AVEVA™ Alarm Client Control

2023 R2 SP1

Contents

Welcome to AVEVA Alarm Client Control

Welcome to this release of AVEVA Alarm Client Control.

We would really value your feedback, because you are helping us to make this a better product. We will work closely with you to understand what you want out of this application so that we can improve future versions. If you experience any difficulty using AVEVA Alarm Client Control, please let us know.

Thank you,

The AVEVA Alarm Client Control Team

Supported browsers

Important: Cookies must be enabled in ALL browsers, even while in private browsing or incognito mode.

Supported browsers are used for configuration in AVEVA CONNECT, for downloading the AVEVA Alarm Client Control client, and for viewing user documentation.

AVEVA Alarm Client Control supports the following browsers:

- Google Chrome
- Microsoft Edge, version 79 or newer
- Mozilla Firefox

AVEVA Alarm Client Control no longer supports the following browsers:

- Microsoft Internet Explorer 11
- Microsoft Edge, version 78 or older

Supported languages

We are pleased to provide examples in the following additional languages: French, Spanish, simplified Chinese. They can be enabled and updated by an administrator.

Supported client operating systems

The following table outlines the software requirements for the server/computer on which the AVEVA Alarm Client Control client resides.

Component	Specification
Microsoft Windows	Windows 10 Enterprise 2021 LTSC Windows 10 Enterprise 22H2
Microsoft Windows Server	Microsoft Windows Server Standard 2022 with Desktop Experience Microsoft Windows Server Standard 2019 with Desktop Experience
Microsoft .NET Framework	4.8
Infragistics	22.2

The Alarm Client Control

The Alarm Client Control is a graphical element you can use in your Industrial Graphics to show current and historical alarms and events.

The Alarm Client Control replaces the Alarm Viewer control and Alarm DB View control in the InTouch® HMI and extends alarm visualization to the Industrial graphics environment.

You can place the Alarm Client Control directly from the **Tools** panel in the **Industrial Graphic Editor** onto the canvas. You can customize it to your needs by adding further graphics, interactions, and scripts.

You can deploy a managed InTouch application containing Alarm Client Controls to a remote node and visualize and interact with alarms at run time with InTouch WindowViewer.

For this documentation, the Alarm Client Control is simply referred to as "Alarm Control."

Client modes

The Alarm Client Control supports five different client modes, which can be grouped depending on their data source. The Alarm Client Control supports an ArchestrA Database (A2ALMDB), and the Historian History Blocks.

InTouch alarm manager

The Alarm Control manages currently active alarms (summary alarms) and recent alarms (historical alarms). These types of alarms are saved to internal alarm memory.

Current alarms

When the Alarm Control is showing alarms in "Current Alarms" mode, it is showing currently active alarms directly from the Alarm Manager.

Recent alarms and events

When the Alarm Control is showing alarms in "Recent Alarms and Events" mode, it is showing historical alarms and events stored in Alarm Manager.

Unlike the "Current Alarms" mode, the "Recent Alarms and Events mode" shows time point data, such as alarm transitions and events, instead of continuous conditions.

When the Alarm Client Control is in the Recent Alarms and Events mode, the displayed alarms cannot be acknowledged.

Alarm and event storage

Alarms and events can be stored to the ArchestrA alarm database (A2ALMDB) or the Historian history blocks.

When the Alarm Control is configured in "Historical Alarms" mode, only alarms stored in the Alarm Database are shown.

When the Alarm Control is configured in "Historical Events" mode, only events stored in the Alarm Database are shown.

When the Alarm Control is configured in "Historical Alarms and Events" mode, both alarms and events stored in the Alarm Database are shown.

