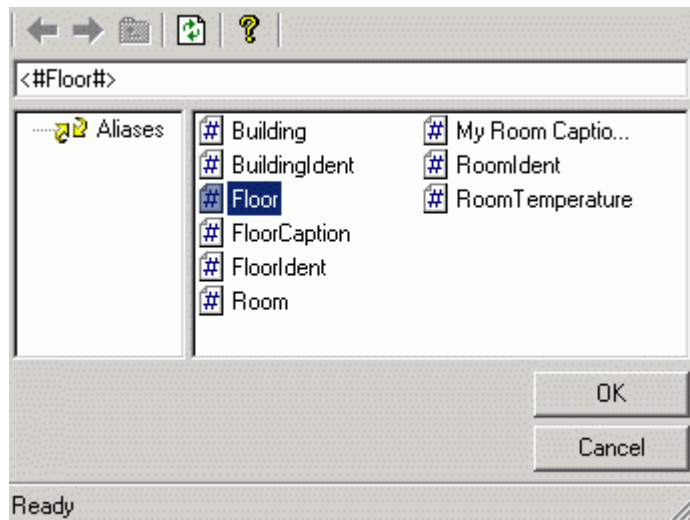
- **Title:** Enter a title for the viewer.
- **About:** Clicking the **About** button displays both technical support information and the ICONICS About Box, which contains registration and serial number information as well as system resource information, such as memory and disk space available.
- **Configuration File:** By default, information is stored as part of the container application's document file. You can also store information as part of a separate (.awv) file. To save the .awv file, click **Save As** to open the **Save AWXView32 File** dialog box. Enter a name for the .awv file in the **File Name** field, and then click **Save**.

You can also open an existing configuration file by clicking the **Load** button on the **General** tab and browsing for the file. If **Configuration File** is selected, the Viewer ActiveX will attempt to load from the file listed under **File name**. If **URL Path** is checked, the ActiveX will use

the specified network URL path upon loading the file.

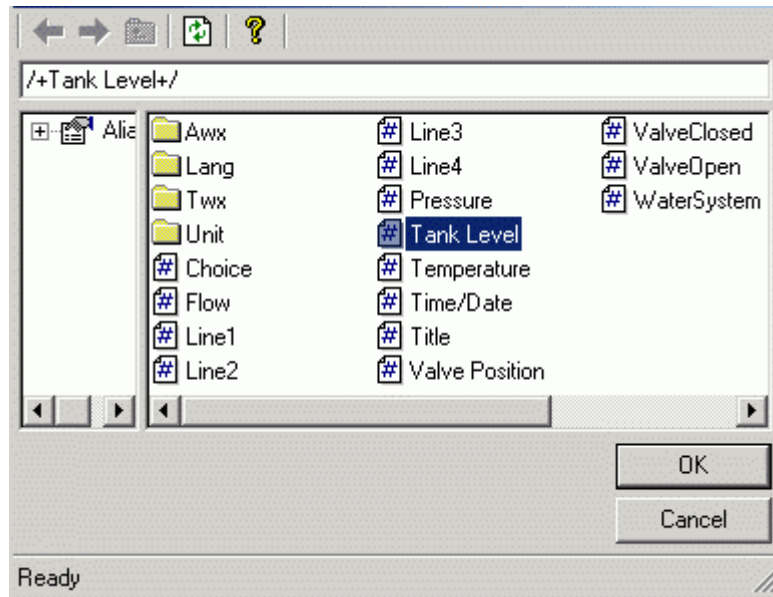
Note: The ActiveX cannot be saved to a URL path. To create a URL file, save a report configuration to a local file and copy the file to the desired network location.

You can also select aliases to use for the configuration file. Clicking the ... button and selecting **Global Alias Browser** from the pop-up menu opens the Global Alias Browser, as 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.



Selecting an Alias From the Global Alias Browser

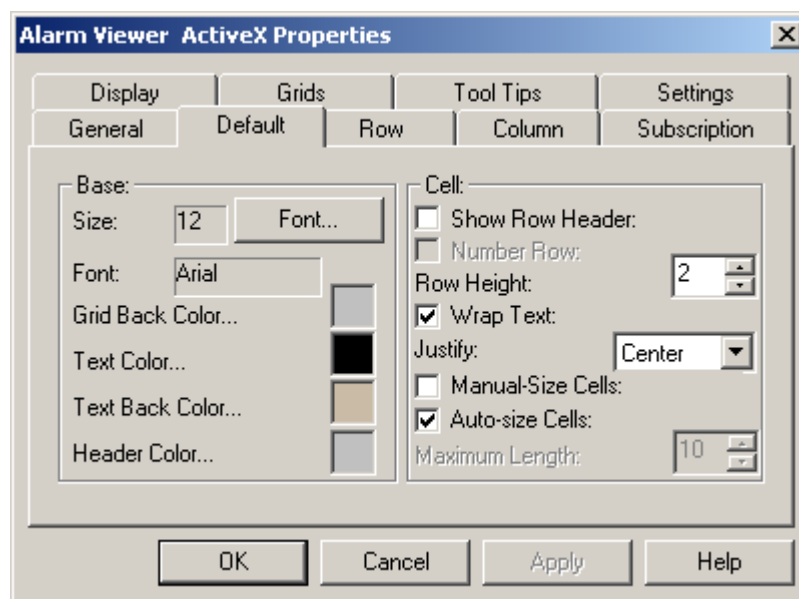
Clicking the ... button and selecting **Language Alias Browser** from the pop-up menu opens the Language Alias Browser, as shown in the figure below. The browser includes all languages aliases in the language database. All language aliases that are configured in the Language Configurator are conveniently available to choose from inside the browser. The tree control of the Language Configurator is mimicked in the tree control of the Language Alias Browser. Select a language alias by double-clicking the alias name. The alias name appears at the top of the browser, which automatically adds the /+ and +/ delimiters to the alias name. Click the **OK** button.



Selecting an Alias From the Language Alias Browser

Default

The **Default** tab of the **Alarm Viewer ActiveX Properties** dialog box, shown below, configures the appearance of the AlarmWorX32 Viewer ActiveX.



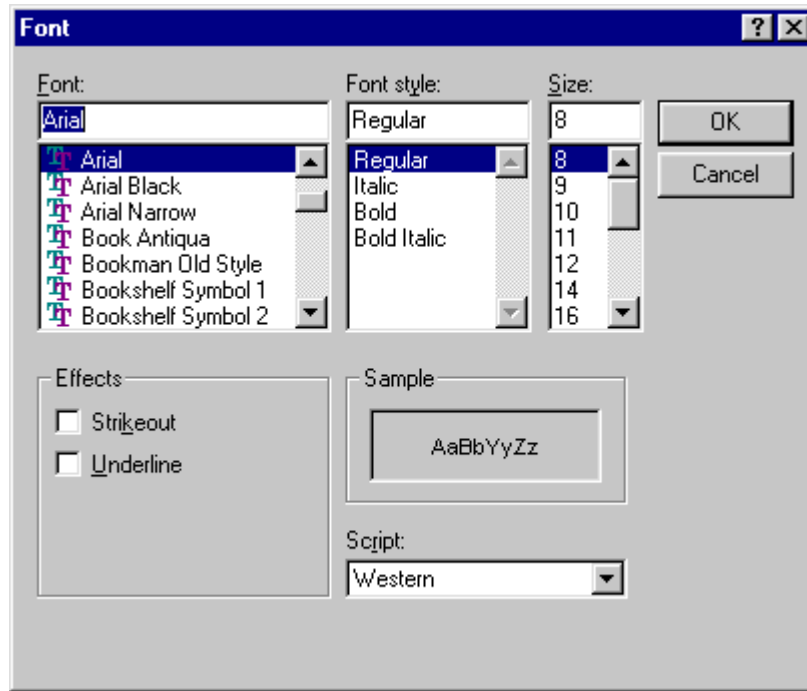
Alarm Viewer ActiveX Properties: Default Tab

Color, Font and Style

The colors section allows you to choose colors for the background of the display, the background color of the text, the background color of the header and the base foreground text color. Under the **Base** field, select a **Grid Back Color**, **Text Color**, **Text Back Color**, and **Header Color** by clicking the color

palette. A **Color** box appears, allowing you to choose a preconfigured color or to create your own color using the **Define Custom Colors** button.

Click the **Font** button to specify the font size, style, and text effects (i.e. underline and strikethrough) for the alarm viewer.



Selecting a Font Style for the Viewer

Cell

The **Cell** section sets the default options to apply to the cells of the viewer. Checking **Show Row Header** causes a row header to be displayed in the viewer. Checking **Show Row Header** also enables the **Number Row** option, but it is not necessary to have both of these options checked. Checking **Number Row** causes the row header to number each row line. If the **Number Row** option is not checked, then the row headers will be blank. The **Row Height** field sets the height of the rows in the viewer.

Note: The **Row Height** must be a positive integer between 1 and 10.

Checking **Wrap Text** box makes the text wrap when it hits the end of the particular cell. When this option is not applied, text will continue in one line although only part of the text will be visible, depending on the size of the cell. Please note that the **Wrap Text** feature is available only with multi-line rows. The **Justify** field determines the justification (left, center or right) of the text within each cell.

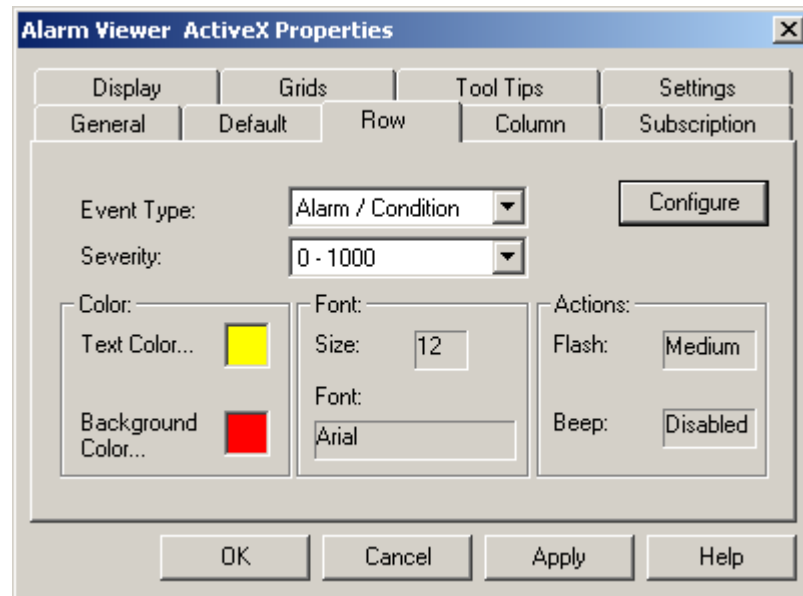
There are three options for determining the size of each cell:

- **Manual-Size Cells:** Checking this box allows you to manually set the width of each column. Manual sizing of columns can only be done during runtime.

- **Auto-Size Cells:** When this box is checked, the viewer automatically sets the width of each column. The viewer automatically sets each column to the width nearest to the size of the ActiveX divided by the number of visible columns.
- **Maximum Length for Each Cell:** Currently this is a range from 1 to 1000, and it applies the maximum cell length to all cells.

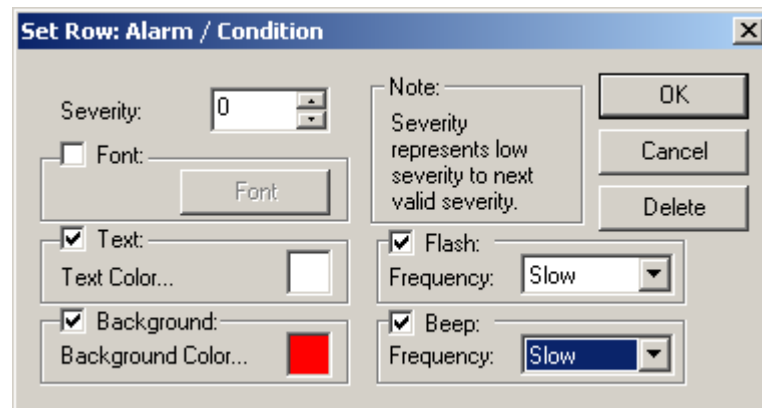
Row

The **Row** tab of the **Alarm Viewer ActiveX Properties** dialog box, shown below, sets the color of the text and background for different event types.



Alarm Viewer ActiveX Properties: Row Tab

Use the **Event Type** drop-down list to select the event type you wish to configure. It is possible to have several different configurations for one event type. To configure an event type, simply click the **Configure** button or click either the **Text Color** or **Background Color** box. This opens the **Set Row** dialog box, as shown in the figure below.



Setting Row Properties for the Alarm/Condition

The **Severity** field sets the lower bound for this particular configuration. This lower bound represents the point at which the viewer will begin to override the current values for text color, background color, and font with the values set in this dialog.

Note: The new height will be resized based on the size of the font selected.

Through this method, it is possible to set up ranges. For example, configuring a severity of 300 with a text color of purple and another severity of 600 with a text color of green would result in a purple range from 300 to 599.

This dialog can also be used to delete a configuration. To delete a configuration, select the lower bound severity of the configuration you wish to delete and click the **Delete** button.

Note: The default severity value '0' cannot be deleted.

Once you have made all desired edits for a particular configuration click, **OK** to save the edits and exit the dialog, or click **Cancel** to exit the dialog without saving any changes. All available configurations can be accessed through the **Severity** drop-down list.

Flash/Beep

Selecting the **Flash** and/or **Beep** check boxes applies the flash or beep to the particular severity. Once the box has been checked, you can set a frequency for the occurrence of the flash or beep (slow, medium, or fast). This frequency will be displayed in the **Actions** section on the **Row** tab.

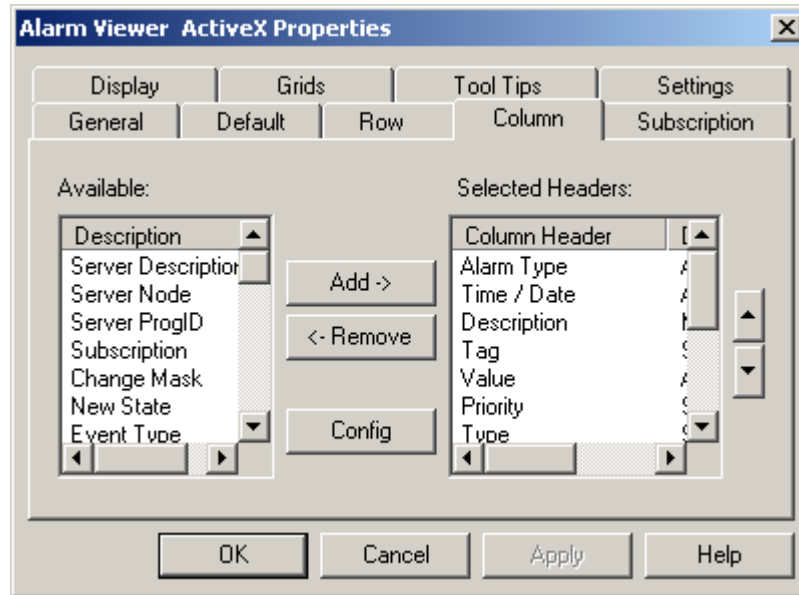
The **Flash/Beep** functionality can be activated on the **Row - Configure** dialog.

It is possible to change the default beep to a *.wav file by accessing **Settings > Control Panel > Sounds > Default Sound**. Please note that if you choose a large file as your default sound, your system may experience difficulty when trying to load it.

The **Flash** option simply inverts the text and background color to indicate the flashing.

Column

The **Column** tab of the **Alarm Viewer ActiveX Properties** dialog box, shown below, allows you to select which columns you want to appear in the viewer.



Alarm Viewer ActiveX Properties: Column Tab

By clicking the **Add** and **Remove** buttons in the middle of and the "up" and "down" arrows on the right side of this the **Column** tab, you can select which columns to include in the report as well as indicate the order of their appearance. To move an item from the **Available** list to the **Selected** list, simply highlight the item and click the **Add** button. To move an item back from the **Selected** list to the **Available** list, highlight the item and click the **Remove** button. To change the order of an item in the **Selected** list, simply select the item and click the "up" and "down" arrows located on the right-hand side.

Column Types

The available column types are listed in the table below.

Note: The following column type descriptions are taken directly from the OPC Alarms and Events Specifications. Please refer to the Specifications for complete information.

Column Type	Description
	The following items are present for all event types.
Source	The source of event notification. This Source can be used in the IOPCEventServer::TranslateToItemIDs method to determine any related OPC Data Access itemIDs.
Time	Time of the event occurrence. For conditions, time that the condition transitioned into the new state or sub-condition. For example, if the event notification is for acknowledgment of a condition, this would be the time that the condition became acknowledged.
Active Time	Array of active times corresponding to each Source and

	ConditionName pair. This parameter uniquely identifies a specific transition of the condition to the active state or into a different sub-condition and is the same as the SubCondLastActive condition attribute. Active times are passed to the client in the ActiveTime member of the ONEVENTSTRUCT by the IOPCEventSink::OnEvent callback. If the condition has become active again or transitioned into a different sub-condition at a later time, this acknowledgment will be ignored.
Message	Event notification message describing the event.
Subscription	Subscription to a given OPC Alarm and Event server.
Server Description	Information about the currently subscribed OPC Alarm and Event server.
Server Node	Node (computer) on which the currently subscribed OPC Alarm and Event server is located.
Event Type	OPC_SIMPLE_EVENT, OPC_CONDITION_EVENT, or OPC_TRACKING_EVENT for Simple, Condition-Related, or Tracking events, respectively.
Event Category	Event categories define groupings of events supported by an OPC Event server. Examples of event categories might include "Process Events", "System Events", or "Batch Events". Event categories may be defined for all event types, i.e. Simple, Tracking, and Condition-Related. However, a particular event category can include events of only one type. A given Source may generate events for multiple event categories. Names of event categories must be unique within the event server. The definition of event categories is server-specific.
Severity	Event severity (0-1000). The severity value is an indication of the urgency of the sub-condition. This is also commonly called 'priority', especially in relation to process alarms. Values will range from 1 to 1000, with 1 being the lowest severity and 1000 being the highest. Typically, a severity of 1 would indicate in event that is informational in nature, while a value of 1000 would indicate an event of catastrophic nature which could potentially result in severe financial loss or loss of life.
Number Event Attributes	The length of the specific event attribute array.
Attributes 1-20	User-specified event attributes.
	The following items are present only for condition-related events.
Condition Name	The name of the condition related to this event notification.
SubCondition Name	The name of the current sub-condition, for multi-state conditions. For a single-state condition, this contains the condition name.

Change Mask	Indicates to the client which properties of the condition have changed to have caused the server to send the event notification. It may have one or more of the following values: OPC_CHANGE_ACTIVE_STATE OPC_CHANGE_ACK_STATE OPC_CHANGE_ENABLE_STATE OPC_CHANGE_QUALITY OPC_CHANGE_SEVERITY OPC_CHANGE_SUBCONDITION OPC_CHANGE_MESSAGE OPC_CHANGE_ATTRIBUTE If the event notification is the result of a Refresh, these bits are to be ignored. For a “new event”, OPC_CHANGE_ACTIVE_STATE is the only bit that will always be set. Other values are server-specific. (A “new event” is any event resulting from the related condition leaving the Inactive and Acknowledged state.)
NewState	A WORD bit mask of three bits specifying the new state of the condition: OPC_CONDITION_ACTIVE, OPC_CONDITION_ENABLED, OPC_CONDITION_ACKED.
Quality	Quality associated with the condition state. Values are as defined for the OPC Quality Flags in the OPC Data Access Server specification.
AckRequired	This flag indicates that the related condition requires acknowledgment of this event. The determination of those events that require acknowledgment is server-specific. For example, transition into a LimitAlarm condition would likely require an acknowledgment, while the event notification of the resulting acknowledgment would likely not require an acknowledgment.
Active Time	Time that the condition became active (for single-state conditions), or the time of the transition into the current sub-condition (for multi-state conditions). This time is used by the client when acknowledging the condition (see IOPCEventServer::AckCondition method).
Cookie	Server-defined cookie associated with the event notification. This value is used by the client when acknowledging the condition (see IOPCEventServer::AckCondition method). This value is opaque to the client.
Time in Alarm	On a periodic basis, this field is updated with the amount of time a specific alarm has been in an active alarm state. The field is reset each time the message goes into alarm, and is only available for condition messages.
	The following is used only for tracking events and for condition-related events that are

	acknowledgment notifications.
ActorID	For tracking events, this is the actor ID for the event notification. For condition-related events, this is the AcknowledgerID when OPC_CONDITION_ACKED is set in NewState. If the AcknowledgerID is a NULL string, the event was automatically acknowledged by the server. For other events, the value is a pointer to a NULL string.

Event Type Values

Event Type	Value	Description
OPC_SIMPLE_EVENT	1	Simple event.
OPC_TRACKING_EVENT	2	Tracking event.
OPC_CONDITION_EVENT	4	Condition-Related event.

Change Mask Values

Change Mask Item	Value	Description
OPC_CHANGE_ACTIVE_STATE	1	The condition's active state has changed.
OPC_CHANGE_ACK_STATE	2	The condition's acknowledgment state has changed.
OPC_CHANGE_ENABLE_STATE	4	The condition's enabled state has changed.
OPC_CHANGE_QUALITY	8	The ConditionQuality has changed.
OPC_CHANGE_SEVERITY	16	The severity level has changed.
OPC_CHANGE_SUBCONDITION	32	The condition has transitioned into a new sub-condition.
OPC_CHANGE_MESSAGE	64	The event message has changed (compared to prior event notifications related to this condition).
OPC_CHANGE_ATTRIBUTE	128	One or more event attributes have changed (compared to prior event notifications related to this condition).

New State Values

New State	Value	Description
OPC_CONDITION_ENABLED	1	The condition has been enabled.
OPC_CONDITION_ACTIVE	2	The condition has become active.
OPC_CONDITION_ACKED	4	The condition has been acknowledged.

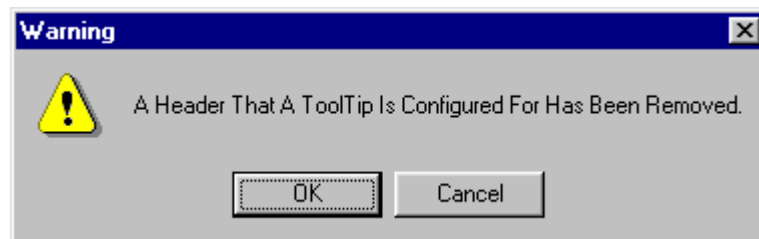
Note: **Alarm Type** cannot be deleted from the **Selected Headers** list.

Note: If you remove a header from the selected list that was used in the sort list, you will get the following warning message.