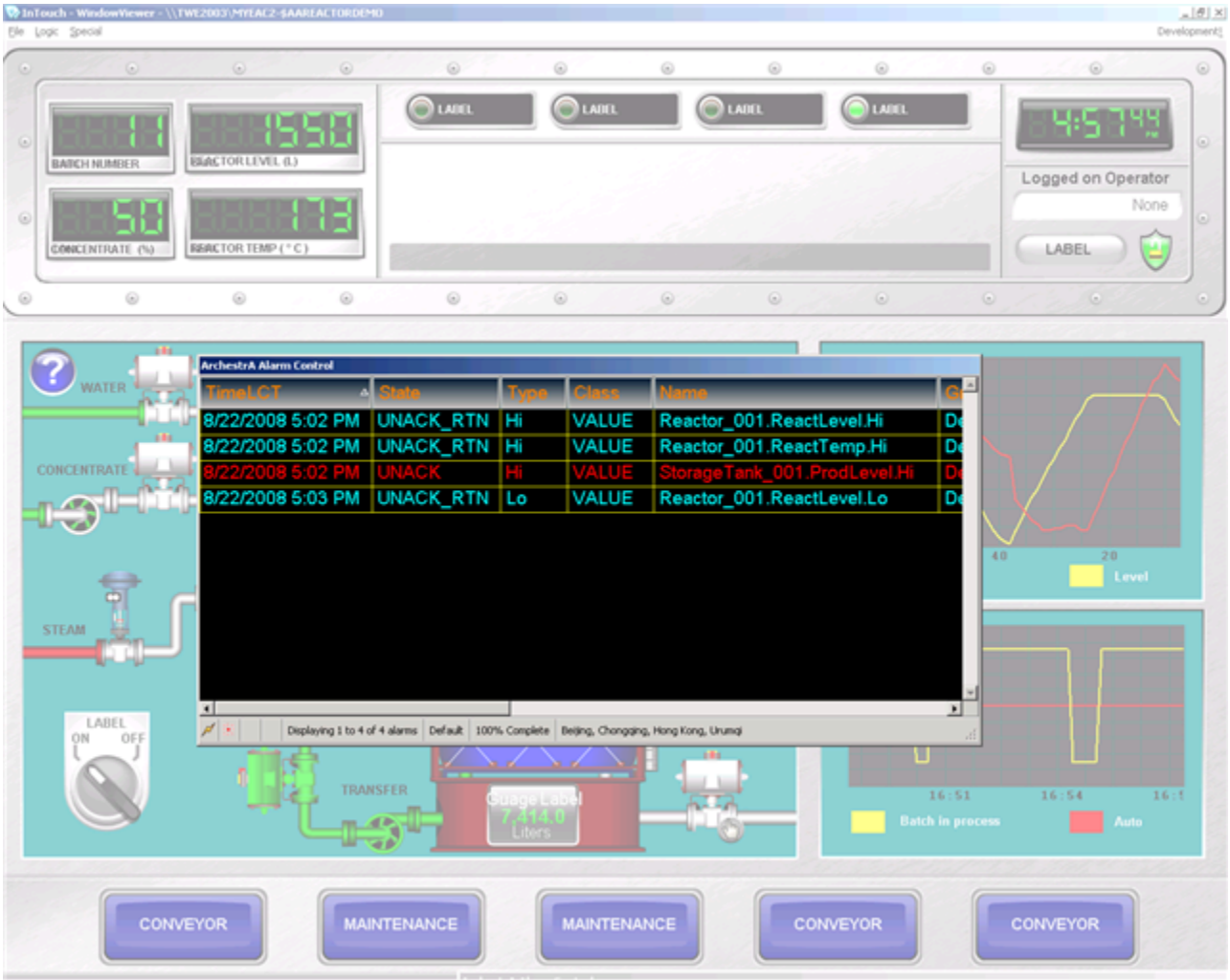
Switch between client modes

The client mode and many other features are controlled by properties and methods.

By default, the Alarm Control is set to show current alarms. You can also change the client mode at run time by using the Alarm Control properties.

Use the Alarm Control in industrial graphics

You can use the Alarm Client Control as a faceplate so that when the operator clicks an icon, an Alarm Client Control showing a specific alarm area opens.



You can also configure the Alarm Client Control to interact with the Galaxy namespace and other Industrial graphics by mapping its properties to application server attributes and symbol elements.

The Alarm Control can be placed into Industrial Graphics hosted by Automation Object templates and instances. You can configure them to retrieve alarms from their hosting Area object or their hosting Automation object.

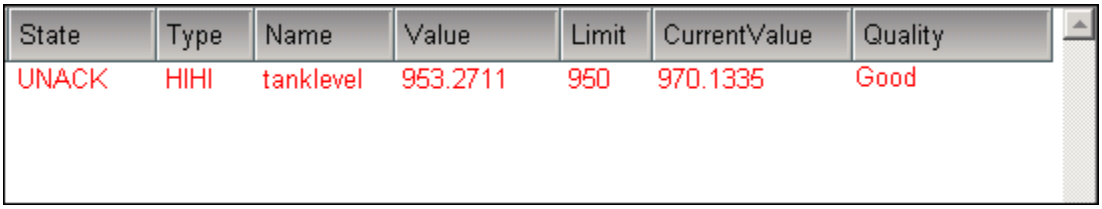
Alarm acknowledgement

You can configure the Alarm Control to require an alarm to be acknowledged even if the condition causing the alarm has passed. This ensures that an operator is aware of events that caused a temporary alarm state but have returned to normal.

You acknowledge alarms at run time using a shortcut menu or through script methods.

Current value and quality display

The Alarm Control in one of the current client modes shows continuously the current value and quality of a tag or attribute in alarm state.



State	Type	Name	Value	Limit	CurrentValue	Quality
UNACK	HIHI	tanklevel	953.2711	950	970.1335	Good

You can see the current value and quality of tags or attributes in alarm from:

- InTouch running on the local computer
- Galaxy namespace

Note: You cannot see current value and quality data from InTouch tags running on a remote computer.

Work with alarm queries and filters

Queries and filters are two methods to retrieve data at **RunTime**, with queries being a superset of filters. You can run a query at run time to retrieve data, then run another query to narrow the search criteria without re-running the query or retrieving the same data. This allows the Alarm Control to use data from a current subscription without re-subscribing. For more information, see Filter alarms.

Alarm queries

The Alarm control supports standard Galaxy alarm query formats, such as:

```
\galaxy!Area_001
```

Alarm query syntax is the same for both Current Alarms Mode and Recent Alarms and Events Mode. Queries in Historical Alarms, Historical Events, and Historical Alarms and Events Modes are actually Alarm Database queries, which follow rules and syntax for SQL Server database queries.

The alarm query syntax changes when you use the run-time alarm comment language switching feature.

The Alarm control also supports relative references for Galaxy alarms in alarm queries. For all alarm modes, relative references are resolved at run time at the point of query to the Alarm Manager or Alarm Database.

You must put the reference part of the alarm query between less-than (<) and greater-than (>) characters.

The following table shows examples of alarm queries.

Alarm Query	**Description**
\provider!group	**Shows all alarms from the given provider and group. For example:** `\intouch!Group_A`
\provider!group!tagname	Shows all alarms from the given provider, group and tag. For example: \galaxy!Mixing_Area!RotorCtrl
\\node\provider!group	Shows all alarms from the given provider and group from a given node. For example: `\\remote\intouch!Group_B`
\\node\provider!group!tagname	Shows all alarms from the given provider, group and tag from a given node. For example: `\\grnode\galaxy!Packaging_Area!Wrapper1`
HotBackupName	Shows all alarms from primary or backup alarm provider as configured in the Hot Backup Manager.
\galaxy!<me.Area>!<me.tagname>.*	Shows all alarms from the Automation Object. Alarms from other Automation Objects in the same area are ignored.
\galaxy!<myArea.tagname> or \galaxy!<me.Area>	Shows all alarms from the Area object hosting the Automation Object
\galaxy!<myPlatform.tagname>	Shows all alarms from the Winplatform object hosting the Automation Object.
\galaxy!<me.area>!<myContainer.tagname>.*	Shows all alarms from the container Automation Object. At run-time the Alarm Control resolves the Container attribute to detect the container.
\galaxy!<myEngine.tagname>	Shows all alarms from the AppEngine object hosting the Automation Object. At run-time the Alarm Control resolves the MyEngine attribute to detect the host.

Alarm query syntax when Register Using Galaxy_<GalaxyName> is enabled

The run-time alarm comment language switching feature requires slightly different alarm query syntax. In the WinPlatform object, when you enable InTouch alarm provider, you can enable **Register using Galaxy_<GalaxyName>** instead of Galaxy.

This option will register the platform to the alarm subsystem using the Galaxy name preferred by "Galaxy_" instead of just the word "Galaxy". This allows an InTouch application to monitor alarms from multiple Galaxies and avoid name conflicts.

Syntax changes slightly when Galaxy_GalaxyName is enabled:

- Use \\ for computer name
- Use \ for Galaxy or Galaxy_<GalaxyName>
- Use ! for Area

For example: \\Galaxy\MyGalaxy!Area001

If Galaxy_GalaxyName is not enabled in WinPlatform, then the default behavior described in Alarm queries applies.