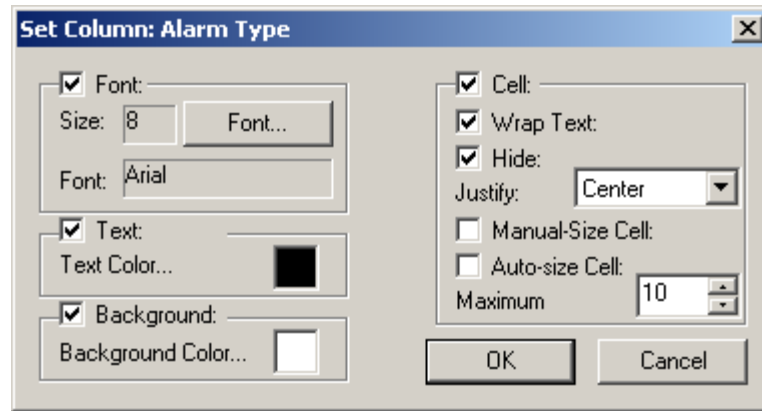
Click **OK** and open the **Display** tab to see how the **Sort** list has been changed.

Note: If you remove a header from the selected list that was used in the help or comment ToolTip configuration, you will get the following warning message:



Click **OK** and open the **ToolTips** tab to see how the ToolTip configuration has been changed.

You can also change the appearance of a column by either double-clicking on an item in the **Selected** list or selecting the item and clicking the **Config** button. The **Set Column** configuration dialog for the item opens, as shown below. From this dialog, you can set the font, color, width, and cell properties by selecting and changing the various options available.



Setting the Column Properties

To change the configuration for **Font**, **Text Color** or **Cell**, the appropriate box must be checked. To change the font size and style, click the **Font**. Once you have made the desired changes, the new size and style will appear in the corresponding boxes.

Note: It is not possible to change the size and style without using the **Font** dialog.

To change the column color, click the **Background Color** box and use the **Color** dialog to make your changes.

To change the text column color, click the **Text Color** box and use the **Color** dialog to make your changes

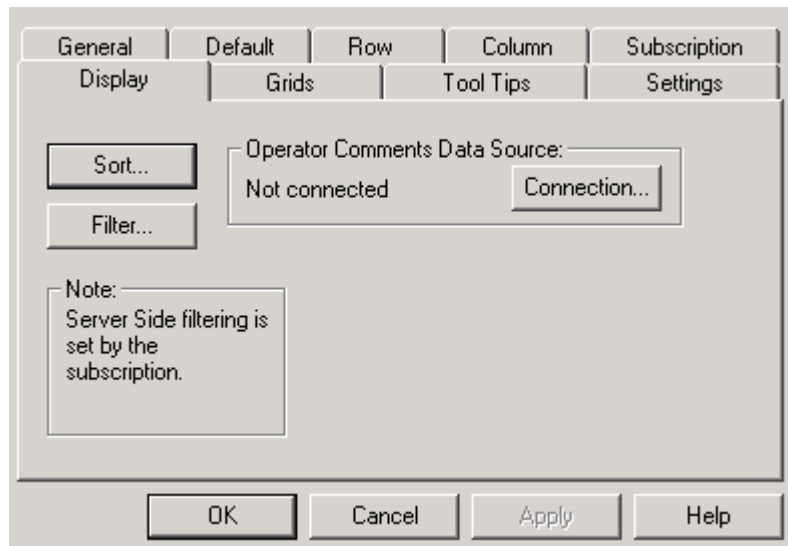
The **Cell** section configures how the text appears within each cell and how the size of each cell is determined. Checking **Wrap Text** box makes the text wrap when it hits the end of the particular cell. When this option is not applied, text will continue in one line although only part of the text will be visible, depending on the size of the cell. Please note that the **Wrap Text** feature is available only with multiline rows. The **Justify** field determines the justification (left, center or right) of the text within each cell.

There are three options for determining the size of each cell:

- **Manual-Size Cells:** Checking this box allows you to manually set the width of each column. Manual sizing of columns can only be done during runtime.
- **Auto-Size Cells:** When this box is checked, the viewer automatically sets the width of each column. The viewer automatically sets each column to the width nearest to the size of the ActiveX divided by the number of visible columns.
- **Maximum Length for Each Cell:** Currently this is a range from 1 to 1000, and it applies the maximum cell length to all cells.

Display

The **Display** tab of the **Alarm Viewer ActiveX Properties** dialog box, shown below, configures options for sorting or client-side filtering.



Alarm Viewer ActiveX Properties: Display Tab

To configure the sorting feature, click the **Sort** button. This opens the **Sort** dialog, as shown in the figure below.

The **Sort Alarms by** section indicates the first criteria the Alarm Viewer should use when sorting all current alarms. The following **Then by** sections indicate how the viewer should sort alarms that have the same value for the criteria selected in the first section.

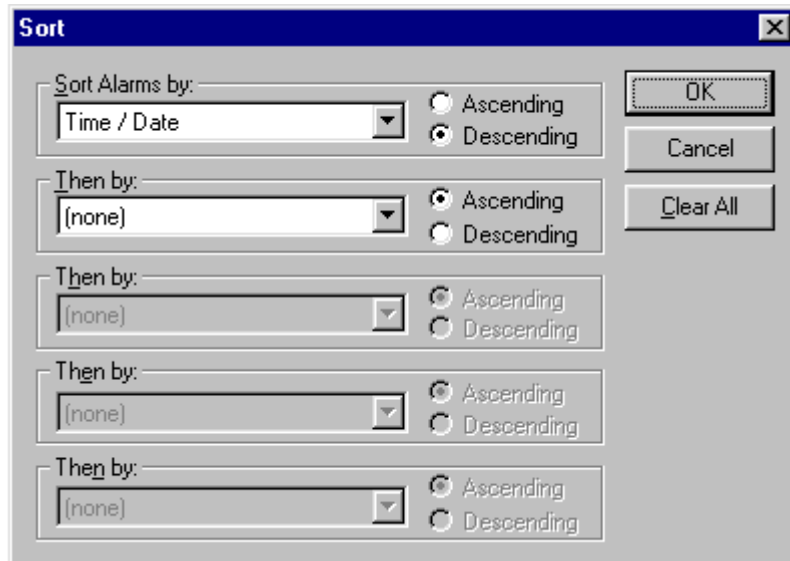
Note: It is not necessary to have any values listed in the **Then by** sections.

The **Sort** dialog allows you to configure which criteria to sort the alarms by. It also allows for multilevel sorting up to five levels with ascending/descending specific to each level. The Alarm Viewer will always sort. The default sort for a new viewer is **Time/Date**.

Sort Ascending sorts the data in the column in ascending order (i.e. from least to greatest).

Sort Descending sorts the data in the column in ascending order (i.e. from greatest to least).

Note: The **Sort** dialog, including any configuration, is also available in runtime.

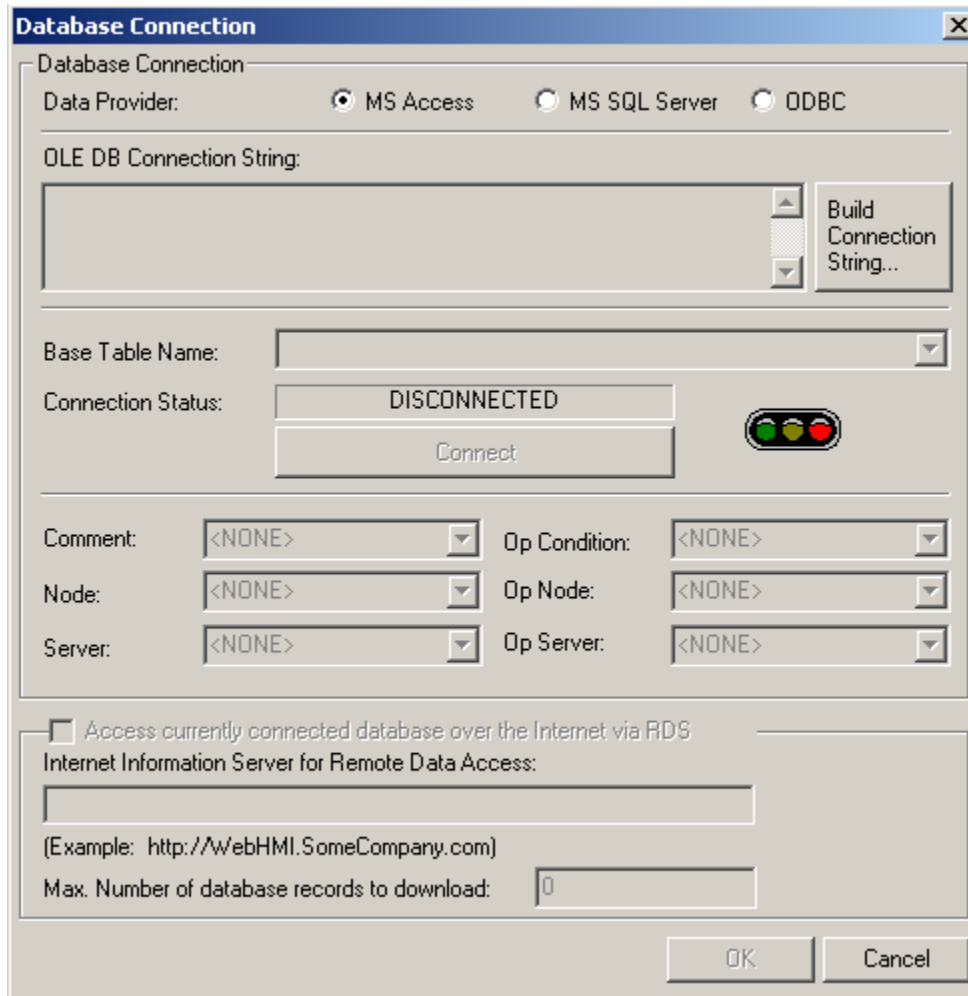


Sorting Alarms

Connecting to a Data Source

To connect to a data source:

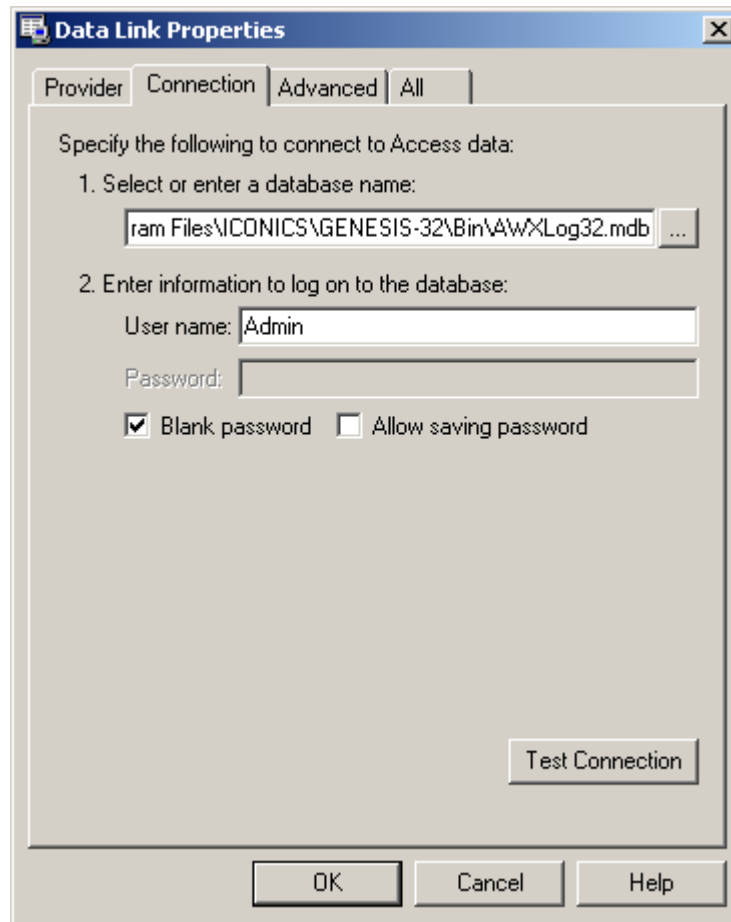
1. Click the **Connection** button. This opens the **Database Connection** dialog box, as shown below.
2. Under **Data provider**, select from Microsoft Access, Microsoft SQL Server, or ODBC (Open Data Base Connectivity), as shown in the figure below.



Connecting to a Database

3. Click the **Build Connection String** button. This opens the Microsoft **Data Link Properties** dialog box, as shown in the figure below. In the **Connection** tab specify the data source and then click **OK**.

Note: All data source connections are made through the **Data Link Properties** dialog box. The **Connection** tab settings may vary depending on which data provider you selected. Click the **OK** button. Click the **Help** button at any time to view the Microsoft Data Link help documentation.



Specifying a Data Source

4. The data source reference appears in the **Database Connection** dialog box in the **OLE DB Connection String** field, as shown below. The **Base Table Name** field lists the tables in the database. You can select a specific table from the drop-down list to use as the reference table for the database.

Note: The **Base Table** list varies according to the active Alarm Logger configuration database.

5. Click the **Connect** button to connect to the database. The traffic light icon changes to green when the connection is successful, as shown in the figure below.

Database Connection

Database Connection

Data Provider: ☒ MS Access ☐ MS SQL Server ☐ ODBC

OLE DB Connection String:

Provider=Microsoft.Jet.OLEDB.4.0;Password="";Data Source=C:\Program Files\ICONICS\GENESIS-32\Bin\Aw\Log32.mdb;Persist Security Info=True

Build Connection String...

Base Table Name: EventLog

Connection Status: CONNECTED

Disconnect

Comment: AckComment Op Condition: ConditionName

Node: Node Op Node: Node

Server: ActorID Op Server: ActorID

☒ Access currently connected database over the Internet via RDS

Internet Information Server for Remote Data Access:

http://WebHMI.Iconics

(Example: http://WebHMI.SomeCompany.com)

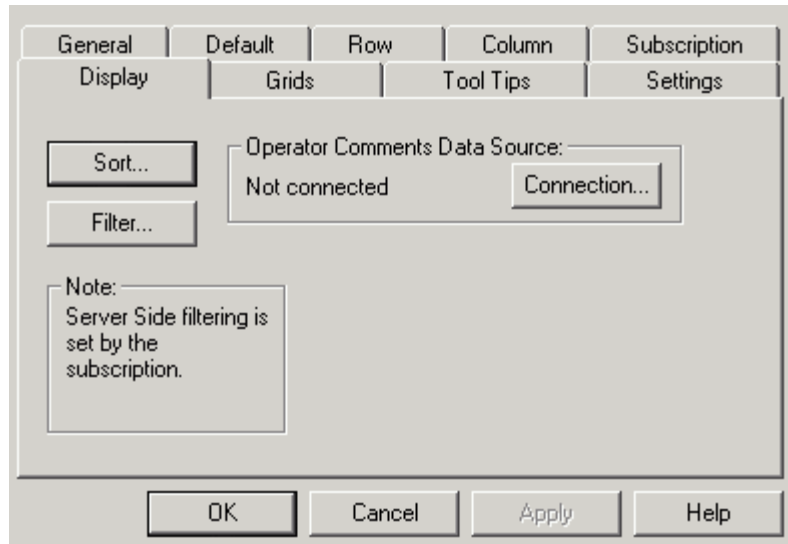
Max. Number of database records to download: 5

OK Cancel

Database Connection

Web Access and Support for Operator Comments

The **Display** tab of the **Alarm Viewer ActiveX Properties** dialog box, shown below, also allows you to connect to an operator comments database.

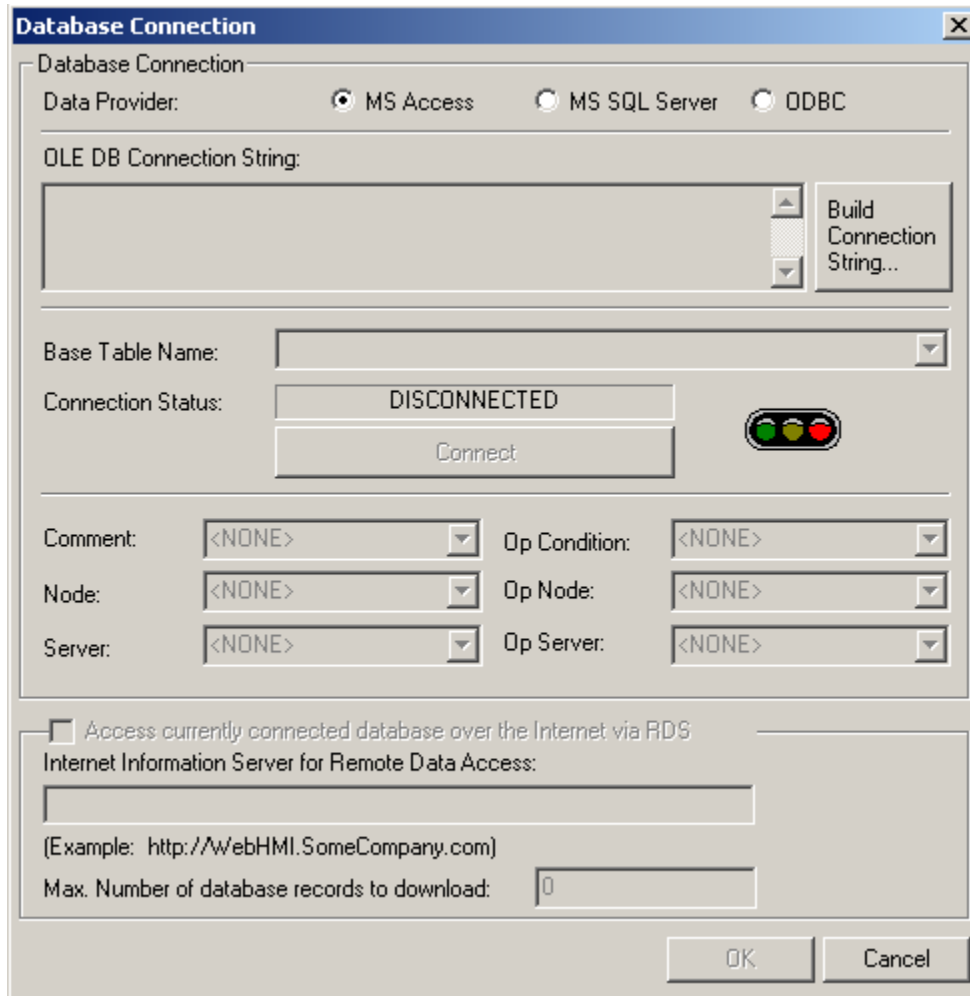


Alarm Viewer ActiveX Properties: Display Tab

Connecting to the Operator Comments Database

In most situations, you will use the **AWXLog32.mdb** Microsoft Access database (located in the GENESIS32 Bin directory) as your operator comments database. This is the standard Alarm Logger database. To connect to the operator comments database:

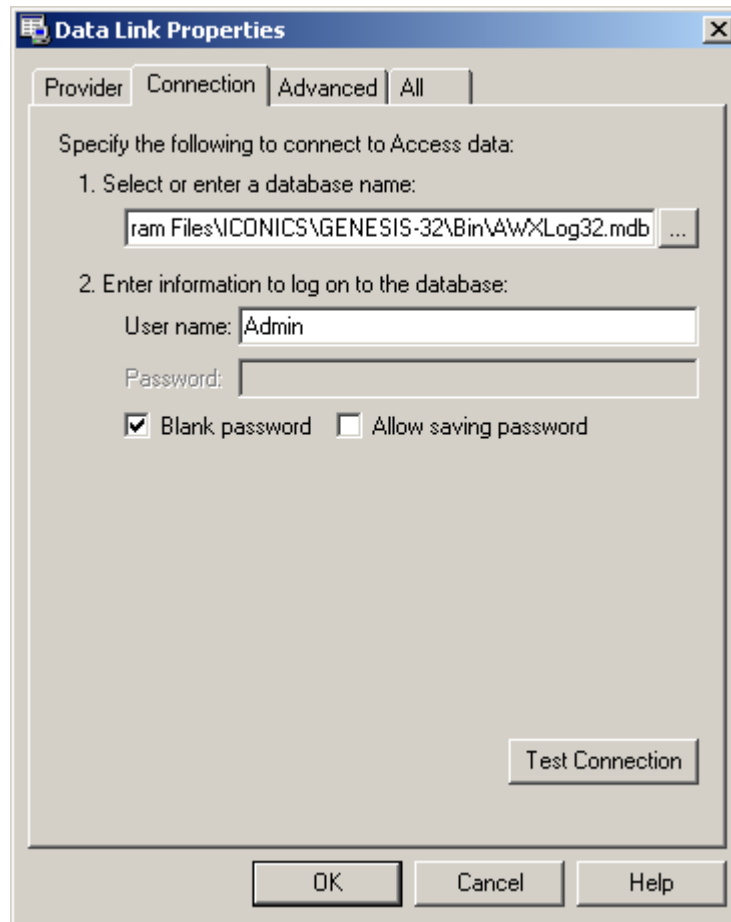
1. Click the **Connection** button on the **Display** tab. This opens the **Database Connection** dialog box, as shown below.
2. Under **Data provider**, select **Microsoft Access**, as shown in the figure below.



Connecting to a Database

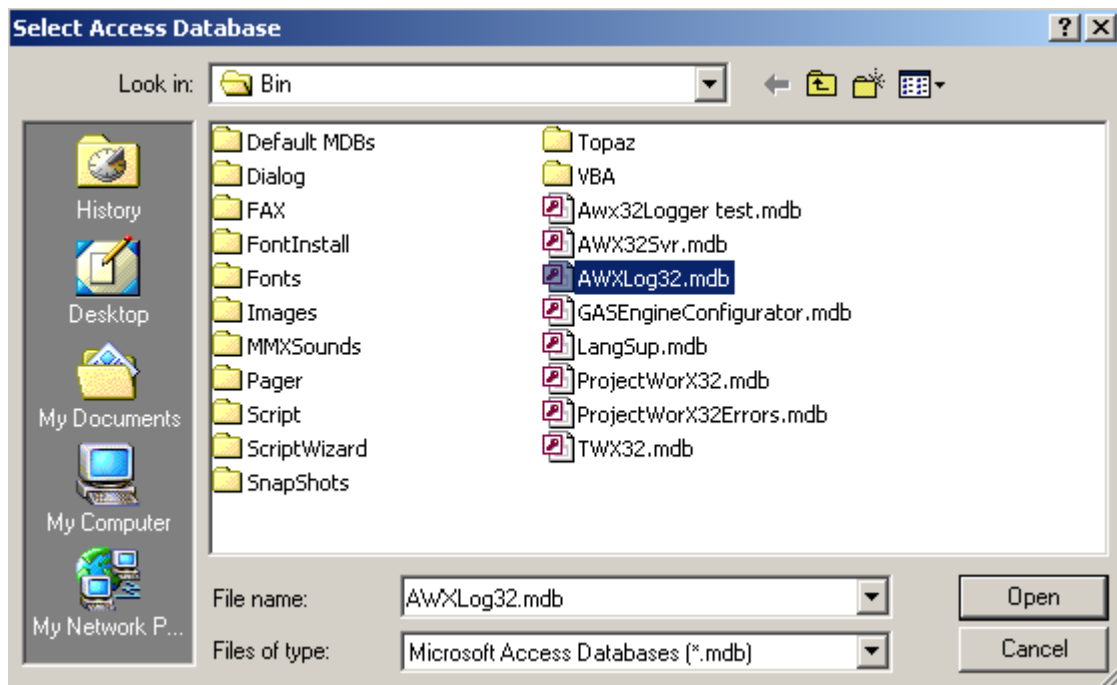
3. Click the **Build Connection String** button. This opens the Microsoft **Data Link Properties** dialog box, as shown in the figure below.

Note: All data source connections are made through the **Data Link Properties** dialog box. The **Connection** tab settings may vary depending on which data provider you selected. Click the **OK** button. Click the **Help** button at any time to view the Microsoft Data Link help documentation.



Specifying a Data Source

4. Click the ... button and select the **AWXLog32.mdb** file from the bin directory. Click the **Open** button.



Selecting the Operator Comments Database

5. The data source reference appears in the **Database Connection** dialog box in the **OLE DB Connection String** field, as shown below. In the **Base Table Name** field, select **EventLog** from the drop-down list.

Note: The **Base Table** list varies according to the active Alarm Logger configuration database.

6. Click the **Connect** button to connect to the database. The traffic light icon changes to green when the connection is successful, as shown in the figure below.

Database Connection

7. Now you must specify which column in the database table you will use to for logging and storing comments. In the **Comment** box, select a column from the drop-down list. For most cases, only the **Comment** box should be configured. The exception is the case where the system is logging from two different alarm servers, using the same tag names in both alarm server configurations. This is the only case where the **Node**, **Server**, **Op Condition**, **Op Node**, and **Op Server** selections should be used.

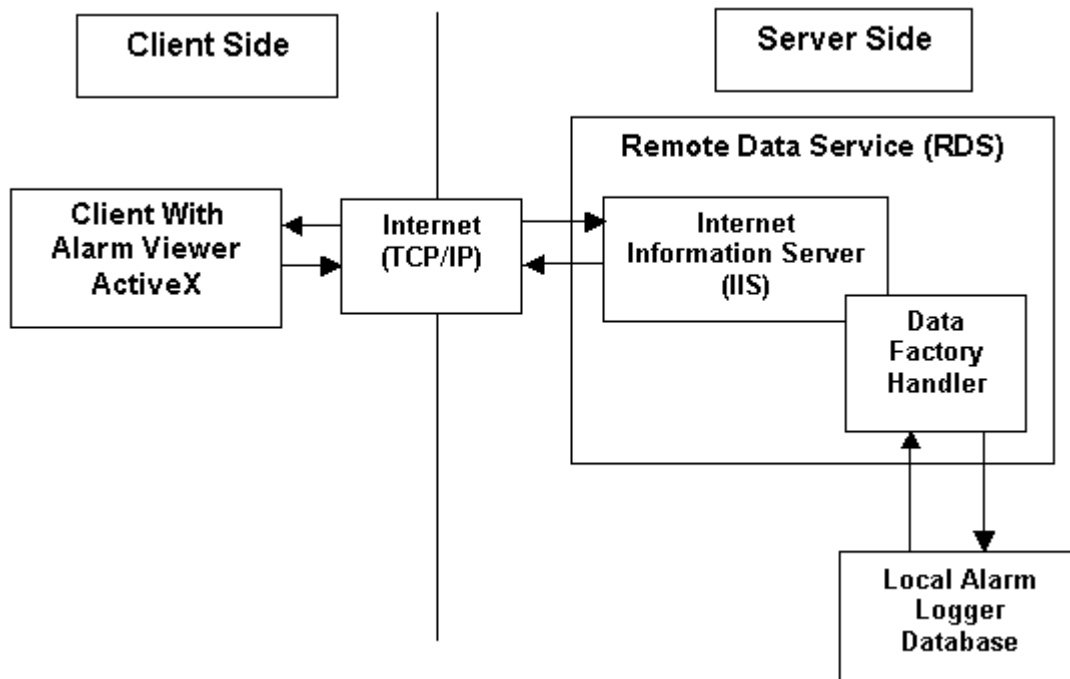
Enabling Remote Web Access to Operator Comments

The Alarm Viewer enables you to gain access to operator comments from the currently connected Alarm Logger database over the Internet. To enable Web access to operator comments, check the **Access currently connected database over the Internet via RDS** check box at the bottom of the **Database Connection** dialog box. A **Remote Data Service (RDS)**, which is hosted by an Internet Information Server (IIS), enables the downloading of operator comments data from the logger database to a client over the Internet.

Once you have connected to the operator comments database. Simply specify the URL name or IP address of the IIS Web server in the **Internet Information Server for Remote Data Access** field in the **Database Connection** dialog box. You can also specify a maximum number of database records to download from the logger database.

When Internet Access to the logger database is enabled, the client makes a request over the Internet to the IIS Remote Data Service. The RDS uses a special component, called the Data Factory Handler, to relay the request to the local logger database on the server, as shown in the figure below. The RDS retrieves the relevant information for the alarm report from the logger database so that the client can download the data from the server over the Internet. The client machine then displays the data in the Viewer ActiveX per the client's configuration settings.

In most typical scenarios, the Viewer ActiveX configuration is done on the server side. Configuration data saved as a *.awv file or a *.gdf display containing the Viewer ActiveX are published to an HTML document on a WebHMI server.



Downloading Operator Comments From a Remote Logger Database

Tips for Remote Data Access

A good understanding of Remote Data Access principles is a key element to building a successful solution for a Web-enabled alarm reporting system. Since the volume of alarm databases can reach hundreds of megabytes, it is very important to keep in mind the following aspects.

Maximum Number of Downloadable Database Records and Adequate Record Filtering

The setting for the maximum number of records to download (Nmax) indicates that any database query operations would return no more than Nmax alarm records (lines to the grid). If more than Nmax records from the current table match the current filtering criteria, then only the portion of the record set containing top Nmax records is returned. To avoid such record set cut-off, strong restrictive record filters should be used all the time. Since the record filters are processed on the database side, the database server uses all existing records to build the record set.

Proper Choice of the Back-end Database

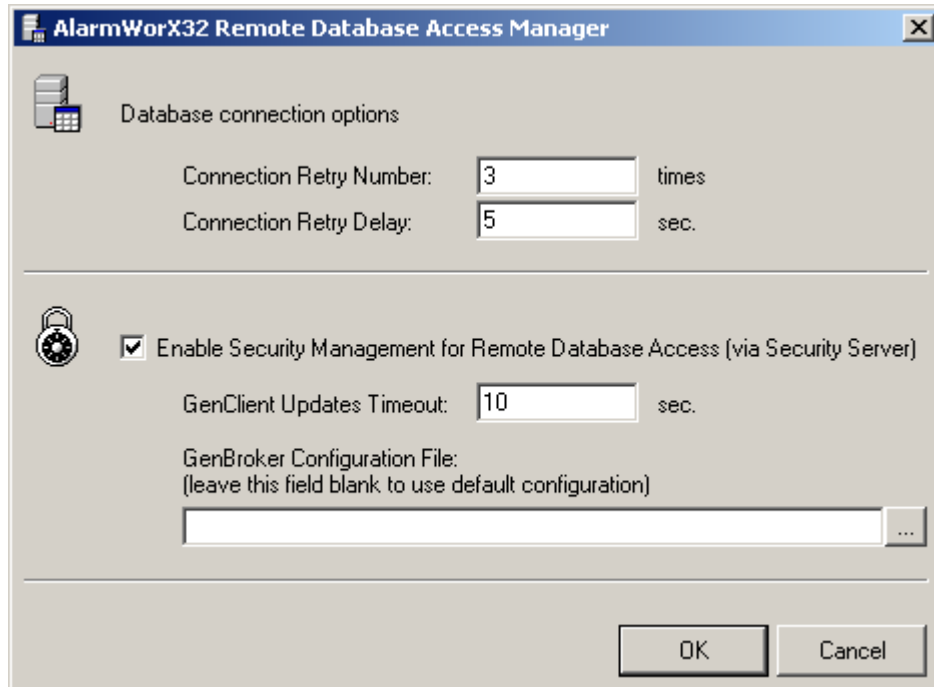
It should be noted that the use of Microsoft Access databases (*.mdb) is not recommended for Web Access. Poor memory management in Microsoft Jet Engine creates a serious scalability problem for any type of Web database application. **The recommended back-end database for Web alarming is MSDE (Microsoft Desktop Engine) or Microsoft SQL Server.**

Table Management in Alarm Logger Configuration

To provide further optimization for the Alarm Logger database operations, the table management should be enabled in the logger configuration. See the Alarm Logger Configurator documentation for more information.

Using the Remote Database Access Manager

The **AlarmWorX32 Remote Database Access Manager**, shown in the figure below, is a separate utility (the **AWXRep32RDSSM.exe** file) located in the GENESIS32 Bin directory. This utility configures the communication parameters for the Data Factory Handler (described above) so that the AlarmWorX32 Viewer ActiveX can connect to a remote Alarm Logger database and retrieve data from the database. It also enables security for remote database access via the GENESIS32 Security Server.



AlarmWorX32 Remote Database Access Manager

Database Connection Options

The **Database Connection Options** section of the AlarmWorX32 Remote Database Access Manager sets the following connection parameters for the Data Factory Handler:

- **Connection Retry Number:** Specifies the maximum number of times the Data Factory Handler will try to reconnect to the Alarm Logger database in case the connection fails.
- **Connection Retry Delay:** Sets the amount of time (in seconds) the Data Factory Handler waits between subsequent connection retries to the Alarm Logger database.

Security Management

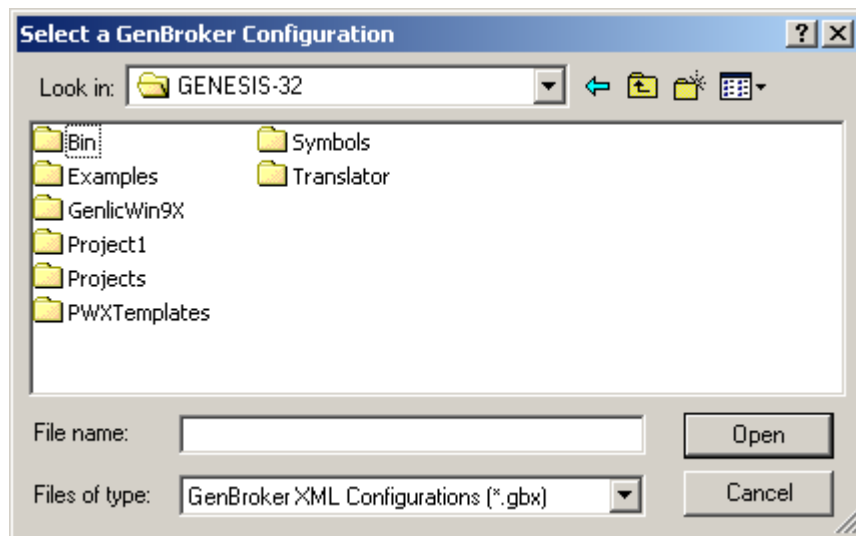
The AlarmWorX32 Remote Database Access Manager allows you to enable security for remote database access via the GENESIS32 Security Server. Check the **Enable Security Management for Remote Database Access (via Security Server)** check box. When security is enabled, each time a user tries to access the Alarm Logger Database, the Security Server will check to verify whether that user is allowed to have remote access to the database.

Note: For information about the Security Server configuration and login, please see the Security Configurator and Security Login Help documentation.

Because the database is accessed over the Internet via TCP/IP communications, GenBroker communications over OPC over TCP/IP must also be enabled. Thus, you must specify a **GenBroker Configuration File** (.gbc or .gbx). Clicking the ... button to the right of this field allows you to browse for a GenBroker configuration file, as shown in the figure below. If you do not specify a GenBroker configuration file, or if no Security Server is specified in the GenBroker configuration, then the default system GenBroker configuration file will be used.

Note: If you are using WebHMI, it is recommended that the Security Server be located on the same machine as the WebHMI Server. For information about how to configure GenBroker communications, please see the GenBroker Help documentation.

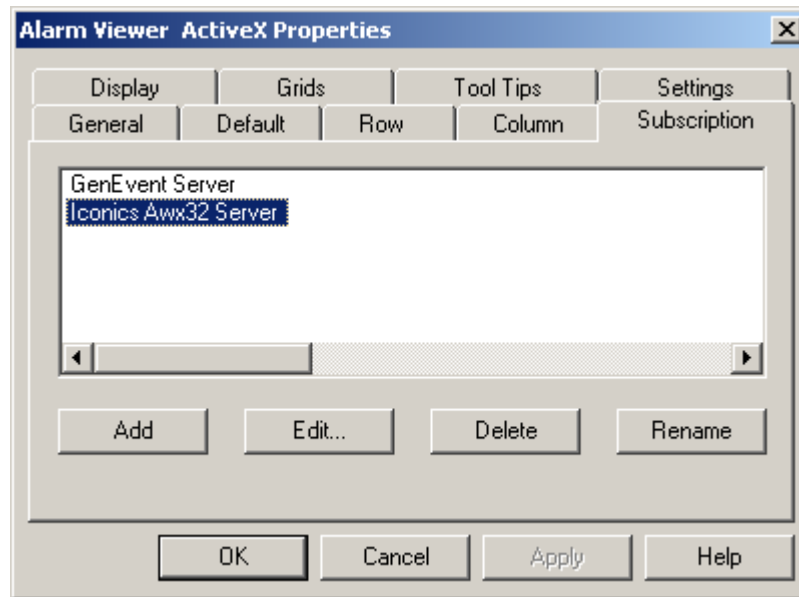
The **GenClient Updates Timeout** sets the maximum amount of time (in seconds) the Data Factory Handler will wait for Security Server validation of a data request from the client (i.e. the client with the Alarm Viewer ActiveX installed) before returning an "Access Denied" error.



Selecting a GenBroker Configuration File

Subscription

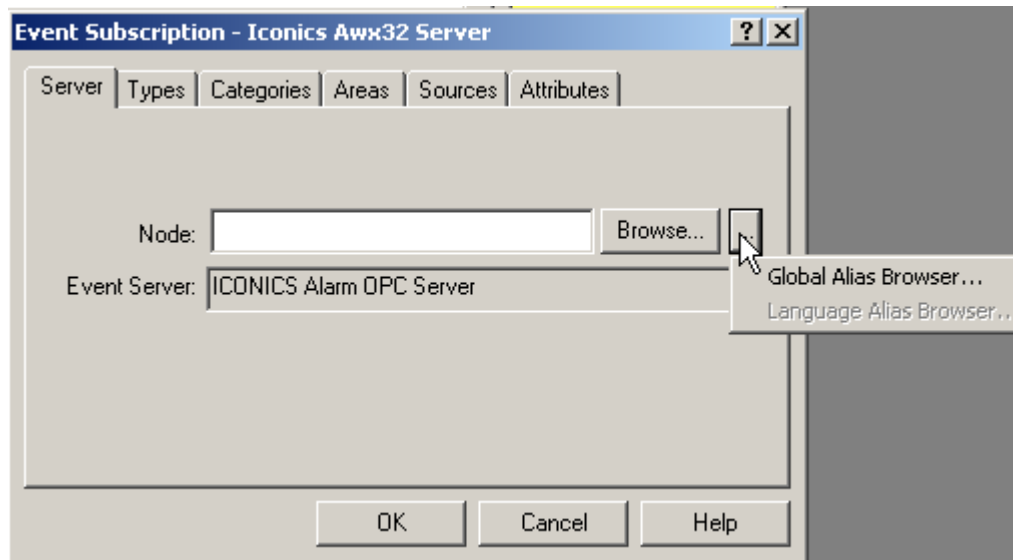
The **Subscription** tab of the **Alarm Viewer ActiveX Properties** dialog box, shown below, determines what type of OPC connection will be made. It allows you to add, rename, delete, or edit a subscription to a particular alarm. To add a new subscription, click the **Add** button.



Alarm Viewer ActiveX Properties: Subscription Tab

The subscription named "New Subscription" appears. This subscription does not contain any data, so it is necessary to immediately edit the new subscription. To do so, click the **Edit** button to open the **Event Subscription** dialog box, as shown in the figure below. It is only possible to add or edit subscriptions that are connected to active OPC Alarm and Event servers.

Note: It is possible for a server to have more than one subscription. In fact, it is a very effective way to achieve filtering.

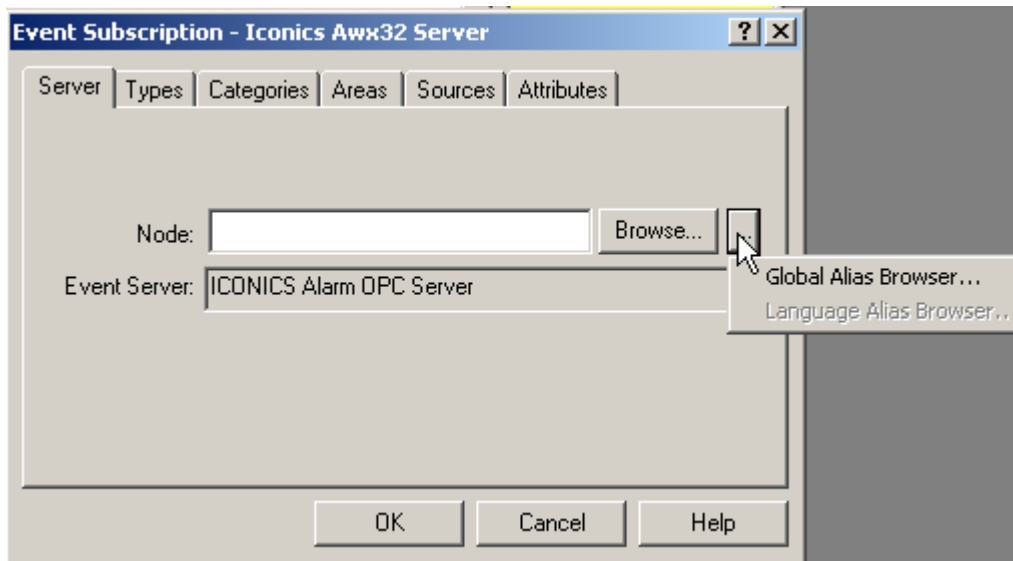


Creating an Event Subscription

Server
Types
Categories
Areas
Source
Attributes

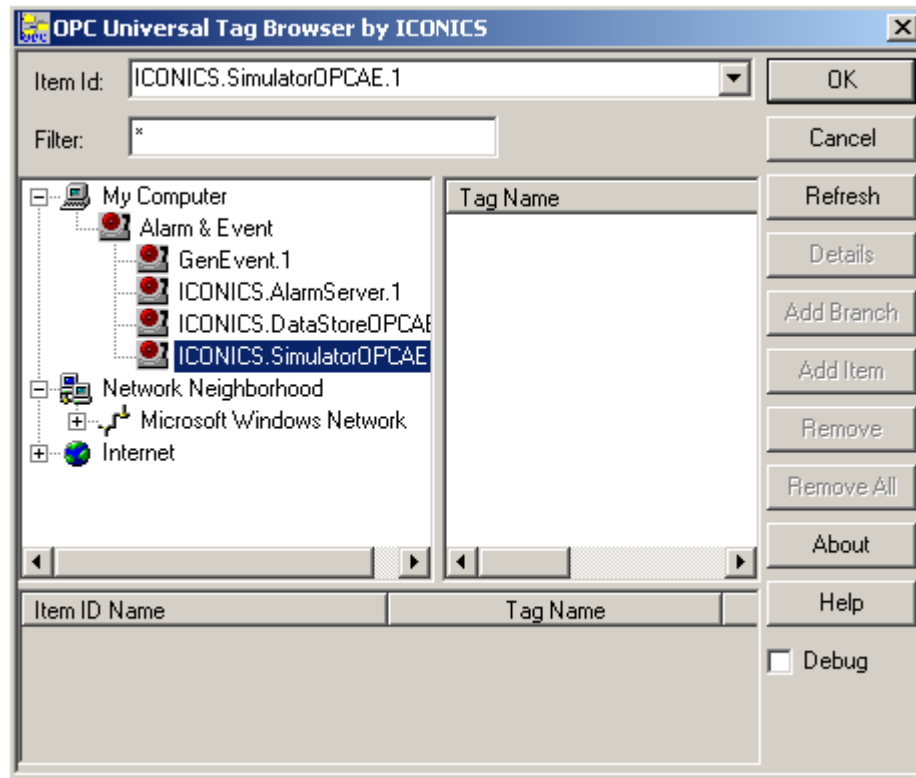
Server

The **Server** tab of the **Event Subscription** dialog box, shown in the figure below, allows you to select an **Alarm and Event Server** and **Node** for each subscription. To select the event server, click the **Browse** button.



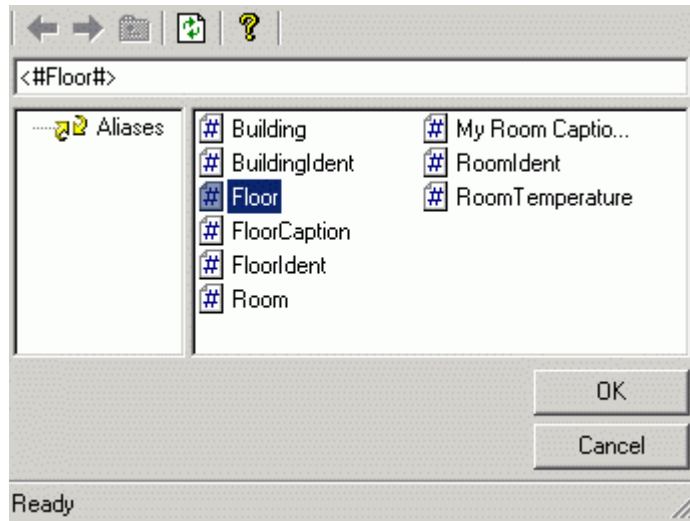
Event Subscription Dialog Box: Server Tab

This opens the **OPC Universal Tag Browser**, shown in the figure below, which lists all available Alarm and Event OPC servers. Select the desired server, and click **OK**. For local servers, it is not necessary to fill in the **Node** field.



Selecting an Alarm and Events Server From the OPC Universal Tag Browser

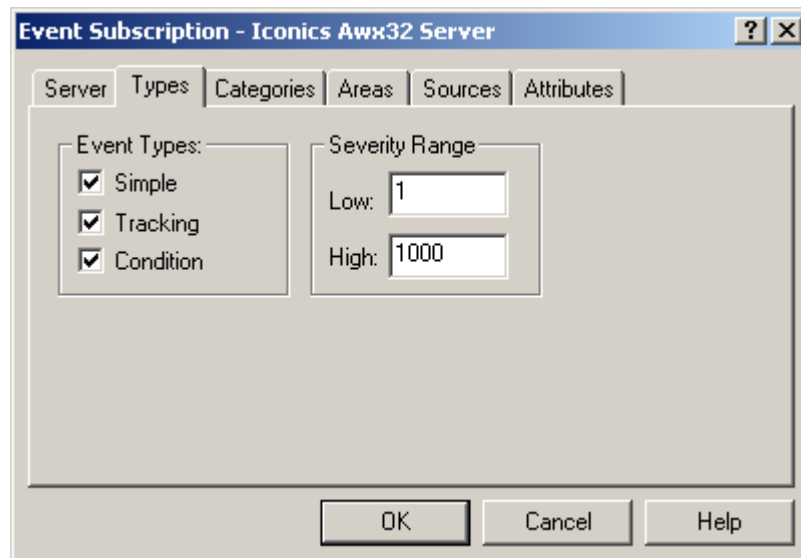
You can also select global aliases to use in the **Node** field. Clicking the ... button and selecting **Global Alias Browser** from the pop-up menu opens the Global Alias Browser, as shown in the figure below. Select a global alias from the Global Alias Browser, which includes all global aliases in 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.



Selecting an Alias From the Global Alias Browser

Types

The **Types** tab of the **Event Subscription** dialog box, shown in the figure below, allows you to configure which OPC-defined event types each subscription should have, as well as to set the ranges for severity (priority). A value of "0" represents the low severity value, and "1000" represents the high severity value. Please note that OPC Alarm and Event (AE) servers are required to scale severity values to the OPC ranges (i.e. an AE server that contained two severity ranges would convert these to "0" and "1000").



Event Subscription Dialog Box: Types Tab

Simple: These messages state information but do not have alarm status, nor do they contain information on what initiated the message. This includes the

following information: Source, Time, Type, EventCategory, Severity, Message, Cookie, and server-specific items.

Example: "FIC101, 12:0:0 1/1/99, Simple, Category1, 100, 'Shift Change', 1"

Simple messages would be similar to an event.

Tracking: These messages contain the additional information about the client that initiated the event. This includes the following information: Source, Time, Type, EventCategory, Severity, Message, Cookie, ActorID, and server-specific items.

Example: "FIC101, 12:0:1 1/1/99, Tracking, Category1, 300, 'Pump pressure Set to 10 psi', 1, Station 12"

Tracking messages are similar to event messages in that the cause of the event is important. An example would be an operator changing a setpoint value. This type of message does not include acknowledge capability.

Note: Simple and tracking messages are removed from the alarm viewer via the acknowledge mechanism.

Condition: These messages contain all of the above information but also include an acknowledgement portion. This includes the following information: Source, Time, Type, EventCategory, Severity, Message, Cookie, ConditionName, SubConditionName, ChangeMask, NewState, ConditionQuality, AckRequired, ActiveTime, ActorID and server-specific items.

Example: "FIC101, 12:0:3 1/1/99, Condition, Category1, 700, 'Pump pressure to high', 1, Limit, HiHi, 1, Active Enabled, Good, TRUE, 12:0:2 1/1/99"

Condition messages would be considered a "typical" alarm message with acknowledge capability.

For further details on any of the included information, please refer to the OPC Alarm and Events specification.

Categories

The **Categories** tab of the **Event Subscription** dialog box, shown in the figure below, allows you to select the categories for a subscription. Select the category from the list of available categories, and then click the **Add ->** button. The category will appear in the **Subscribed** list. To remove a category from this list, select it in the **Subscribed** list and click the **<- Remove** button. If no categories are listed in the **Subscribed** list, then all categories are selected by default.

Event Types and Categories

All events generated are of the OPC defined Event Type **Condition**. The AlarmWorX32 Server defines the following **Event Categories** and uses the identical names for the **Condition Names**:

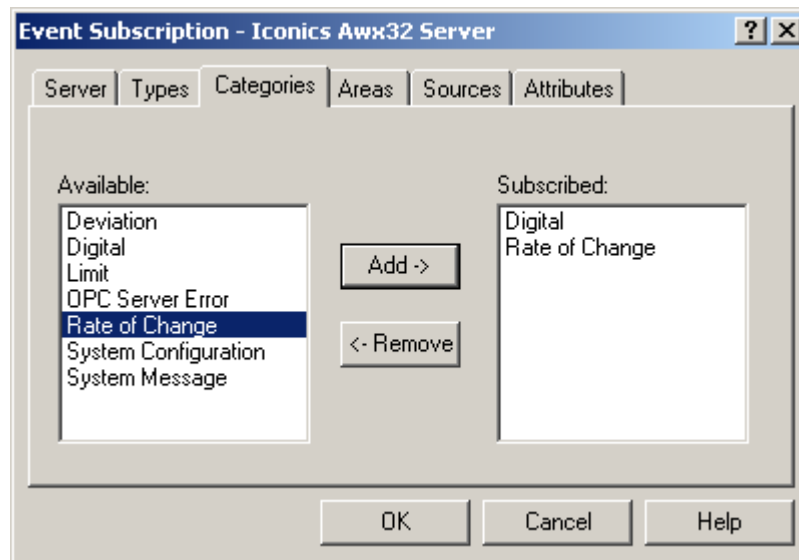
- Deviation
- Digital
- Limit
- Rate of Change

The **Limit** and **Deviation** conditions have the following Sub-conditions:

- HiHi
- Hi
- Lo
- LoLo

Note: The following categories are not used in the ICONICS Alarm Server:

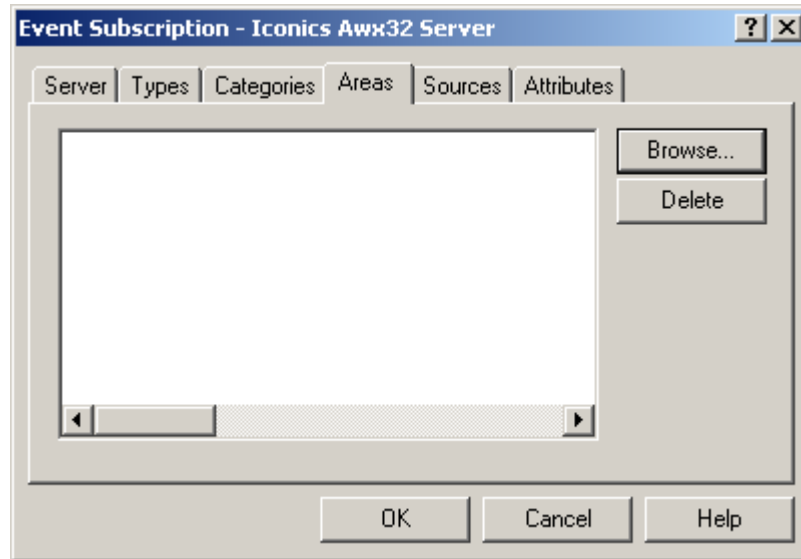
- OPC Server Error
- System Configuration
- System Message



Event Subscription Dialog Box: Categories Tab

Areas

Use the **Areas** tab of the **Event Subscription** dialog box, shown in the figure below, to select an area or a group of areas for filtering in a subscription. An **area** is used to group OPC alarm tags. This information can be used by clients for filtering purposes. A tag can exist under multiple areas, and an area can have a multiple levels. An area is defined by its name and the name of its parent branches. For example, a branch called Area 1 under plant Area A is different from Area 1 under plant Area B. Areas can be configured in the AlarmWorX32 Server Configurator.



Event Subscription Dialog Box: Areas Tab

Clicking the **Browse** button opens the **OPC Event Server Area / Source Browser**, shown in the figure below, which lists all available areas for your event server. Area subscription supports wildcards, which subscribe to the format of the Microsoft Visual Basic 'like' command. For example, "Area1*" will subscribe to all alarm areas that contain strings beginning with "Area1". "Area1*" will subscribe to the root area and its "child" areas. A detailed explanation of the wildcard support can be found in the OPC Alarm and Events documentation. It is recommended that you thoroughly read the wildcard documentation before attempting to use complicated expressions.

Note: For the like operator: "string" equals the string to search in; "pattern" equals the string to search for (can include wildcards); nonzero for case-sensitive search; zero for case-insensitive search. String syntax is "\$string\$".

You can use these special characters in pattern matches in string:

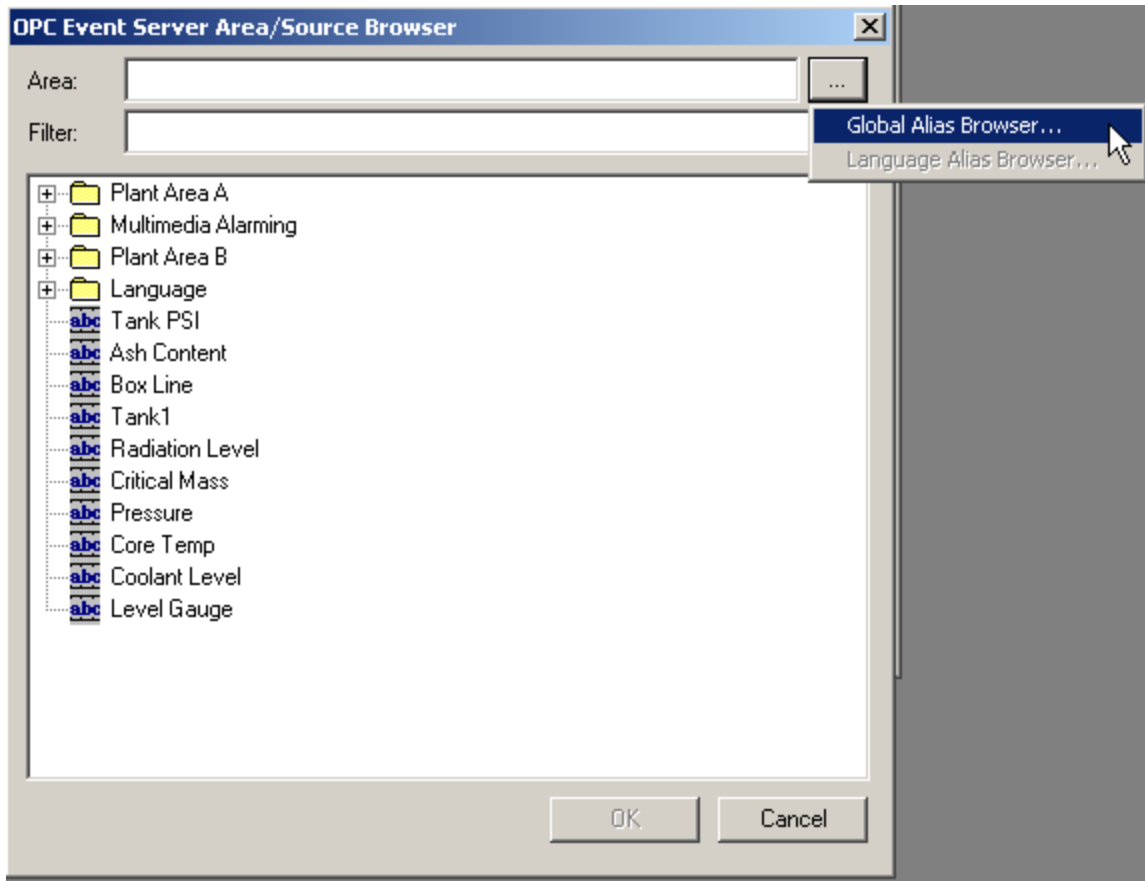
- ? Any single character.
- Zero or more characters.
- # Any single digit (0-9).
- [charlist] Any single character in charlist.
- [!charlist] Any single character not in charlist.

It is also possible to delete an area from the **Areas** tab. To delete an area, select it from the list of areas for this particular subscription, and then click the **Delete** button. Not all OPC Alarm and Event servers support area filtering as part of the subscription.

Note: If no areas are listed, then all areas are selected by default.

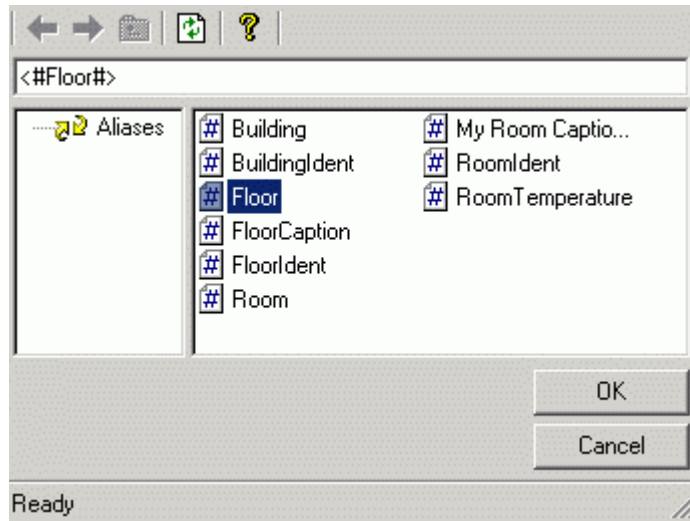
Note: It is recommended that you have an area in the alarm server to which you subscribe; this ensures that the relevant alarms are received. Any available

areas listed in the active Alarm Server configuration database are available to choose from in the Area/Source Browser, as shown in the figure below.



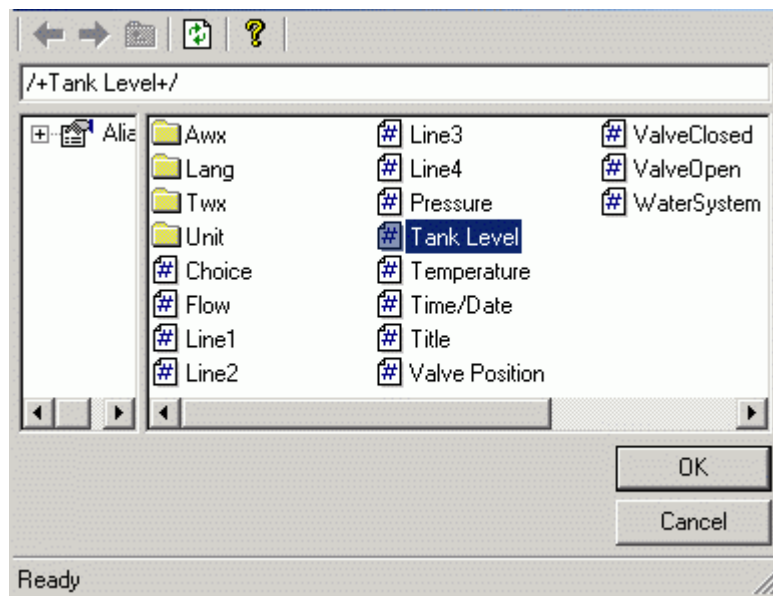
OPC Event Server Area/Source Browser

You can also select global aliases to use in the **Area** field. Clicking the ... button and selecting **Global Alias Browser** from the pop-up menu opens the Global Alias Browser, as 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.



Selecting an Alias From the Global Alias Browser

Clicking the ... button and selecting **Language Alias Browser** from the pop-up menu opens the Language Alias Browser, as shown in the figure below. The browser includes all languages aliases in the language database. All language aliases that are configured in the Language Configurator are conveniently available to choose from inside the browser. The tree control of the Language Configurator is mimicked in the tree control of the Language Alias Browser. Select a language alias by double-clicking the alias name. The alias name appears at the top of the browser, which automatically adds the /+ and +/ delimiters to the alias name. Click the **OK** button.

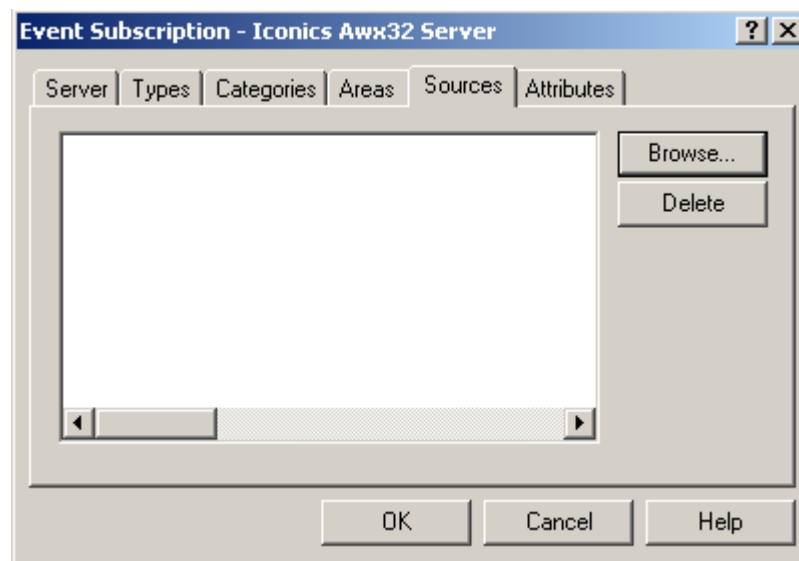


Selecting an Alias From the Language Alias Browser

Sources

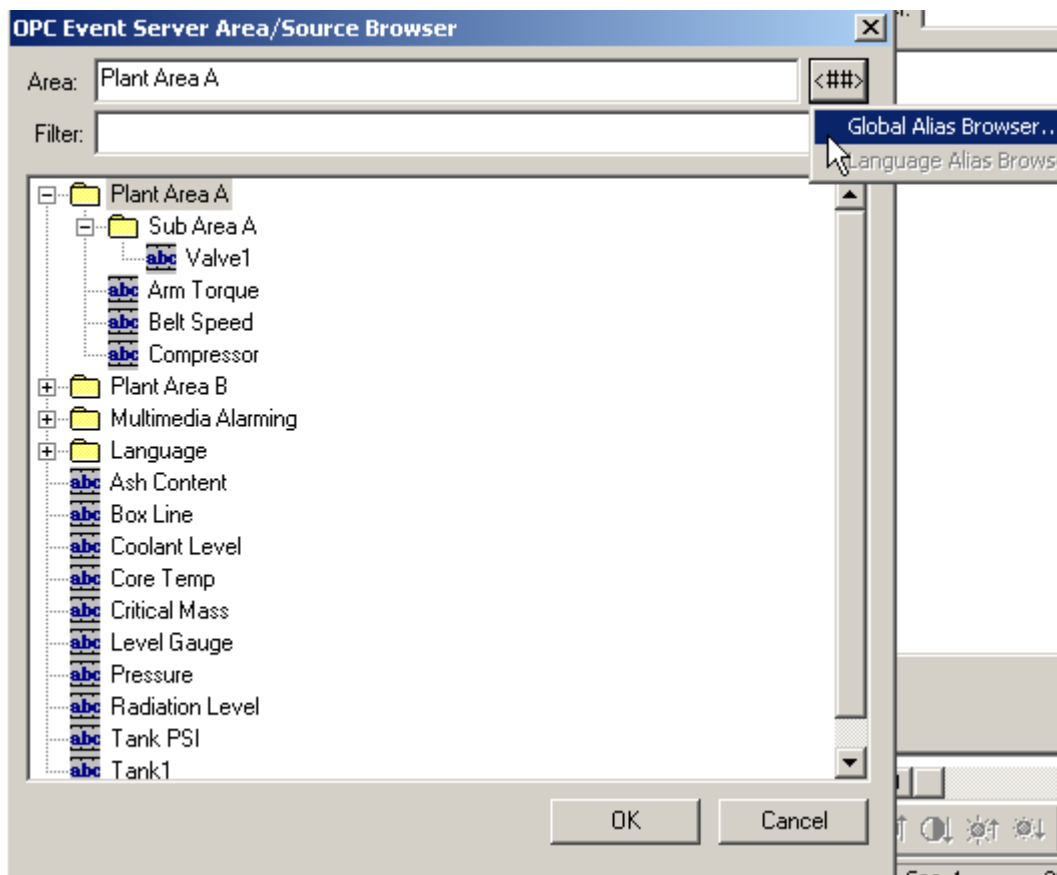
The **Sources** tab of the **Event Subscription** dialog box, shown in the figure below, allows you to filter on a source or group of sources for a particular subscription. A **source** is an OPC alarm tag in an alarm configuration. In the AlarmWorX32 Server Configurator, a source can be associated with a particular area (group of alarm tags). It is also possible to delete a source for a particular subscription. To delete a source, select it from the list of sources for this particular subscription and click the **Delete** button. Not all OPC Alarm and Event servers support source filtering as part of the subscription.

Note: If no sources are listed, then all sources are selected by default. Any available sources listed in the active Alarm Server configuration database are available to choose from in the Area/Source Browser, as shown in the figure below.



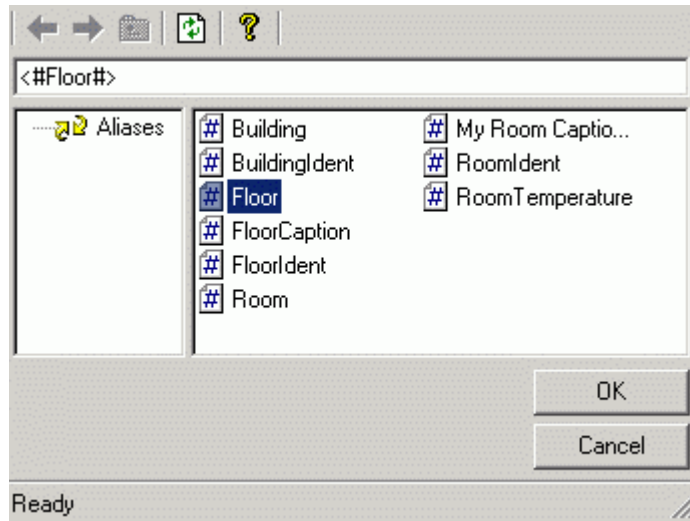
Event Subscription Dialog Box: Sources Tab

To select a source, click the **Browse** button and select one from the **OPC Event Server Area / Source Browser**, shown in the figure below. Source subscription provides the wildcard support found in the area subscription.



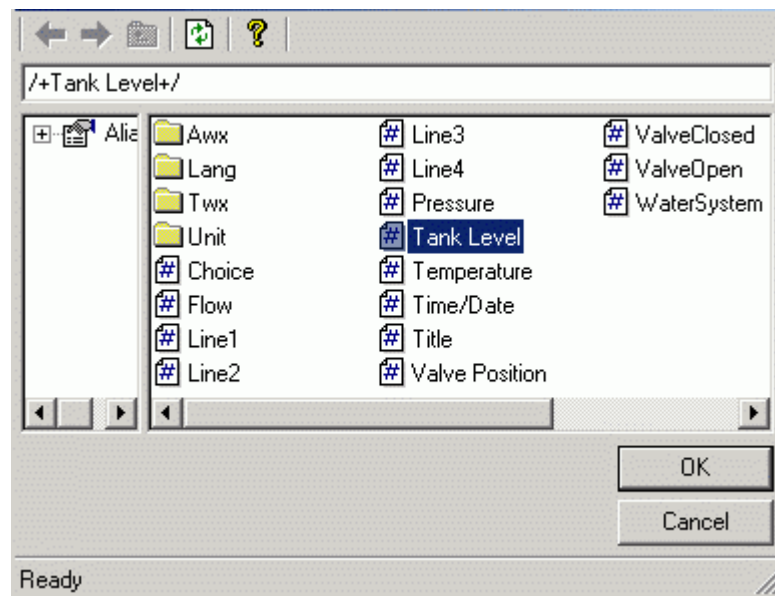
OPC Event Server Area/Source Browser

You can also select global aliases to use in the **Area** field. Clicking the ... button and selecting **Global Alias Browser** from the pop-up menu opens the Global Alias Browser, as 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.



Selecting an Alias From the Global Alias Browser

Clicking the ... button and selecting **Language Alias Browser** from the pop-up menu opens the Language Alias Browser, as shown in the figure below. The browser includes all languages aliases in the language database. All language aliases that are configured in the Language Configurator are conveniently available to choose from inside the browser. The tree control of the Language Configurator is mimicked in the tree control of the Language Alias Browser. Select a language alias by double-clicking the alias name. The alias name appears at the top of the browser, which automatically adds the /+ and +/ delimiters to the alias name. Click the **OK** button.



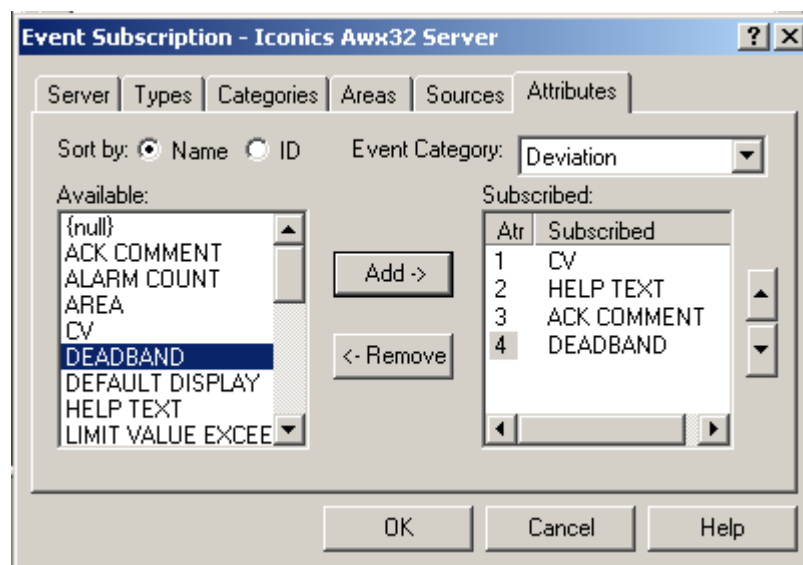
Selecting an Alias From the Language Alias Browser

Attributes

The **Attributes** tab of the **Event Subscription** dialog box, shown in the figure below, allows you to add extra attributes to a particular event category within a subscription. Select the event category from the corresponding drop-down list. Choose the desired items from the available attributes list, and click the **Add** button. To remove an attribute from the subscribed list, select that particular attribute and click **Remove**.

Note: The following categories are not used in the ICONICS Alarm Server:

- OPC Server Error
- System Configuration
- System Message



Event Subscription Dialog Box: Attributes Tab

It is important to note that the order of the subscribed attributes does matter. The order of the attributes determines the order they will be selected from the alarm server, and also determines with which viewer attribute column they are associated. To change the order of the attributes in the subscribed field, simply select an attribute and click the "up" and "down" arrow buttons. Server-specific information will be displayed in the Attributes1-n columns. The attributes columns are added in the **Column** tab of the **Alarm Viewer ActiveX Properties** dialog box.

Note: To receive extra attributes, you must request them per event category.

Event Attributes

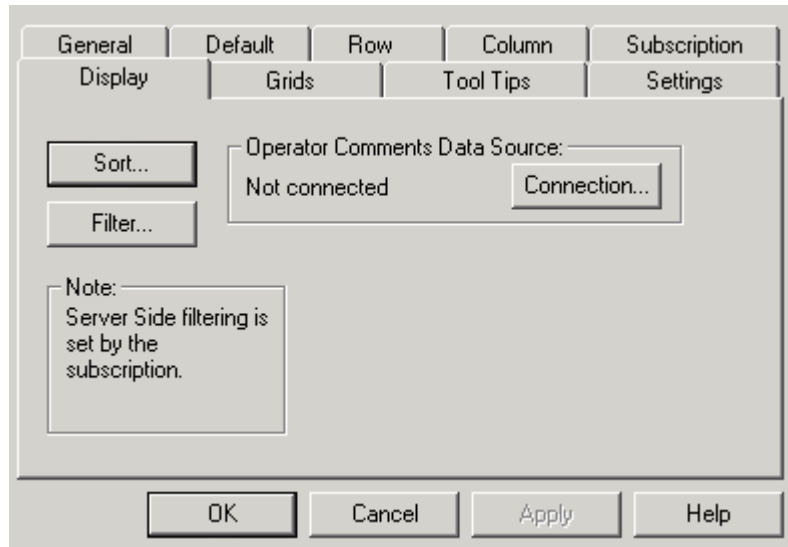
The following event attributes are made available to subscribing clients:

Attribute	Deviation	Digital	Limit	Rate of Change	Type	Comment
ACK COMMENT	✓	✓	✓	✓	VT_BSTR	Comment entered when alarm was last acknowledged.
ALARM COUNT	✓	✓	✓	✓	VT_I4	Number of alarm transitions before acknowledge.
AREA	✓	✓	✓	✓	VT_ARRAY VT_BSTR	Array of areas
CV	✓	✓	✓	✓	VT_R8 VT_BOOL	Current Value (OPC Input 1)
DEADBAND	✓		✓	✓	VT_R8	Alarm detection Deadband.
DEFAULT DISPLAY	✓	✓	✓	✓	VT_BSTR	A text string, usually a file name that when launched provides related information.
HELP TEXT	✓	✓	✓	✓	VT_BSTR	Help information
LIMIT VALUE EXCEEDED	✓		✓	✓	VT_R8	The Current Value at the time the limit was exceeded.
LOOP DESC	✓	✓	✓	✓	VT_BSTR	A description of the Source (Tag)
NEXT_LIM	✓		✓		VT_R8	The value of the next limit.
NORMAL STATE		✓			VT_BOOL	Value of the normal state (TRUE or FALSE).
PREV LIM	✓		✓		VT_R8	Value of the previous limit reached.
RELATED VALUE 01-10	✓	✓	✓	✓		Associates up to 10 OPC tags with each alarm to provide “supplemental” information to the alarm.
SP	✓				VT_SP	Setpoint (OPC Input 2)

Client Side Filtering

There are two ways to achieve filtering using the Alarm Viewer: on the server side and on the client side. **Server-side filtering** takes place through the use of subscriptions, while **client-side filtering** takes place through the use of the **Filter** option on the **Display** tab of the **Alarm Viewer ActiveX Properties** dialog box, as shown in the figure below.

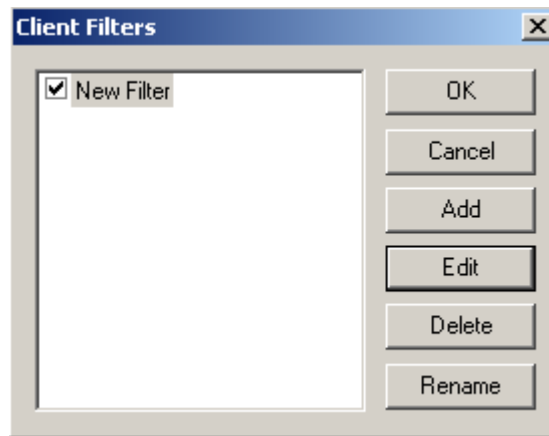
Note: When possible, server-side filtering should be the primary filtering method.



Alarm Viewer ActiveX Properties: Display Tab

Clicking the **Filter** button opens the **Client Filters** dialog box, as shown in the figure below.

Note: You can also open this dialog box by right-clicking the viewer during runtime and selecting **Filter** from the pop-up menu.

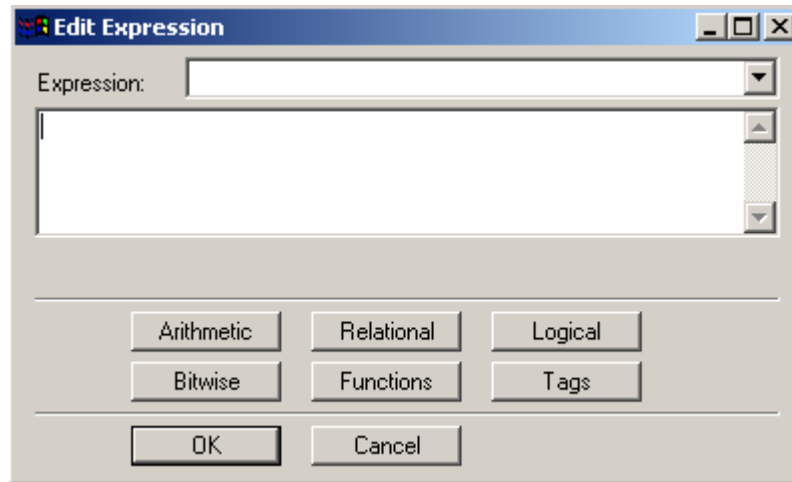


Creating Client-Side Filters

Adding and Editing Filters

To add a filter, click the **Add** button. A new filter will appear in the list box. To enter the criteria for the filter, select the filter and click the **Edit** button. This opens the **Expression Editor**, as shown in the figure below. The Expression Editor box has the following options: Arithmetic, Relational, Logical, Bitwise,

Functions, and Tags. Each of these options is explained in greater detail in following sections.



Filter Expression Configuration

Deleting Filters

To delete a filter all together, select the filter in the **Client Filters** dialog box and click the **Delete** button. Your system will ask if you really want to delete the filter; click **Yes** and the filter will be deleted.

Renaming Filters

To rename a filter, select the filter in the **Client Filters** dialog box and click the **Rename** button. You must change the filter name if you are going to add more than one filter. Try to give the filter a name that describes what it does or the values it considers when filtering.

Note: Depending on the security level of the user, filters may or may not be able to be created or activated during runtime.

Arithmetic
Relational
Logical
Bitwise
Functions
Tags
Grids
Settings

Arithmetic

The symbols '+', '-', '*', '/' and '%' use the following format:

expression :: parameter **symbol** parameter

Where

Parameter	A local variable, an OPC tag, a constant, or another expression
Symbol	+ or - or * or / or %

Result

The expression results in a number of any type (float, long, etc.).

Examples

Symbol	Description	Example	Result
	Addition	~~var1~~ + ~~var2~~	8+3 = 11
	Subtraction	~~var1~~ - ~~var2~~	8-3 = 5
	Multiplication	~~var1~~ * ~~var2~~	8*3 = 24
	Division	~~var1~~ / ~~var2~~	8/3 = 2.66667
	Calculates the remainder after division	~~var1~~ % ~~var2~~	8%3 = 2
	Gives precedence to parts of the calculation	~~var1~~ / (~~var2~~ + ~~var3~~)	8/(3+2) = 1.6

Relational

The symbols '<', '>', '<=', '>=', '==' and '!=' use the following format:

expression :: parameter **symbol** parameter

Where

Parameter	A local variable, an OPC tag, a constant, or another expression
------------------	---

	expression
Symbol	< or > or <= or >= or == or !=

Result

The expression results in a Boolean value (0 or 1).

Examples

Symbol	Description	Example	Result
<	Less than	~~var1~~ < ~~var2~~	8<3 = 0
>	Greater than	~~var1~~ > ~~var2~~	8>3 = 1
<=	Less than or equal to	~~var1~~ <= ~~var2~~	8<=3 = 0
>=	Greater than or equal to	~~var1~~ >= ~~var2~~	8>=3 = 1
==	Equal to	~~var1~~ == ~~var2~~	8==3 = 0
!=	Not equal to	~~var1~~ != ~~var2~~	8!=3 = 1

Logical

The symbols '&&' and '||' use the following format:

expression :: parameter **symbol** parameter

The symbol '!' uses the following format:

expression :: **symbol** parameter

Where

Parameter	A local variable, an OPC tag, a constant, or another expression
Symbol	&& or or !

Result

The expression results in a Boolean value (0 or 1).

Truth table

~~var1~~	0		not 0	
~~var2~~	0	not 0	0	not 0
~~var1~~ && ~~var2~~	0	0	0	1
~~var1~~ ~~var2~~	0	1	1	1
!~~var1~~	1	1	0	0

Examples

Symbol	Description	Example	Result
&&	And	~~var1~~ && ~~var2~~	8 && 3 = 1
	Or	~~var1~~ ~~var2~~	8 3 = 1
!	Not	!~~var1~~	!8 = 0

Bitwise

The symbols '&', '|', and '^' of the bitwise group use the following format:

expression :: parameter **symbol** parameter

The symbol '~' of the logical group uses the following format:

expression :: **symbol** parameter

The symbols 'shl' and 'shr' of the bitwise group use the following format:

expression :: **symbol** (value, shift by)

Where

Parameter	A local variable, an OPC tag, a constant, or another expression
Symbol	&& or or ^ or shl or shr or ~

Result

The expression results in a number when the parameters used contain numbers.

Bit Table

Binary (Decimal)	Binary (Decimal)
------------------	------------------

~~var1~~	0000.0000.0000.1000 (8)	0000.0000.0110.0000 (96)
~~var2~~	0000.0000.0000.1010 (10)	0000.0000.0000.1000 (8)
~~var1~~ & ~~var2~~	0000.0000.0000.1000 (8)	0000.0000.0000.0000 (0)
~~var1~~ ~~var2~~	0000.0000.0000.1010 (10)	0000.0000.0110.1000 (104)
~~var1~~ ^ ~~var2~~	0000.0000.0000.0010 (2)	0000.0000.0110.1000 (104)
shl (~~var1~~,3)	0000.0000.0100.0000 (64)	0000.0011.0000.0000 (768)
shr (~~var1~~,3)	0000.0000.0000.0001 (1)	0000.0000.0000.1100 (12)
~(~~var1~~)	1111.1111.1111.0111 (-9)	1111.1100.1111.1111 (-97)
bittest(~~var1~~,3)	0000.0000.0000.0001 (1)	0000.0000.0000.0000 (0)

Examples

Symbol	Description	Example	Result
&	Bit And	~~var1~~ & ~~var2~~	8 && 3 = 0
	Bit Or	~~var1~~ ~~var2~~	8 3 = 11
^	Bit eXclusive Or	~~var1~~ ^ ~~var2~~	8^3=11
shl	Bit shift left	shl(~~var1~~,3)	8<<3=64
shr	Bit shift right	shr(~~var1~~,3)	8>>3=1
~	Not (two's complement)	~(~~var1~~)	!8 = -9
bittest	Bit Test	bittest (5 , 0)	1

Note: The bittest function requires you to specify the position of the bit to be tested. You must indicate that it starts from 0. In other words, a bit position of "0" indicates the "less significant" bit.

Functions

The symbols 'sin', 'asin', 'cos', 'acos', 'tan', 'atan', 'log', 'ln', 'exp', 'sqrt', 'abs', 'ceil', and 'floor' use the following format:

expression :: **symbol** (parameter)

The symbols 'pow', 'min', and 'max' use the following format:

expression :: **symbol** (parameter,parameter)

The symbol 'if' uses the following format:

expression :: **symbol** (parameter,parameter,parameter)

Where

Parameter	A local variable, an OPC tag, a constant, or another expression
Symbol	sin, asin, cos, acos, tan, atan, log, ln, exp, sqrt, abs, ceil, floor, min, max, pow, or if

Result

The expression results in a number.

Examples

Symbol	Description	Example	Result
sin	sine of an angle in radians	sin(~var1~)	sin(0.785)=0.71
cos	cosine of an angle in radians	cos(~var1~)	cos(0.785)=0.71
tan	tangent of an angle in radians	tan(~var1~)	tan(0.785)=1.0
asin	arc sine returns an angle in radians	asin(~var1~)	asin(0.5)=0.52
acos	arc cosine returns an angle in radians	acos(~var1~)	acos(0.5)=1.05
atan	arc tangent returns an angle in radians	atan(~var1~)	atan(1)=0.785
sqrt	Returns the square root	sqrt(~var1~)	sqrt(100)=10
pow	Returns value 1 raised to the power value 2	pow(~var1~,~var2~)	pow(100,1.5)=1000
log	10 based	log(~var1~)	log(100)=2

Symbol	Description	Example	Result
	logarithm		
ln	e based logarithm	ln(~var1~)	ln(7.389)=2
exp	Exponential	exp(~var1~)	exp(2)=7.389
abs	Absolute value	abs(~var1~)	abs(-1)=1
ceil	Integer ceiling	ceil(~var1~)	ceil(7.39)=8
floor	Integer floor	floor(~var1~)	floor(7.39)=7
min	Lowest value of two	min(~var1~,~var2~)	min(10,5)=5
max	Highest value of two	max(~var1~,~var2~)	min(10,5)=10
if	Conditional statement	if(~var1~<~var2~,~var1~,~var2~)	if(5<8,5,8)=5
like	Wildcard string compare	Like(string, pattern, casesensitive')	
quality	Quality of tag or expression	See below.	See below.
tostring	Type conversion	See below.	See below.
0x	Hexadecimal constant	x=0x11	17
0t	Octal constant	x=0t11	9
0b	Binary constant	x=0b11	3

Note: For the like operator: "string" equals the string to search in; "pattern" equals the string to search for (can include wildcards); nonzero for case-sensitive search; zero for case-insensitive search. String syntax is \$"string"\$.

You can use these special characters in pattern matches in string:

- ? Any single character.
- Zero or more characters.
- # Any single digit (0-9).
- [charlist] Any single character in charlist.
- [!charlist] Any single character not in charlist.

Quality

The **quality** option on the **Functions** menu of the **Expression Editor** is used to evaluate the quality of an OPC tag or an expression.

The following general syntax is used for quality expressions:

x=quality(expression)

Note: The "(expression)" can also be a simple expression composed of a single tag.

The **quality** function returns the OPC quality of the string between parentheses as one of the following results:

- 192: quality is GOOD
- 64: quality UNCERTAIN
- 0: quality BAD

Note: The OPC Foundation establishes the value ranges for quality. There are actually varying degrees of quality:

- GOOD: 192-252
- UNCERTAIN: 64-191
- BAD: 0-63

For more information, refer to the *OPC Data Access Custom Interface Standard* available for download at the OPC Foundation's Web site, www.opcfoundation.org/.

Example Quality Expression

Expression	Result
x=quality({{ICONICS.Simulator.1\SimulatePLC.PumpStatus}})	192 (Quality GOOD)

The quality of an expression is determined through the evaluation of each single tag in the expression. Thus, if you have multiple tags in an expression (and each tag has a different quality), the result of the expression (i.e. 192 [GOOD], 64 [BAD], or 0 [UNCERTAIN]) corresponds to the quality of the tag with the lowest quality. If an expression contains a conditional statement (e.g. if, then, or else), then the result of the expression is affected only by the quality of the branch being executed.

Consider the following sample expression:

x= if (quality({{Tag1}}) == 192, {{Tag1}}, {{Tag2}})

This expression can be read as follows:

"If the quality of Tag1 is GOOD (i.e. 192), then the expression result (x) is the value of Tag1. In all other cases (i.e. the quality of Tag1 is UNCERTAIN or BAD), the expression result (x) is the value of Tag2."

We can calculate the results for this expression using different qualities for Tag1 and Tag2, as shown in the table below.

Case	Tag1 quality	Tag2 quality	Result	Result quality
1	GOOD	GOOD	Tag1	192 (GOOD)
2	GOOD	UNCERTAIN	Tag1	192 (GOOD)
3	GOOD	BAD	Tag1	192 (GOOD)
4	UNCERTAIN	GOOD	Tag2	192 (GOOD)
5	UNCERTAIN	UNCERTAIN	Tag2	64 (UNCERTAIN)
6	UNCERTAIN	BAD	Tag2	0 (BAD)
7	BAD	GOOD	Tag2	192 (GOOD)
8	BAD	UNCERTAIN	Tag2	64 (UNCERTAIN)
9	BAD	BAD	Tag2	0 (BAD)

In cases 1-3 above, the quality of Tag1 is GOOD, and therefore the result of the expression is GOOD. Thus, the result of the expression is not affected by the quality of Tag2 (the "else" branch of the expression), which is ignored.

In cases 4-6, the quality of Tag1 is UNCERTAIN, and therefore the result of the expression is the quality of Tag2.

In cases 7-9, the quality of Tag1 is BAD, and therefore the result of the expression is the quality of Tag2.

Note: The "quality()" function returns a value that represents the quality of the expression within the parentheses but is always GOOD_QUALITY. For example, if Tag1 is BAD_QUALITY then the expression "x=quality({{Tag1}})" will return 0 with GOOD_QUALITY.

The result of an expression is the minimum quality of the evaluated tag in the expression and is affected only by the quality of the conditional (if, then, or else) branch that is executed.

Consider the following sample expression:

x= if ({{TAG_01}}>0,{{TAG_02}},{{TAG_03}})

This expression can be read as follows:

"If the value of TAG_01 is greater than 0, then the expression result (x) is TAG_02. If the value of TAG_01 is less than or equal to 0, then the expression result (x) is TAG_03."

Let's assume that the following values and qualities for these tags:

TAG_01=5 with quality GOOD

TAG_02=6 with quality UNCERTAIN

TAG_03=7 with quality BAD

Because the value of TAG_01 is 5 (greater than 0), the expression result is TAG_02. Thus, the final expression result is 6, and the final expression quality is UNCERTAIN.

Type Conversion

The **tostring** option on the **Functions** menu of the **Expression Editor** takes the value of whatever item is in parentheses and converts it into a string as follows:

The value is +(value)+unit

It can be used to convert from number to string, and it can be very useful for string concatenation.

The proper syntax for the **tostring** option is:

x="\$The value is "\$ + tostring(value) + \$" unit"\$

Note: In the expression above, the word "unit" is placeholder text for a user-specified unit of measurement or variable (e.g. Watt, inches, meters, etc.).

Example Expressions Type Conversion

Expression	Result
x="\$The value is "\$ + tostring({{gfwsim.ramp.float}}) + \$" Watt"\$	"The value is 543.2345152 Watt"

Constants

The **Functions** menu of the **Expression Editor** supports constant values, including hexadecimal, octal, and binary formats.

Example Expressions Using Constants

Expression	Result
------------	--------

x=0x11	17
x=0t11	9
x=0b11	3

The **Expression Editor** conveniently inserts the 0x and 0t and 0b prefixes for you so do not have to recall them.

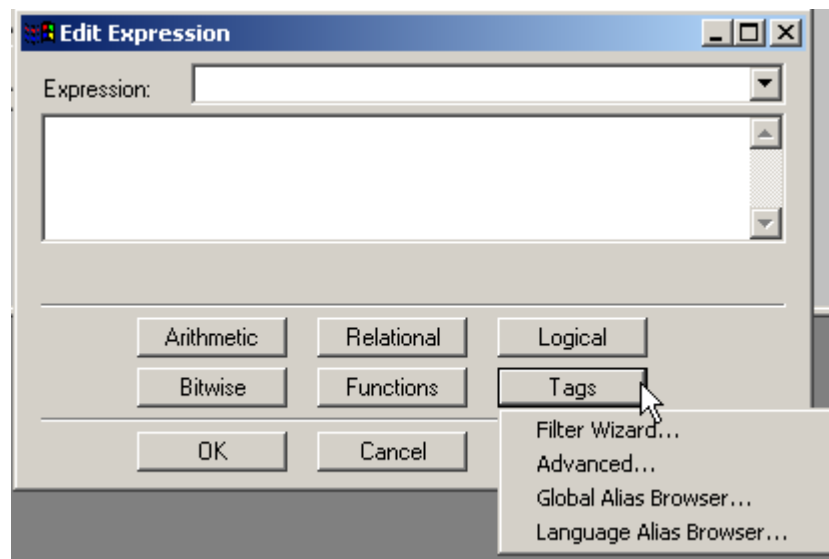
Interpreting and Translating Constants

The examples below show how values are calculated for each type of constant.

- **Hexadecimal:** $0x20A = 2 * (16^2) + 0 * (16^1) + 10 * (16^0) = 2*256 + 0*16 + 10 * 1 = 512 + 0 + 10 = 522$
- **Octal:** $0t36 = 3 * (7^1) + 6 * (7^0) = 3* 7 + 6* 1 = 21 + 6 = 27$
- **Binary:** $0b110 = 1 * (2^2) + 1 * (2^1) + 0 * (2^0) = 1 * 4 + 1 * 2 + 0 * 1 = 4+2+0 = 6$

Tags

The menu options available under the **Tags** button of the Expression Editor are shown in the figure below. The **Expression Editor** dialog box can also be used to create and edit alarm filters. The Expression Editor provides a **Filter Wizard** and an **Alarm Tag** list to help you create simple alarm filters. If you want to customize your alarm filters, you can use the other functions in the Expression Editor to set up your alarm filters manually.

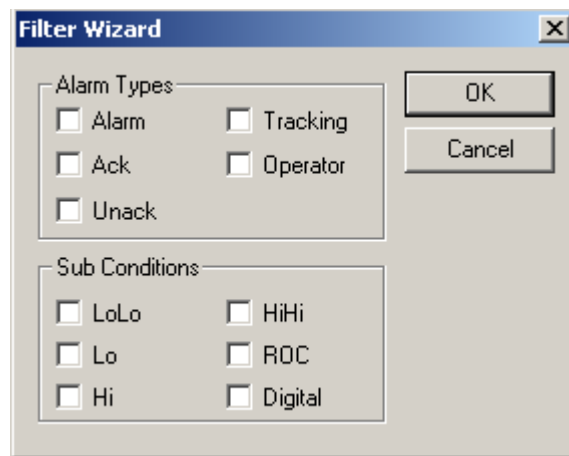


Creating Alarm Filters Using the Expression Editor

Filter Wizard

The **Filter Wizard**, shown in the figure below, allows you to choose from the following to items enter in your expression. Select one or more items, and then click **OK**. The filter string is automatically inserted into the **Edit Expression** dialog box.

- **Alarm Types:** Alarm, Acknowledgement, Unacknowledgement, Tracking, and Operator
- **Subconditions:** LoLo, Lo, Hi, HiHi, Rate of Change, and Digital



Filter Wizard

Selecting Alarm Attributes

Selecting **Advanced** from the **Tags** menu of the Expression Editor opens the **Alarm Tag** list, shown in the figure below, which allows you to choose alarm attributes for your alarm filter. Select the attribute that you want to include in the filter expression and click **OK**.



Alarm Attributes List

There are two additional attributes available for use in filtering: **Alarm Type** and **Current Time**. The Alarm Type attribute allows you to filter alarms according to

ALARM 1, ACK 2, UNACK 3, OPER 4, TRACK 5 or NORM 6. For example, you can set up a filter with the condition:

$X = \{\{\text{AlarmType}\}\}$

If the **Alarm Type** is true, then the alarms are displayed. If they are false then, the alarms are not displayed.

The **Current Time** attribute allows you to filter according to the current time. Only alarms occurring around the current time will be displayed.

Example Alarm Filters

Expression	Result
$X = \{\{\text{Severity}\}\} > 500.$	Only alarm messages with a severity greater than 500 will be visible.
$X = \text{Like}(\{\{\text{Source}\}\}, \$\text{"Tag"}\$,0)$	Only messages with the tag in the source name will be displayed.
$X = 1.$	Filter displays all messages.
$X = 0.$	Filter does not display any messages.

All filters resolve to TRUE or FALSE. All nonzero values resolve to TRUE.

Global Aliases

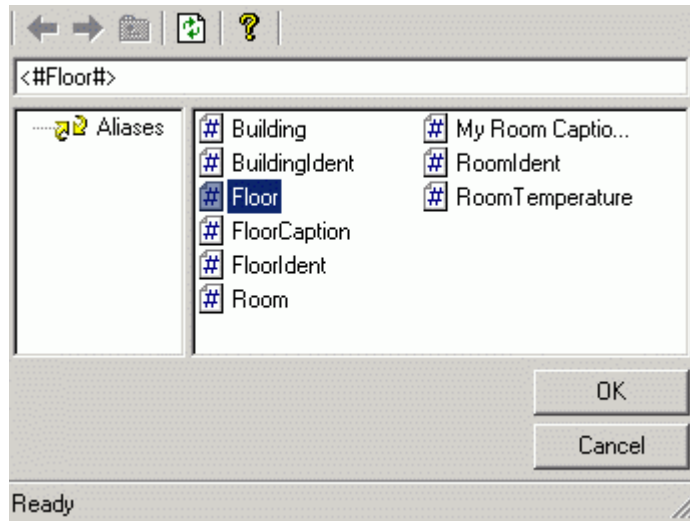
For global aliases within the expression, use the following syntax:

$\text{<\#global_alias_name\#>}$

Example:

$x = \text{<\#RoomTemperature\#>}$

Selecting **Global Alias Browser** opens the Global Alias Browser, as 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 $\text{<\#}$ and $\text{\#>}$ delimiters to the alias name. Click the **OK** button.



Selecting an Alias From the Global Alias Browser

Language Aliases

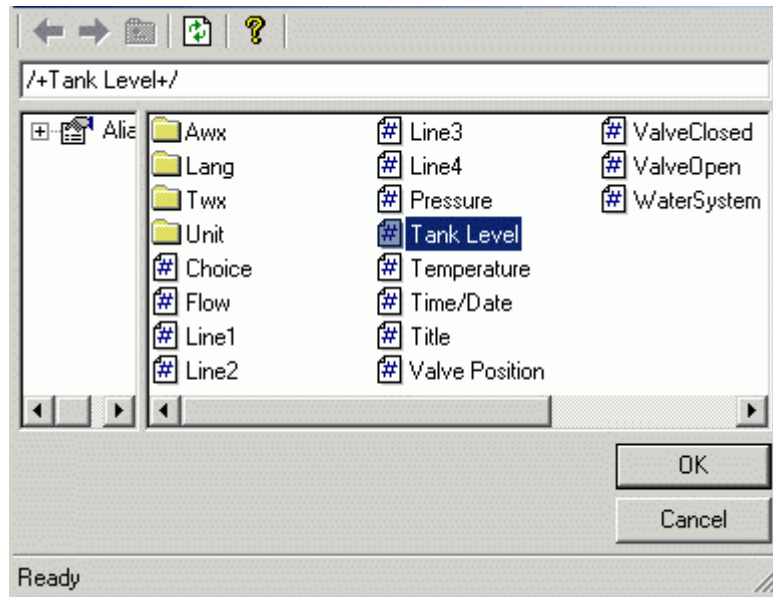
For language aliases within the expression, use the following syntax:

`/+language_alias_name+/-`

Example:

`x=/+WaterSystem+/-`

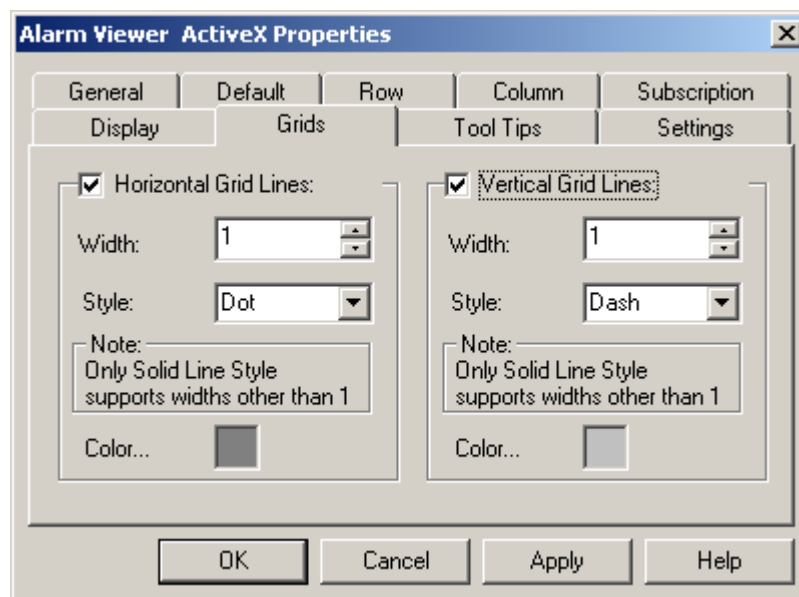
Selecting **Language Alias Browser** from the pop-up menu opens the Language Alias Browser, as shown in the figure below. The browser includes all languages aliases in the language database. All language aliases that are configured in the Language Configurator are conveniently available to choose from inside the browser. The tree control of the Language Configurator is mimicked in the tree control of the Language Alias Browser. Select a language alias by double-clicking the alias name. The alias name appears at the top of the browser, which automatically adds the `/+` and `+/-` delimiters to the alias name. Click the **OK** button.



Selecting an Alias From the Language Alias Browser

Grids

The **Grids** tab of the **Alarm Viewer ActiveX Properties** dialog box, shown below, configures the appearance of the grid lines that appear on the Alarm Viewer.



Alarm Viewer ActiveX Properties: Grids Tab

Configure the following grid settings:

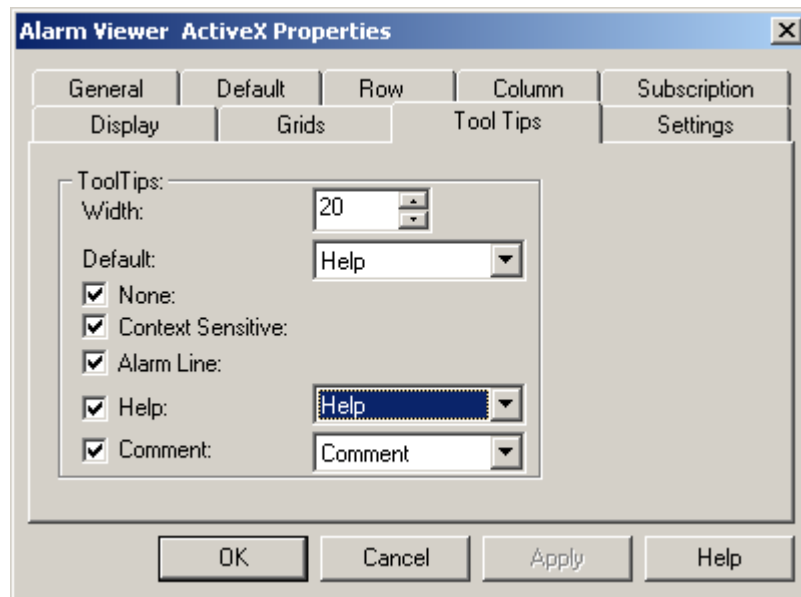
- **Colors:** Specifies colors for the grid text, headers, and background.

- **Grid Lines:** Both **Horizontal** and **Vertical** grid lines appear on the alarm report. You can specify the **Width, Style** (e.g. Solid, Dashed, etc.), and **Color** of the grid lines. To change the color of the grid lines, simply click the **Color** check box and select a color from the color palette.

Note: It is not necessary to have grid lines, but it is helpful in distinguishing between different records.

ToolTips

The **ToolTips** tab of the **Alarm Viewer ActiveX Properties** dialog box, shown below, allows you to determine which ToolTip options will be available during runtime by selecting the corresponding check boxes.



Alarm Viewer ActiveX Properties: ToolTips Tab

The **Width** field determines the width of the ToolTip when it is shown in the display.

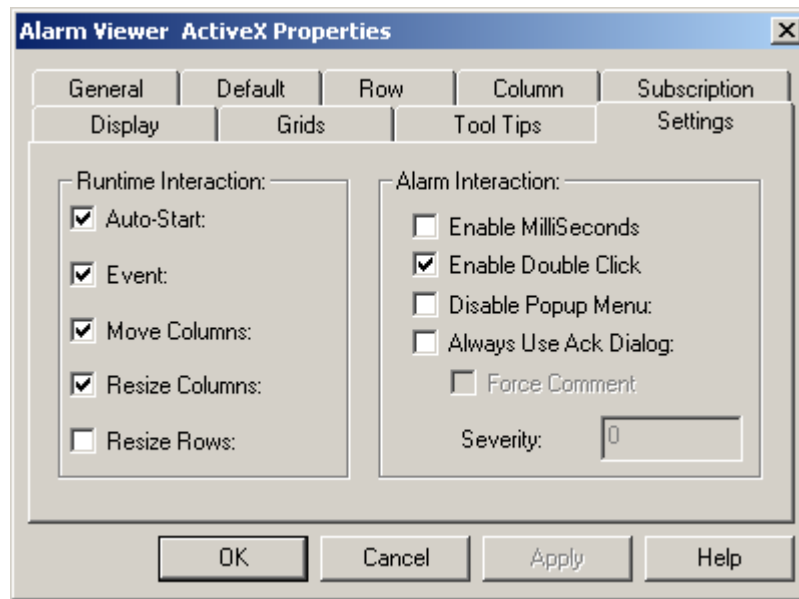
The **Default** field allows you to select the following ToolTips that will be the default in runtime:

- **None:** No ToolTips.
- **Context:** Context information ToolTips.
- **Help:** Displays the information associated with the column configured for help. This is set in the **Settings** tab of the **Alarm Viewer ActiveX Properties** dialog box.
- **Comment:** Displays the information associated with the column configured for comment. This is set in the **Settings** tab of the **Alarm Viewer ActiveX Properties** dialog box.
- **Alarm Line:** Displays the entire alarm line.

Both the **Help** and **Comment** fields have corresponding drop-down list boxes that determine which columns will be displayed for these selections.

Settings

The **Settings** tab of the **Alarm Viewer ActiveX Properties** dialog box, shown below, configures how runtime will be started and which functions will be enabled once it has. It also contains settings for alarm interactions.



Alarm Viewer ActiveX Properties: Settings Tab

Checking the **Auto-Start** check box triggers the server to start runtime when the container is activated. Checking the **Event** check box enables the automation event trigger (VB event) during runtime. The 'Always Use Ack Dialog' forces the user to acknowledge all alarms through the Acknowledge Alarm dialog.

Runtime Interaction

Checking the **Auto-Start** check box triggers the server to start runtime when the container is activated.

Checking the **Event** check box enables the automation event trigger (VB event) during runtime.

Checking the **Move Columns** check box enables you to change the order of the columns in runtime. Selecting a column header and moving it to the new location achieves this in runtime. Additionally, you can decide if columns and rows can be resized while in runtime mode by checking **Resize Columns** or **Resize Rows**.

Alarm Interaction

When **Enable Milliseconds** is checked, milliseconds will be displayed after time in the Time/Date column of the viewer during runtime mode.

When **Enable Double Click** is checked, double-clicking an alarm during runtime acknowledges only the row double-clicked if the **CTRL** key is not pressed down.

When **Disable Popup Menu** is checked, the right-click menu acknowledgement options will not be available during runtime mode.

When **Always Use Ack Dialog** is checked, the user is forced to acknowledge all alarms through the **Acknowledge Alarm** dialog. When **Force Comment** is checked and an alarm is acknowledged, the alarm cannot be acknowledged until a comment is entered in the **Comments** field of the **Acknowledge Alarm** dialog. The **Force Comment** function applies to alarms with a severity level that is greater than or equal to the **Severity** setting.

Runtime Operations

During runtime mode, the AlarmWorX32 Viewer ActiveX provides support for the following functions:

- Alarm acknowledgement
- Operator comments
- Sorting
- ToolTips
- VB event
- Application
- Display

As you can see in the figure below, these functions become available when you right-click on an alarm line during runtime mode. The selected alarm has a 3-D look so you can distinguish it from the other alarm lines, as shown in the figure below.

Note: You can also left mouse click and drag the cursor to select and acknowledge multiple alarm lines.

Time	Condition Name	Quality
12:02:00 PM 10/	Limit	Good - Non-Sp
12:02:00 PM 10/	Limit	Good - Non-Sp
12:02:00 PM 10/	Limit	Good - Non-Sp
10:50:32 PM 10/		Good - Non-Sp
10:50:37 PM 10/		Good - Non-Sp
10:50:38 PM 10/		Good - Non-Sp

Ack
Comments
Sort
Filter
Tooltips
VB Event

None
Context
☒ Help
Comment
Alarm Line

Runtime Operations

Note: When the scroll bar is moved from the top position, the Alarm Viewer enters a "freeze" mode. This mode will enable a user to scroll without having the alarm sorting interfere with reading the alarm viewer. All alarms that existed before the user entered freeze mode are displayed. This includes any change of state for those alarms. If one of these alarms enters a normal state while the viewer is in freeze mode, a placeholder line will be used to maintain visual placement. All alarms that did not exist before freeze mode are processed for beep, header flash, and automation support, but are not displayed visually. In freeze mode, the header of the alarm viewer will flash if any alarm is not visible.

Alarm Acknowledgement
Double Acknowledging an Alarm
Sorting
ToolTips
VB Event
Display Option
Application

Alarm Acknowledgement

When an alarm is sent from the server to the viewer, it will remain there until it is acknowledged in some way. The process of acknowledgement involves communication between the Alarm Viewer ActiveX and the subscribed Alarm OPC Server from which the alarm signals are coming. The initial communication results in the alarm signal, which appears in the Current Alarms Viewer. Once you acknowledge an alarm, an **ack** message is sent to the appropriate server. The server then posts the new state of the alarm (i.e. "acknowledged") to the viewer. The alarm line is then updated with the new

information. Events are removed during the acknowledgement process if included in the list of items to be acknowledged. The Alarm and Event servers do not receive communication for event removal. There are several ways in which an alarm or event can be acknowledged.

Note: The alarm header will flash whenever there is an unacknowledged alarm that is not visible in the window.

Point Acknowledgement

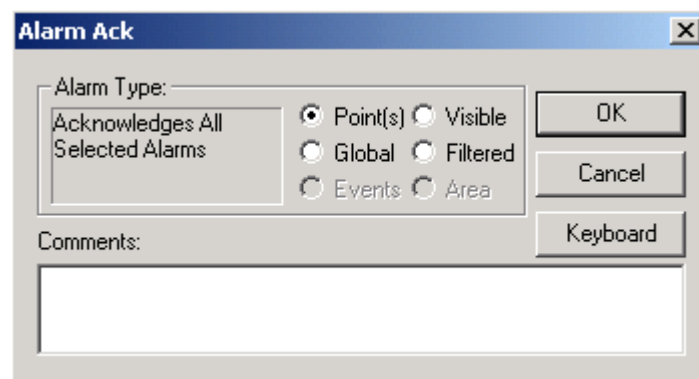
Point acknowledgement will acknowledge all alarms currently selected. To select and acknowledge more than one alarm, hold the control key down and left mouse click on all rows of interest, right mouse click, select point, and then click **OK**. A left double mouse click will also acknowledge all selected points if the control key is held down.

Note: The row that is under the mouse on a right-click is selected if no other rows are currently selected.

Double-clicking an alarm during runtime (if this option is selected in the **Settings** tab of the **Alarm Viewer Properties** dialog box) acknowledges only the row double-clicked on if the **CTRL** key is not pressed down.

Right-clicking and selecting **Ack** during runtime opens the **Alarm Ack** dialog box, as shown in the figure below.

Note: You can also left mouse click and drag the cursor to select and acknowledge multiple alarm lines.



Acknowledging Alarms

The **Alarm Type** field allows you to see which type of alarm you are acknowledging. Clicking the **OK** button acknowledges a particular alarm.

Note: The **Global**, **Visible**, **Filtered**, and **Area** acknowledgements are described below.

If a different user acknowledges the alarm, the alarm will appear to automatically acknowledge.

Clicking the **Keyboard** button opens the keypad, which allows operators to key in information directly on the screen.

Global Acknowledgement

When you choose **Global** from the **Alarm Ack** dialog, you are acknowledging all alarms received up to that point. An advantage of using this option is that it quickly clears all alarms from the current view.

Visible Acknowledgement

The **Visible** option allows the user to acknowledge all visible alarms. Thus, if the size of the viewer allows for five alarms to be seen and a total of eight alarms are coming in, only the five visible alarms will be acknowledged. If there are more alarms than the visible ones, a scroll bar will appear on the right-hand side of the viewer.

Filtered Acknowledgement

The **Filtered** option allows the user to acknowledge all filtered alarms. This differs from global acknowledgement in the fact that filtered acknowledgement does not acknowledge locally filtered alarms.

Note: Filtered acknowledgement is only available in version 6.0 and later. In earlier versions, this option behaves the same as global acknowledgement.

Area Acknowledgement

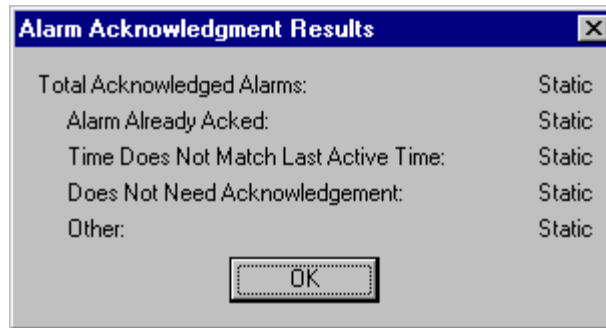
The **Area** option allows the user to acknowledge all alarms that have the same area as the alarm that was selected.

Event Removal

Selecting **Events** on the Alarm Acknowledgement dialog box removes all event messages from the viewer.

Double Acknowledging an Alarm

It is not possible to acknowledge an alarm more than once, but since the alarm does not immediately disappear from the viewer once it is acknowledged, it is possible for a user to try to acknowledge an alarm twice. When this occurs, the user will see the results dialog box shown below.



Alarm Acknowledgement Results

Total Acknowledged Alarms: Provides information about the number of alarms acknowledged.

Alarm Already Acked: Counts the number of alarms attempted to be acknowledged but were already acknowledged. This can occur if the alarm is acknowledged by another client before the acknowledge message is received by the client.

Time Does Not Match Last Active Time: The number of alarms that were attempted to be acknowledged but did not contain the most recent alarm timestamp. This can occur if the alarm being acknowledged is not the most recent state of the alarm. This will occur if the acknowledge message is sent before an update of the current alarm state is received by the client.

Does Not Need Acknowledgement: The number of messages that did not require acknowledgement. This can occur if an acknowledgement is attempted on a non-condition message.

Other: The number of messages that produced a non-OPC alarm acknowledgement COM error. Persistent occurrences of this problem could be due to a bad network connection or erroneous alarm message.

The Alarm Viewer will detect the current state of an alarm and will not attempt to acknowledge an alarm that is already marked as acknowledged. No dialog will be posted for this occurrence.

To clear the message, simply click the **OK** button.

Operator Comments

Selecting **Comments** from the Runtime menu opens the **Operator Comments** dialog box, as shown in the figure below. This dialog describes the alarm line the user selected, and it provides a method for entering one or multiple operator comments in a text field. It also indicates whether the user is currently connected to the operator comments database. Clicking the **Add** button logs these comments to a database using the Event Logger. The dialog displays all past operator comments for the alarm selected under the Time, Comment, and Operator columns. The Event Logger is mandatory for operator comments.

EventTime	UserComment	Operator
10/17/2002 4:30:41 ...	a much shorter message	
11/15/2002 1:15:13 ...	testing the comments	
10/17/2002 3:56:17 ...	tempurature	
10/9/2002 6:48:51 PM	testing the logger	
10/9/2002 6:50:38 PM	added a comment	
10/9/2002 6:50:50 PM	adding a second comment	
10/10/2002 4:25:19 ...	testing a second message	
10/10/2002 4:27:32 ...	another message for testing	

Database:	Connected			Add	Update
Server Node:	Local	Server Name:	ICONICS.AlarmServer.1		
Source:	Temperature	Condition:	Rate of Change		
Message:	Rate of change too big				

OK

Entering Operator Comments

Sorting

Selecting **Sorting** from the Runtime menu opens the **Sort** dialog box, as shown in the figure below.

Sort		
Sort Alarms by:	Time / Date	☐ Ascending ☒ Descending
Then by:	(none)	☒ Ascending ☐ Descending
Then by:	(none)	☒ Ascending ☐ Descending
Then by:	(none)	☒ Ascending ☐ Descending
Then by:	(none)	☒ Ascending ☐ Descending

OK
 Cancel
 Clear All

Sorting Alarms

The **Sort Alarms by** section indicates the first criteria the Alarm Viewer should use when sorting all current alarms. The following **Then by** sections indicate how the viewer should sort alarms that have the same value for the criteria selected in the first section.

Note: It is not necessary to have any values listed in the **Then by** sections.

The **Sort** dialog allows you to configure which criteria to sort the alarms by. It also allows for multilevel sorting up to five levels with ascending/descending specific to each level. The Alarm Viewer will always sort. The default sort for a new viewer is **Time/Date**.

Sort Ascending sorts the data in the column in ascending order (i.e. from least to greatest).

Sort Descending sorts the data in the column in ascending order (i.e. from greatest to least).

Note: The **Sort** dialog, including any configuration, is also available in runtime.

ToolTips

The **Default** of the ToolTips tab of the Alarm Viewer ActiveX Properties dialog box determines which of the following ToolTips that will be the default in runtime:

- **None:** No ToolTips.
- **Context:** Context information ToolTips.
- **Help:** Displays the information associated with the column configured for help. This is set in the **Settings** tab of the **Alarm Viewer ActiveX Properties** dialog box.
- **Comment:** Displays the information associated with the column configured for comment. This is set in the **Settings** tab of the **Alarm Viewer ActiveX Properties** dialog box.
- **Alarm Line:** Displays the entire alarm line.

VB Event

This option allows a user to trigger an automation event during runtime mode. Using this event requires VBA programming tied to the user event.

Display Option

This option only appears if the Alarm Viewer creates the display list by checking the extra attributes field for the registered file types. This attribute is server-specific (for example, when using the ICONICS Alarm OPC Server, select **Default Display**). All displays configured for a particular alarm will be displayed. Selecting one, by clicking on the attribute cell, launches the display.

Application

This option only appears if the user selects the extra attribute associated with it. This attribute is server-specific (for example, when using the ICONICS Alarm OPC Server, select **Default Display**). This allows the user to launch applications (i.e. *.exe, *.com or *.bat files) by clicking on the corresponding attribute cell.



Your comments and suggestions on the operation of this software are welcome. Please address them to:

ICONICS

100 Foxborough Blvd.

Foxborough, MA 02035

Tel: 508-543-8600

Fax: 508-543-1503

E-mail: support@iconics.com

Web: www.iconics.com

© ICONICS, Inc. All rights reserved.

Contents

Introduction

Automation Interfaces

Available Control Properties

Available Control Methods

Available Control Events

Introduction

This reference describes the OLE Automation features available in the AlarmWorX32 Viewer ActiveX.

Automation Interfaces

The AlarmWorX32 Viewer ActiveX provides a COM interface that allows automation interfaces run from within the ActiveX container to manipulate the Viewer as it is running. The interface is available to all programming languages that support COM including Microsoft Visual Basic (VB), Visual Basic for Applications (VBA) and Microsoft Visual C++.

To access the automation interface from VB and VBA, the AlarmWorX32 Viewer must be made available by choosing **Project - Components** from the main menu in the VB or VBA development environment and selecting **AWXVIEW32 ActiveX** in the list of available components.

Available Control Properties

boolean Check

Description

Enable/Disable URL file support.

Remarks

If check is True URL file support is enabled. FileEnabled must also be True. URL file support is only available for loading.

Example

```
AWXVIEW32OCX1.Check = True
```

boolean FileEnabled

Description

Enable/Disable file storage

Remarks

If FileEnabled is True, file support is enabled. File support consists of loading and saving configuration information to a file other than the container document. It is recommended that the LoadSaveFile Method be used instead.

Example

```
AWXVIEW32OCX1.FileEnabled = True
```

BSTR Caption

Description

String used for Title

Remarks

This property has no function at this time.

BSTR Filename

Description

Name of the file to load/save

Remarks

If FileEnabled is True, file support is enabled. The Filename is the name of the file and the path of the file to load and save configuration information to. File names must have an .awv extension. It is recommended that the LoadSaveFile Method be used instead.

Example

Sets the file name

```
AWXVIEW320CX1.Filename = "C:\test.awv"
```

Gets the file name

```
Dim Name as String  
Name = AWXVIEW320CX1.Filename
```

BSTR UrlPath

Description

Name of the URL path and file to load from

Remarks

The UrlPath is the name of the file and the path of the file to load configuration information from. File names must have an .awv extension. Please note that configuration information cannot be saved to a URL file.

Example

Sets the URL file name

```
AWXVIEW320CX1.UrlPath = "http:\12.13.150.2\test.awv"
```

Gets the URL file name

```
Dim Name as String  
Name = AWXVIEW320CX1.UrlPath
```

boolean AutoSize

Description

Enable/Disable AutoSize as default sizing method.

Remarks

AutoSize determines the base size of a column based on the size of the header text of that column.

Example

```
AWXVIEW320CX1.AutoSize = True
```

boolean DispRow

Description

Enable/Disable displaying row header

Remarks

Displaying the row header enables an operator to change the row width size during runtime. This property can be used with the NumRowCheck property to provide a numerical value for each row.

Example

```
AWXVIEW320CX1.DispRow = True
```

boolean ManSize

Description

Enable/Disable Manual Size as default size

Remarks

Enabling this property will prevent the viewer from altering the sizes of any columns except for the last column. Use this property if you wish to manually set base column sizes.

Example

```
AWXVIEW320CX1.AutoSize = False
```

boolean NumRowCheck

Description

Enable/Disable numbering row header. Only enabled when DispRow is Enabled.

Remarks

Use this property in conjunction with the DispRow property to display a number for each alarm line.

Example

Example displays the number of each alarm line

```
AWXVIEW320CX1.DispRow = True  
AWXVIEW320CX1.NumRowCheck = True
```

boolean Scale

Description

N/A

Remarks

This property has no function at this time.

boolean Title

Description

N/A

Remarks

This property has no function at this time.

boolean WrapText

Description

Enable/Disable text wrapping. Only visible when row size is greater than 1.

Remarks

Enabling this property causes the all column text to wrap to the next line. The wrap effect is contained within each individual column cell. Please note that this effect is only visible if the RowSize property is greater than 1 and the text information is larger than one column length.

Example

This example sets the Column height to 2 and enables text wrapping.

```
AWXVIEW320CX1.WrapText = True  
AWXVIEW320CX1.RowSize = 2
```

long MaxLength

Description

Sets default cell length.

Remarks

This property will set all columns in the viewer to the sizes specified (excluding the last column). Use this property if you wish to set all base column sizes to the same size.

Example

```
AWXVIEW320CX1.MaxLength = 10
```

JUSTIFY BaseJustify

Description

Sets default justification.

Remarks

This property sets the base justification for each column. It makes use of the enumerated values JustifyCenter, JustifyLeft, JustifyRight. The second half of the enumerated value name denotes the type of justification. For example JustifyCenter would center the text for each column.

Example

```
AWXVIEW320CX1.BaseJustify = JustifyCenter
```

long RowSize

Description

Sets base Row height.

Remarks

This property directly effect how much of the screen space is taken up for each row.

Example

This example sets the Column height to 2.

```
AWXVIEW320CX1.RowSize = 2
```

OLE_COLOR BackColor

Description

Sets the default background color

Remarks

This color is used for the base background color of each row (excluding row and column headers).

RGB defines colors in the order of red, green, blue. A good trick to use when picking colors is to open a graphics package and use the color palate to determine the RGB numbers. This will allow a visual representation of the color and provide the exact corresponding RGB values.

Example

```
AWXVIEW320CX1.BackColor = RGB(255, 255, 255)
```

OLE_COLOR HeaderColor

Description

Sets the default Header color

Remarks

RGB defines colors in the order of red, green, blue. A good trick to use when picking colors is to open a graphics package and use the color palate to determine the RGB numbers. This will allow a visual representation of the color and provide the exact corresponding RGB values.

Example

```
AWXVIEW320CX1.HeaderColor = RGB(255, 255, 255)
```

OLE_COLOR TextColor

Description

Sets the default text color

Remarks

RGB defines colors in the order of red, green, blue. A good trick to use when picking colors is to open a graphics package and use the color palate to determine the RGB numbers. This will allow a visual representation of the color and provide the exact corresponding RGB values.

Example

```
AWXVIEW320CX1.TextColor = RGB(255, 255, 255)
```

OLE_COLOR GridColor

Description

Sets the default background color

Remarks

This color is used for all grid space not covered by a row.

RGB defines colors in the order of red, green, blue. A good trick to use when picking colors is to open a graphics package and use the color palate to determine the RGB numbers. This will allow a visual representation of the color and provide the exact corresponding RGB values.

Example

```
AWXVIEW320CX1.BackColor = RGB(255, 255, 255)
```

GRIDLINE HStyle

Description

Sets horizontal grid style

Remarks

Grid styles other than solid can not have widths greater than 1. Also some grid styles, other than solid, have trouble printing on some printers.

GRIDLINE is an enumerated list containing the following values:
GridLineDash, GridLineDashdot, GridLineDashdotdot, GridLineDot,
GridLineSolid.

Example

Example of a valid grid style and width settings

```
AWXVIEW320CX1.HStyle = GridLineSolid  
AWXVIEW320CX1.HWidth = 2
```

```
AWXVIEW320CX1.HStyle = GridLineDot
```

AWXVIEW320CX1.HWidth = 1

GRIDLINE VStyle

Description

Sets vertical grid style

Remarks

Grid styles other than solid can not have widths greater than 1. Also some grid styles, other than solid, have trouble printing on some printers.

GRIDLINE is an enumerated list containing the following values:
GridLineDash, GridLineDashdot, GridLineDashdotdot, GridLineDot,
GridLineSolid.

Example

Example of a valid grid style and width settings

```
AWXVIEW320CX1.VStyle = GridLineSolid  
AWXVIEW320CX1.VWidth = 2
```

```
AWXVIEW320CX1.VStyle = GridLineDot  
AWXVIEW320CX1.VWidth = 1
```

boolean HCheck

Description

Enable/Disable horizontal grid lines

Example

```
AWXVIEW320CX1.HCheck = True
```

boolean VCheck

Description

Enable/Disable vertical grid lines

Example

```
AWXVIEW320CX1.VCheck = True
```

long HWidth

Description

Sets horizontal grid line width

Remarks

Grid styles other than solid cannot have widths greater than 1. Also some grid styles, other than solid, have trouble printing on some printers.

Example

Example of a valid grid style and width settings

```
AWXVIEW320CX1.HStyle = GridLineSolid
```

```
AWXVIEW320CX1.HWidth = 2
```

```
AWXVIEW320CX1.HStyle = GridLineDot  
AWXVIEW320CX1.HWidth = 1
```

long VWidth

Description

Sets vertical grid line width

Remarks

Grid styles other than solid can not have widths greater than 1. Also some grid styles, other than solid, have trouble printing on some printers.

Example

Example of a valid grid style and width settings

```
AWXVIEW320CX1.VStyle = GridLineSolid  
AWXVIEW320CX1.VWidth = 2
```

```
AWXVIEW320CX1.VStyle = GridLineDot  
AWXVIEW320CX1.VWidth = 1
```

OLE_COLOR HorColor

Description

Sets horizontal grid line color

Remarks

RGB defines colors in the order of red, green, blue. A good trick to use when picking colors is to open a graphics package and use the color palate to determine the RGB numbers. This will allow a visual representation of the color and provide the exact corresponding RGB values.

Example

```
AWXVIEW320CX1.HorColor = RGB(255, 255, 255)
```

OLE_COLOR VerColor

Description

Sets vertical grid line color

Remarks

RGB defines colors in the order of red, green, blue. A good trick to use when picking colors is to open a graphics package and use the color palate to determine the RGB numbers. This will allow a visual representation of the color and provide the exact corresponding RGB values.

Example

```
AWXVIEW320CX1.VerColor = RGB(255, 255, 255)
```

boolean AutoStart

Description

Enable/Disable AutoStart feature

Remarks

AutoStart allows the viewer to enter a runtime state when the container has AmbientUserMode set to True.

Example

```
AWXVIEW320CX1.AutoStart = True
```

boolean NoToolTips**Description**

Enable/Disable NoToolTips as runtime option

Remarks

Setting this property to False will gray out the NoToolTips option on the right click runtime pop-up menu.

Example

```
AWXVIEW320CX1.NoToolTips = False
```

boolean Help**Description**

Enable/Disable Help ToolTips as runtime option

Remarks

Setting this property to False will gray out the Help option on the right click runtime pop-up menu.

Example

```
AWXVIEW320CX1.Help = False
```

boolean Event**Description**

Enable/Disable Event ToolTips as runtime option

Remarks

Setting this property to False will gray out the Event option on the right click runtime pop-up menu.

Example

```
AWXVIEW320CX1.Event = False
```

boolean Context**Description**

Enable/Disable Context ToolTips as runtime option

Remarks

Setting this property to False will gray out the Context option on the right click runtime pop-up menu.

Example

```
AWXVIEW320CX1.Context = False
```

boolean Comment

Description

Enable/Disable Comment ToolTips as runtime option

Remarks

Setting this property to False will gray out the Comment option on the right click runtime pop-up menu.

Example

```
AWXVIEW320CX1.Comment = False
```

boolean AlarmLine

Description

Enable/Disable entire alarm line popup ToolTips as runtime option

Remarks

Setting this property to False will gray out the AlarmLine option on the right click runtime pop-up menu.

Example

```
AWXVIEW320CX1.AlarmLine = False
```

TOOLTIP DefaultTooltip

Description

Set default ToolTip setting for runtime.

Remarks

This property sets ToolTip to use on startup. Valid ToolTip values are as follows:

- PopupAlarmLine
- PopupComment
- PopupContext
- PopupHelp
- PopupNone

Example

AWXVIEW32OCX1.Comment = False

short TooltipWidth

Description

Set tooltip width.

Remarks

Longer ToolTip widths will cause the ToolTip to have fewer rows.

Example

AWXVIEW32OCX1.TooltipWidth = 15

long CommentHeader

Description

Defines a header to associate with the Comment pop-up ToolTips

Remarks

This call should be used in conjunction with the position of the desired header. This causes the comment option to display the information in the associated column.

Example

```
Dim Val1 As Integer  
Val1 = AWXVIEW32OCX1.ColumnNametoPosition("Column1")  
AWXVIEW32OCX1.CommentHeader = Val1
```

long HelpHeader

Description

Defines a header to associate with the Help pop-up ToolTips

Remarks

This call should be used in conjunction with the position of the desired header. This causes the help option to display the information in the associated column.

Example

```
Dim Val1 As Integer  
Val1 = AWXVIEW32OCX1.ColumnNametoPosition("Column1")  
AWXVIEW32OCX1.HelpHeader = Val1
```

boolean AlarmDouble

Description

Enable/Disable left mouse double click to acknowledge alarms

Example

AWXVIEW320CX1.AlarmDouble = False

boolean AckDialog

Description

Enable/Disable the forcing of alarm acknowledgement through ack dialog.

Remarks

Enabling this property forces all acknowledgements through the ack dialog. This differs from the AlarmDouble property in the fact that a double click with AckDialog set to true will bring up the acknowledge dialog.

Example

AWXVIEW320CX1.AckDialog = False

boolean EnableMoveCol

Description

Enable/Disable the changing of column order during runtime

Example

AWXVIEW320CX1.EnableMoveCol = True

boolean EnableResizeCol

Description

Enable/Disable the resizing of columns during runtime

Remarks

This feature should only be used for design and testing due to the fact that it would allow the user to completely hide a column of information.

Example

AWXVIEW320CX1.EnableResizeCol = True

boolean EnableResizeRow

Description

Enable/Disable the resizing of rows during runtime

Remarks

This feature should only be used for design and testing due to the fact that it would allow the user to completely hide alarm lines.

Example

AWXVIEW320CX1.EnableResizeRow = True

BSTR SelectedSource

Description

Returns the selected source string. Source name is chosen based on messages selected, alarm state, severity, and time

Remarks

This property will return the source string (tag name) that best fits the following criteria:

- 1) Highlighted alarm line.
- 2) Highest alarm state.
- 3) Highest severity.
- 4) Most recent time stamp.

Alarm state priority is defined as follows:

- 1) In Alarm and unacknowledged
- 2) In Alarm but acknowledged
- 3) Passed out of Alarm but unacknowledged
- 4) Tracking message
- 5) Operator message

Example

```
Dim SourceName As String
SourceName = AWXVIEW320CX1.SelectedSource
MsgBox (SourceName)
```

BSTR PrioritySource

Description

Returns the priority source string. Source name is chosen based on messages in alarm, severity, and time.

Remarks

This property returns a source name (tag name) that best fits the following criteria:

- 1) In alarm.
- 2) Highest severity.
- 3) Most recent time stamp.

If no alarm messages are in alarm and unacknowledged this property will return an empty string.

Example

```
Dim SourceName As String
SourceName = AWXVIEW320CX1.PrioritySource
MsgBox (SourceName)
```

Available Control Methods

boolean ShowColumn(COLIMNID ColumnId, boolean bShow)

Description

This method will hide and show columns. If bShow is True it will attempt to show the column. Returns a True on success.

Example

This example shows the column alarm type

```
Dim bVal as Boolean  
bVal = AWXVIEW320CX1.ShowColumn(ALARMTYPE, True)
```

boolean SetToolTip(TOOLTIP Id)

Description

This method will set the current Tooltip mode to the type associated with TOOLTIP id.

Remarks

See the DefaultTooltip property for TOOLTIP id list.

Example

```
Dim bVal as Boolean  
bVal = AWXVIEW320CX1.SetToolTip(PopupAlarmline)
```

boolean SortOn(short Count, COLIMNID Sort1, boolean Direction1, COLIMNID Sort2, boolean Direction2, COLIMNID Sort3, boolean Direction3, COLIMNID Sort4, boolean Direction4, COLIMNID Sort5, boolean Directon5)

Description

Sorts grid. Pass in columnID and direction combinations to set up sort. Count is the number of items to sort on. Pass zero for all unused parameters. The first pair can not be zero. Returns true on success. A direction of True causes accending order.

Remarks

The count parameter is used to determin how many columnID and direction pairs to use for sorting.

Example

This example causes the grid to be sorted first by alarm type and then by time. The effect is to have the earliest most recent alarms on top and the oldest operator messages on the bottom.

```
Dim bVal as Boolean
```

```
bVal = AWXVIEW320CX1.SortOn(2, ALARMTYPE, True,  
AEACTIVETIME,False , 0, 0, 0, 0, 0, 0)
```

boolean SetColumnName(BSTR Name, short Pos)

Description

This method sets the name of the column located at the position passed in.

Remarks

Column position can change due to runtime user interaction or script. Please be aware that column position counts hidden columns when doing calculations.

Example

```
Dim bVal As Boolean  
bVal = AWXVIEW320CX1.SetColumnName("NewCol1",3)
```

short ColumnNameToPosition(BSTR Name)

Description

Pass in string name (Case sensitive) returns column position. A return of zero is a failure.

Remarks

Column position can change due to runtime user interaction or script. Please be aware that column position counts hidden columns when doing calculations.

Example

```
Dim ColPos as Integer  
ColPos = AWXVIEW320CX1.ColumnNameToPosition("Column1")
```

short ColumnNameToId(BSTR Name)

Description

Pass in string name (Case sensitive) returns column id. A Value of zero is a failure.

Remarks

The column id value corresponds to the following list:

AECHANGEMASK = 0,

AENEWSTATE = 1,

AESOURCE = 2,

AETIME = 3,

AEMESSAGE = 4,

AEEVENTTYPE = 5,

AEEVENTCATEGORY = 6,

AESEVERITY = 7,

AECONDITIONNAME = 8,
AESUBCONDITIONNAME = 9,
AEQUALITY = 10,
AEACKREQUIRED = 11,
AEACTIVETIME = 12,
AECOOKIE = 13,
AENUMEVENTATTRS = 14,
AEACTORID = 15,
ATTRIB1 = 16,
ATTRIB2 = 17,
ATTRIB3 = 18,
ATTRIB4 = 19,
ATTRIB5 = 20,
ATTRIB6 = 21,
ATTRIB7 = 22,
ATTRIB8 = 23,
ATTRIB9 = 24,
ATTRIB10 = 25,
ATTRIB11 = 26,
ATTRIB12 = 27,
ATTRIB13 = 28,
ATTRIB14 = 29,
ATTRIB15 = 30,
ATTRIB16 = 31,
ATTRIB17 = 32,
ATTRIB18 = 33,
ATTRIB19 = 34,
ATTRIB20 = 35,
ALARMTYPE = 36,

Example

```
Dim ColId as Integer  
ColId = AWXVIEW320CX1.ColumnNameToId("Column1")
```

BSTR IdtoColumnName(COLIMNID Id)

Description

Pass in columnID. Returns string name.

Remarks

See ColumnNameToId function for Id list.

Example

```
Dim ColName as String  
ColName = AWXVIEW320CX1.IdtoColumnName(AEACTIVETIME)
```

BSTR PositiontoColumnName(short Position)

Description

Pass in position. Returns string name.

Remarks

Column position can change due to runtime user interaction or script. Please be aware that column position counts hidden columns when doing calculations.

Example

```
Dim ColName as String  
ColName = AWXVIEW320CX1.PositiontoColumnName(2)
```

long GetNumberOfAlarms()

Description

Returns current number of items in alarm.

Example

```
Dim Val1 As Integer  
Val1 = AWXVIEW320CX1.GetNumberOfAlarms()
```

long GetNumberOfAked()

Description

Returns current number of items in acknowledged state.

Example

```
Dim Val1 As Integer  
Val1 = AWXVIEW320CX1.GetNumberOfAked()
```

long GetNumberOfUnacked()

Description

Returns current number of items in unacknowledged state.

Example

```
Dim Val1 As Integer  
Val1 = AWXVIEW320CX1.GetNumberOfUnacked()
```

long GetNumberOfTracking()**Description**

Returns current number of tracking items.

Example

```
Dim Val1 As Integer  
Val1 = AWXVIEW320CX1.GetNumberOfTracking()
```

long GetNumberOfOperator()**Description**

Returns current number of operator items.

Example

```
Dim Val1 As Integer  
Val1 = AWXVIEW320CX1.GetNumberOfOperator()
```

long GetTotalNumberOfMessages()**Description**

Returns total number of current messages.

Example

```
Dim Val1 As Integer  
Val1 = AWXVIEW320CX1.GetNumberOfMessages()
```

long GetNumberOfFilterAlarms()**Description**

Returns current number of items in alarm that pass through the client filter.

Example

```
Dim Val1 As Integer  
Val1 = AWXVIEW320CX1.GetNumberOfFilterAlarms()
```

long GetNumberOfFilterAcked()**Description**

Returns current number of items in acknowledged state that pass through the client filter.

Example

```
Dim Val1 As Integer  
Val1 = AWXVIEW320CX1.GetNumberOfFilterAcked()
```

long GetNumberOfFilterUnacked()

Description

Returns current number of items in unacknowledged state that pass through the client filter.

Example

```
Dim Val1 As Integer  
Val1 = AWXVIEW320CX1.GetNumberOfFilterUnacked()
```

long GetNumberOfFilterTracking()

Description

Returns current number of tracking items that pass through the client filter.

Example

```
Dim Val1 As Integer  
Val1 = AWXVIEW320CX1.GetNumberOfFilterTracking()
```

long GetNumberOfFilterOperator()

Description

Returns current number of operator items that pass through the client filter.

Example

```
Dim Val1 As Integer  
Val1 = AWXVIEW320CX1.GetNumberOfFilterOperator()
```

long GetTotalNumberOfFilterMessages()

Description

Returns total number of current messages that pass through the client filter.

Example

```
Dim Val1 As Integer  
Val1 = AWXVIEW320CX1.GetTotalNumberOfFilterMessages()
```

short Positiontold(short Pos)

Description

Pass in position returns column ID.

Remarks

Column position can change due to runtime user interaction or script. Please be aware that column position counts hidden columns when doing calculations.

Example

```
Dim Val1 As Integer  
Val1 = AWXVIEW320CX1.PositiontoId(2)
```

short IdtoPosition(COLIMNID Id);

Description

Pass in column ID return position.

Remarks

Column position can change due to runtime user interaction or script. Please be aware that column position counts hidden columns when doing calculations.

Example

```
Dim Val1 As Integer  
Val1 = AWXVIEW320CX1.IdtoPosition(AEACTIVETIME)
```

boolean SizeColAtPosition(short Pos, long Size);

Description

Pass in position and new size, changes size of column at that position. Return of true means success. Size must be larger than minimum width as seen in property pages. (equal to or larger than 1)

Remarks

Column position can change due to runtime user interaction or script. Please be aware that column position counts hidden columns when doing calculations. Width values are based on an internal algorithm composed of font size and type

Example

Changes column in position 2 to a width of 15

```
Dim bVal As Boolean  
bVal = AWXVIEW320CX1.SizeColAtPosition(2,15)
```

boolean SizeColWithId(COLIMNID Id, long Size)

Description

Pass in column ID and new size, changes size of column at that position. Return of true means success. Size must be larger than minimum width as seen in property pages. (equal to or larger than 1)

Remarks

Width values are based on an internal algorithm composed of font size and type

Example

Changes the time column to a width of 15

```
Dim bVal As Boolean  
bVal = AWXVIEW320CX1.SizeColWithId(AEACTIVETIME,15)
```

boolean MovCol(short FromStartPos, short FromEndPos, short ToPos)

Description

Move columns. FromStartPos is first column selected. FromEndPos is last column selected. Moves group of columns to ToPos. Cannot move past last column. Return of True means success.

Remarks

Please be aware that column position counts hidden columns when doing calculations.

Example

This Example will move columns 3, 4, and 5 in front of column 2.

```
Dim bVal As Boolean
bVal = AWXVIEW320CX1.MovCol(3,5,2)
```

boolean EnableColTextColorOverride(COLIMNID Id, boolean Enable)

Description

Changes text color for specified column. Enable = true enables override. Returns true on success.

Example

This Example enables the text color of the time column to be overwritten and then changes the text color.

```
Dim bVal As Boolean
bVal =
AWXVIEW320CX1.EnableColTextColorOverride(AEACTIVETIME,
True)
Call
AWXVIEW320CX1.SetColTextColor(AEACTIVETIME, RGB(255,255,255)
)
```

boolean EnableColBackColorOverride(COLIMNID Id, boolean Enable)

Description

Changes background color for specified column. Enable = true enables override. Returns true on success.

Example

This Example enables the background color of the time column to be overwritten and then changes the background color.

```
Dim bVal As Boolean
bVal =
AWXVIEW320CX1.EnableColBackColorOverride(AEACTIVETIME,
True)
Call
AWXVIEW320CX1.SetColBackColor(AEACTIVETIME, RGB(255,255,255)
)
```

void SetColTextColor(COLIMNID Id, OLE_COLOR Color)

Description

Sets text color of specified column. Automatically enables text color override.

Remarks

This function will enable the text color to be overwritten.

Example

```
Dim bVal As Boolean
Call
AWXVIEW320CX1.SetColTextColor(AEACTIVETIME, RGB(255, 255, 255)
)
```

void SetColBackColor(COLIMNID Id, OLE_COLOR Color)

Description

Sets background color of specified column. Automatically enables background color override.

Remarks

This function will enable the background color to be overwritten.

Example

```
Dim bVal As Boolean
Call =
AWXVIEW320CX1.SetColBackColor(AEACTIVETIME, RGB(255, 255, 255)
)
```

OLE_COLOR GetColBackColor(COLIMNID Id)

Description

Returns text color of specified column. - **Note**: this is the override color.

Remarks

This returns the column background color not the base default background color.

Example

```
Dim rcolor As OLE_COLOR
rcolor = AWXVIEW320CX1.GetColBackColor(AEACTIVETIME)
```

OLE_COLOR GetColTextColor(COLIMNID Id)

Description

Returns background color of specified column. - **Note**: this is the override color.

Remarks

This returns the column text color not the base default text color.

Example

```
Dim rcolor As OLE_COLOR
rcolor = AWXVIEW320CX1.GetColTextColor(AEACTIVETIME)
```

boolean EnableRowTextColor(MESSAGETYPE wType, long LowSeverity, boolean Enable)

Description

Enables/Disables text color override for associated message type and rowID. Enable = True enables override. Returns true on success.

Remarks

The messagetype values are as follows:

- Alarm = 1
- Acknowledge = 2
- Unacknowledge = 3
- Operator = 4
- Tracking = 5
- Normal = 6

Example

This Example will disable the setting of the row text color for all alarm lines in alarm for severity values starting at 400 to the next configured severity or 1000. Which ever is less.

Dim bVal As Boolean

```
bVal = AWXVIEW320CX1.EnableRowTextColor(Alarm, 400, False)
```

boolean EnableRowBackColor(MESSAGETYPE wType, long LowSeverity, boolean Enable)

Description

Enables/Disables background color override for associated message type and rowID. Enable = True enables override. Returns true on success.

Remarks

See EnableRowTextColor method for available messagetype values.

Example

This Example will disable the setting of the row background color for all alarm lines in alarm for severity values starting at 400 to the next configured severity or 1000. Which ever is less.

Dim bVal As Boolean

```
bVal = AWXVIEW320CX1.EnableRowBackColor(Alarm, 400, False)
```

void SetRowTextColor(MESSAGETYPE wType, long LowSeverity, OLE_COLOR Color)

Description

Sets text color associated with message type and rowID. Automatically enables text color override.

Remarks

See EnableRowTextColor method for available messagetype values.

Example

This Example will change the row text color for all alarm lines in alarm for severity values starting at 400 to the next configured severity or 1000. Which ever is less.

```
Call  
AWXVIEW320CX1.SetRowTextColor(Alarm, 400, RGB(255, 255, 255))
```

void SetRowBackColor(MESSAGETYPE wType, long LowSeverity, OLE_COLOR Color)

Description

Sets background color associated with message type and rowID. Automatically enables background color override.

Remarks

See EnableRowTextColor method for available messagetype values.

Example

This Example will change the row background color for all alarm lines in alarm for severity values starting at 400 to the next configured severity or 1000. Which ever is less.

```
Call  
AWXVIEW320CX1.SetRowBackColor(Alarm, 400, RGB(255, 255, 255))
```

OLE_COLOR GetRowTextColor(MESSAGETYPE wType, long LowSeverity)

Description

Returns the row text color of the associated message type rowID combination. **Note** - this is the override color.

Remarks

See EnableRowTextColor method for available messagetype values.

Example

This example will get the row text color value for severity values starting at 400 to the next configured severity or 1000. Which ever is less.

```
Dim rcolor As OLE_COLOR  
rcolor = AWXVIEW320CX1.GetRowTextColor(Alarm, 400)
```

OLE_COLOR GetRowBackColor(MESSAGETYPE wType, long LowSeverity)

Description

Returns the row background color of the associated message type rowID combination. **Note** - this is the override color.

Remarks

See EnableRowTextColor method for available messagetype values.

Example

This example will get the row background color value for severity values starting at 400 to the next configured severity or 1000. Which ever is less.

```
Dim rcolor As OLE_COLOR  
rcolor = AWXVIEW320CX1.GetRowBackColor(Alarm, 400)
```

GlobalAck(BSTR comment)

Description

Acknowledges all received alarms.

Remarks

The comment is passed to the server.

Example

```
Call AWXVIEW320CX1.GlobalAck("down for maintenance")
```

AckSelectedRows(BSTR comment)

Description

Acknowledges rows selected by the operator.

Remarks

The comment is passed to the server.

Example

```
Call AWXVIEW320CX1.AckSelectedRows("down for maintenance")
```

void AckDisplayed(BSTR comment)

Description

Acknowledges all displayed alarms

Remarks

This function will acknowledge all alarm messages visible on the screen. The comment is passed to the server.

Example

```
Call AWXVIEW320CX1.AckDisplayed("down for maintenance")
```

void AckFiltered(BSTR comment)

Description

Acknowledges all alarms that pass through the client filter. This differs from the global acknowledge by the fact that the global acknowledge will also acknowledge requested alarms that the client filter has filtered out.

Remarks

The comment is passed to the server.

Example

Call AWXVIEW320CX1.AckFiltered("down for maintenance")

void SetActorID(BSTR Operator, BSTR Station)

Description

Creates string of defining user who acknowledged an alarm. Syntax is as follows user: Operator Node: Station

Remarks

Once ActorID is called the ocx will not get the ActorID from the security system unless resetActorID is called.

Example

Call AWXVIEW320CX1.SetActorID("NewOperator", "NewStation")

void SetSecurity(long Security)

Description

Allows the AlarmWorX viewer security to be set through an automation interface.

Remarks

Once enabled, Iconics security server will not be used for the life of the ocx. The security map is as follows:

m_bSecAckPoint	0x0001
m_bSecGlobalAck	0x0002
m_bSecPageAck	0x0004
m_bSecScreenAck	0x0008
m_bSecAreaAck	0x0010
m_bSecVBEvent	0x0020
m_bSecSort	0x0040
m_bSecFilter	0x0080
m_bSecDetails	0x0100
m_bSecProperty	0x0200
m_bSecApplication	0x0400
m_bSecDisplay	0x0800
m_bSecTooltip	0x1000
m_bSecColumn	0x2000
m_bSecLanguage	0x4000
m_bSecChFilter	0x8000

A 1 enables the feature and a 0 disables the feature. By default all items are disabled.

A registry setting of Security 0 must be added to HKEY_LOCAL_MACHINE\software\iconics\awxview32 for this automation method to be enabled. This registry setting disables the use of the security server for the Alarm viewer.

Example

This Example enables all options except AckPoint (point acknowledge)

```
Dim Security As Integer
Security = 65534 '65534 = 0xffff
Call AWXVIEW32OCX1.SetSecurity(Security)
```

void ResetActorID()

Description

Sets the Actor ID back to the internal actor id string.

Example

```
Call AWXVIEW32OCX1.ResetActorID()
```

void FilterOff()

Description

Turns off all client filters.

Example

```
Call AWXVIEW32OCX1.ResetActorID()
```

BSTR GetActiveFilter()

Description

Returns the active client filter name.

Remarks

This is the name of the filter not the filter expression

Example

```
Dim FilterName As String
FilterName = AWXVIEW32OCX1.GetActiveFilter()
```

BSTR GetFilter(LPCTSTR Filter)

Description

Returns the filter expression.

Remarks

Given the name of an existing filter this method will return the filter expression string. An empty string is returned if the filter does not exist.

Example

```
Dim FilterName As String
Dim FilterExpression As String
FilterExpression = AWXVIEW320CX1.GetFilter(FilterName)
```

boolean SetFilter(LPCTSTR Filter, BSTR Expression)

Description

Creates Filter if Filter does not already exist. Sets Expression as the expression string associated with Filter. Returns a true on success.

Remarks

The best way to ensure accurate expression syntax is to create the string using the client filter dialog and then copy the string expression. This method only adds the filter and filter name to the list of possible client filters. It does not activate the filter.

Example

This example creates a filter named filter1 with an expression that only passes alarms with a source name beginning with Tag.

```
Dim FilterName As String
Dim FilterExpression As String

FilterName = "Filter1"
'actual expression x= like({{Source}},"Tag*"$,0)
FilterExpression = "like({{Source}},"Tag*"$,0)"
FilterExpression =
AWXVIEW320CX1.SetFilter(FilterName,FilterExpression)
```

boolean EnableFilter(LPCTSTR Filter, boolean State)

Description

Turns Filter on if state is True or off if state is False. Returns true on success.

Remarks

This method will turn filters on and off. When a filter is turned on, the previous filter will be turned off. It is only possible to have one filter running at any given time.

Example

This example turns the filter named filter1 on.

```
Dim FilterName As String
Dim bVal as Boolean
FilterName = "Filter1"
bVal = AWXVIEW320CX1.EnableFilter(FilterName,True)
```

boolean AlarmFilter(BSTR Filter, long StartSeverity, long EndSeverity, long AlarmType, BSTR AttributeField1, BSTR Attribute1, BSTR AttributeField2, BSTR Attribute2)

Description

Sets Filter with the following syntax:

```
( {{Severity}} >= StartSeverity )&&
( {{Severity}} <= EndSeverity )&&
( {{AlarmType}} == AlarmType )&&
( like({{Attribute1}}, $"AttributeField1"$,0) )&&
( like({{Attribute2}}, $"AttributeField2"$,0) )
```

Remarks

AlarmType decodes as follows:

- ALARM 1
- ACK 2
- UNACK 3
- OPER 4
- TRACK 5
- NORM 6

An AlarmType of 7 or greater removes alarm type from the filter.

The attribute field compare the string AttributeField with the string found in the column of the associated attribute. Comparison is done using the like function. The like function behaves identically to the VB like operator.

Attribute1 is removed from the filter if AttributeField1 is the string *

Attribute2 is removed from the filter if AttributeField2 is the string *

It is recommended that the SetFilter and EnableFilter methods are used instead of this method.

Example

```
Dim bVal as Boolean
bVal = AWXVIEW320CX1.AlarmFilter("Filter1",0,1000, Alarm,
"\"", "extracol1", "\"", "extracol2")
```

void EnterAnimate()

Description

Causes the ActiveX to enter runtime mode.

Example

This example causes the viewer to exit runtime, load file test.awv, and enter runtime.

```
Dim FileName As String
Dim bVal as Boolean

FileName = "c:\test\test.awv"
Call AWXVIEW320CX1.ExitAnimate()
BVal = AWXVIEW320CX1.LoadSaveFile(FileName,1)
Call AWXVIEW320CX1.EnterAnimate()
```

void ExitAnimate()

Description

Causes the ActiveX to exit runtime mode and enter design mode.

Example

This example causes the viewer to exit runtime, load file test.awv, and enter runtime.

```
Dim FileName As String
Dim bVal as Boolean

FileName = "c:\test\test.awv"
Call AWXVIEW320CX1.ExitAnimate()
BVal = AWXVIEW320CX1.LoadSaveFile(FileName,1)
Call AWXVIEW320CX1.EnterAnimate()
```

boolean LoadSaveFile(BSTR FileName, short Load)

Description

Will cause the ActiveX to load or save an *.awv file. If Load is 1 then the file will be loaded. Load 0 will cause the file to be saved.

Remarks

This method will only load or save a file when the ActiveX is in design mode. The ActiveX properties Check must be False and FileEnabled must be True. This should be done through the property pages and saved as part of configuration. To set the property pages to load and save from a file, select Configuration File in the general properties tab and disable URL Path option. FileName should be the full path and filename including extension.

Example

This example causes the viewer to exit runtime, load file test.awv, and enter runtime.

```
Dim FileName As String
Dim bVal as Boolean

FileName = "c:\test\test.awv"
Call AWXVIEW320CX1.ExitAnimate()
BVal = AWXVIEW320CX1.LoadSaveFile(FileName,1)
```

Call AWWVIEW32OCX1.EnterAnimate()

long ControlSize(long cx, long cy)

Description

Sets the size of the ocx. Returns a 0 on success.

Remarks

Cx and cy are in windows pixel units.

Example

```
Dim size As Integer
size = AWWVIEW32OCX1.ControlSize(200, 300)
```

boolean ChangeNode(BSTR OldNode, BSTR NewNode)

Description

This method changes the node name associated with the alarm subscription. This method will cause the alarm subscription to switch the network nodes it is using for data collection.

Remarks

There are possible performance hits when using this function.

Example

```
Dim bVal As Boolean
bVal = AWWVIEW32OCX1.ChangeNode("PC12", "PC13")
```

void UpdateDisplay()

Description

This method causes a redraw of the internal grid information.

Remarks

This should be used at the end of a script to force the update of automation color changes.

Example

```
Dim bVal As Boolean
Call
AWXVIEW32OCX1.SetColBackColor(AEACTIVETIME, RGB(255, 255, 255)
)
Call
AWXVIEW32OCX1.SetColTextColor(AEACTIVETIME, RGB(255, 255, 255)
)
Call AWWVIEW32OCX1.UpdateDisplay()
```

BSTR SelectedColRowInfo(COLIMNID Id)

Description

This method returns a string containing the text in the column corresponding to 'Id' for the row selected in the Alarm Viewer ActiveX.

Example

```
Dim TimeInfo as string
'Get the time of the selected row
TimeInfo = AWXVIEW320CX1.SelectColRowInfo(AETIME)
```

BSTR ColRowInfo(long Row, COLIMNID Id)

Description

This method takes the row number and column ID as input parameters. It returns the row and column information in the form of a string.

Example

```
Dim s As String
s = AWXVIEW320CX1.ColRowInfo(1, AESOURCE)
MsgBox (s)
```

This example will pop up a message box with the source of the alarm message.

void ScrollUp();

Description

Scroll up the value of one arrow click

Example

```
AWXVIEW320CX1.ScrollUp
```

void ScrollDwn();

Description

Scroll down the value of one arrow click

Example

```
AWXVIEW320CX1.ScrollUp
```

void ScrollPgUp();

Description

Scroll up the value of one scroll bar click

Example

```
AWXVIEW320CX1.ScrollPgUp
```

void ScrollPgDwn();

Description

Scroll down the value of one scroll bar click

Example

AWXVIEW320CX1.ScrollPgDwn

void ScrollLeft();

Description

Scroll left the value of one arrow click

Example

AWXVIEW320CX1.ScrollLeft

void ScrollRight();

Description

Scroll right the value of one arrow click

Example

AWXVIEW320CX1.ScrollRight

void ScrollPgLeft();

Description

Scroll left the value of one scroll bar click

Example

AWXVIEW320CX1.ScrollPgLeft

void ScrollPgRight();

Description

Scroll right the value of one scroll bar click

Example

AWXVIEW320CX1.ScrollPgRight

long GetNumberOfCol();

Description

Returns the number of columns

Example

```
Dim col As Long  
col = AWXVIEW320CX1.GetNumberOfCol()  
MsgBox (col)
```

This example will pop up a message box with the number of columns.

void ComputerName();

Description

Returns the local node name

Example

```
Dim s As String  
s = AWXVIEW320CX1.ComputerName()  
MsgBox (s)
```

This example will pop up a message box with the computer name.

long GetColSizeAtPosition(short Pos);

Description

Pass in the column position. Returns the size of the column.

Example

```
Dim size As Long  
size = AWXVIEW320CX1.GetColSizeAtPosition(1)  
msgbox (size)
```

This example will pop up a message box with the size of the first column.

long GetColSizeWithId(short Id);

Description

Pass in the column Id. Returns the size of the column.

Example

```
Dim size As Long  
size = AWXVIEW320CX1.GetColSizeWithId(AESOURCE)  
MsgBox (size)
```

This example will pop up a message box with the size of the AESOURCE column.

boolean IsEnableRowTextColor(short wType, long LowSeverity);

Description

Pass in the message type and severity. Returns true if the row text color setting is true. Returns false otherwise.

Remarks

See EnableRowTextColor method for available messagetype values.

Example

This example will get the EnableRowTextColor value for severity values starting at 400 to the next configured severity or 1000. Which ever is less.

```
Dim b As Boolean  
b = AWXVIEW320CX1.IsEnableRowTextColor(Alarm, 400)
```

boolean IsEnableRowBackColor(short wType, long LowSeverity);

Description

Pass in the message type and severity. Returns true if the row background color setting is true. Returns false otherwise.

Remarks

See EnableRowTextColor method for available messagetype values.

Example

This example will get the EnableRowBackColor value for severity values starting at 400 to the next configured severity or 1000. Which ever is less.

```
Dim b as Boolean  
b = AWXVIEW320CX1.IsEnableRowBackColor(Alarm, 400)
```

boolean IsEnableColTextColorOverride(short Id);

Description

Pass in the column Id. Returns true if the column text color setting is true. Returns false otherwise.

Example

```
Dim b as Boolean  
b = AWXVIEW320CX1.IsEnabledColTextColorOverride (AESOURCE)
```

boolean IsEnableColBackColorOverride(short Id);

Description

Pass in the column Id. Returns true if the column background color setting is true. Returns false otherwise.

Example

```
Dim b as Boolean  
b = AWXVIEW320CX1.IsEnableColBackColorOverride (AESOURCE)
```

long GetNumberOfSelectedRows();

Description

Returns the number of selected rows.

Example

```
Dim l As Long  
l = AWXVIEW320CX1.GetNumberOfSelectRows()  
MsgBox (l)
```

This example will pop up a message box with the number of selected rows.

void SetGasParentCookie(long newCookie);

Description

For Global Aliasing System (GAS) support only

Remarks

This function is hidden and should not be used through custom scripts

Example

None

boolean SetGASThemes(BSTR strThemes);

Description

For Global Aliasing System (GAS) support only

Remarks

This function is hidden and should not be used through custom scripts

Example

None

Available Control Events

OnRightMouseClicked()

Description

Event posted on user right mouse click

Example

```
Private Sub AWXVIEW320CX1_OnRightMouseClicked()  
'Your code here  
End Sub
```

OnLeftMouseClicked()

Description

Event posted on user left mouse click

Example

```
Private Sub AWXVIEW320CX1_OnLeftMouseClicked()  
'Your code here  
End Sub
```

OnRightMouseDownClick()

Description

Event posted on user right double mouse click

Example

```
Private Sub AWXVIEW320CX1_OnRightMouseDownClick()
```

```
'Your code here  
End Sub
```

OnLeftMouseDBLClick()

Description

Event posted on user left double mouse click

Example

```
Private Sub AWXVIEW320CX1_OnLeftMouseDbLClick()  
'Your code here  
End Sub
```

OnNewAlarm()

Description

Event posted when any new message enters the viewer.

Remarks

Please Note: This event occurs when any new OPC alarm and events message is passed into the viewer. This includes client filtered messages, operator messages, tracking messages, alarm messages, and acknowledge messages. No information provided to the event that denotes message type.

Example

```
Private Sub AWXVIEW320CX1_OnNewAlarm()  
'Your code here  
End Sub
```

OnPopupEvent()

Description

Event posted when the user selects the VB event menu item from the right click pop-up menu.

Example

```
Private Sub AWXVIEW320CX1_OnPopupEvent()  
'Your code here  
End Sub
```