You can determine if Galaxy_<GalaxyName> has been enabled by monitoring the run-time attribute of the platform ITAlarmProvider.ProviderNameAsGalaxyNameEnabled.

Filter alarms

The Query Favorites of InTouch Alarm Viewer control define a set of alarm provider, alarm group, an optional node name, and a priority range under one name. The alarm provider, alarm group, and the node name are used for subscribing to a specific alarm group. The priority range on the other hand is used to filter the alarms from the given alarm group.

The Filter Favorites of InTouch Alarm DB View control define a set of any number of criteria you want to filter from the Alarm Database under one name.

In summary, Filter Favorites fulfill a purely filtering function whereas Query Favorites fulfill a subscription and a filtering function at the same time.

The Alarm Control filtering feature unites both these concepts by exclusively using filter conditions and subscribing to the necessary alarm providers on demand.

Filters can be saved and used in both run time and historical modes.

The filter conditions can be re-used between different client modes. For example, if you define node name, provider name, alarm group, and a priority range for the current alarms, you can also use this filter to retrieve the historized alarm data of the same source from the Alarm Database instead.

Filter definitions will be saved per user so operators working on the same server can access different saved filters.

Alarm queries to filters translation

You can define queries for current alarms in the \\node\provider!group format, but they are translated by the Alarm Control to a filter after you save.

For example, the query string \\GRNode\galaxy!MixingArea is translated to the following filter string:

Node = ‘GRNode’ AND Provider=’galaxy’ AND Group=’MixingArea’

You can modify the filter in a tree to query only alarms in the priority range 1 to 250, such as:

```
AND
   Node = ‘GRNode’
   Provider = ‘Galaxy’
   Group = ‘MixingArea’
   Priority >= ‘1’
   Priority <= ‘250’
```

Shelve alarms

Operators can temporarily shelve selected alarms from the list of an Alarm Control’s active alarms. A shelved alarm is suppressed and removed from the list of active alarms. Typically, operators shelve lower severity nuisance alarms because they provide little diagnostic value and interfere with the operator’s ability to manage a plant process.

An alarm is shelved for a specified period. After the period ends, alarms are automatically unshelved and appear again in the list of active alarms. Operators can also manually unshelve an alarm before the end of the specified shelved period.

By default, Medium and Low severity alarms are enabled for shelving. Critical and High severity alarms are not because of the potential risk of shelving and ignoring alarms that represent serious operating states. For more information about enabling shelving based on alarm severity, see "Enabling Alarm Shelving" in the *Application Server Help*.

Shelve alarms during runtime

When shelving an alarm from the Alarm Control, operators set an associated time period in which the alarm remains shelved and enter a mandatory comment. Operators can select from a list of Alarm Control Context commands during run-time to:

- Shelve one or more selected alarms
- Shelve all alarms
- Shelve only those alarms visible in the Alarm Control
- Shelve all alarms within the same alarm group as an alarm selected from the Alarm Control
- Shelve alarms by selected tags or attributes
- Shelve alarms by selected alarm priorities
- Shelve all alarms that have the same severity as an alarm selected from the Alarm Control

For more information about configuring shelve Context commands, see Configure the runtime shortcut menu.

When application security is used, alarms can be shelved and unshelved only by operators with proper authorization. For more information about setting shelving authorization, see Configure the Alarm Control to require a SHELVE signature.

Unshelve alarms during runtime

Alarms are unshelved automatically at the end of the shelving time period. An unshelved alarm reappears in the Alarm Control active list and resumes its state at the time it was shelved. Operators can manually unshelve a shelved alarm before the end of the shelved period and enter an optional comment.

Hide alarms

The "hiding" and "unhiding" of alarm records is known in the corresponding InTouch alarm controls as "suppressing" and "unsuppressing".

When the Alarm Control is hiding alarms, it ignores certain alarms. If an alarm matches the exclusion criteria, it is not visible.

The actual alarm generation is completely unaffected by hiding. Alarm records are still logged into the alarm history.

As in the InTouch HMI, you can unhide specific alarms and also use properties and methods to interact with the alarm hiding feature at run time.

Freeze the Alarm Control grid

You can freeze the Alarm Control to prevent the Alarm control tree from being updated with any further changes.

For example, if new alarms occur while the Alarm Control is frozen, the new alarms are only shown after you unfreeze the Alarm Control.

You can configure a time period after which the Alarm Control automatically unfreezes to avoid the Alarm Control being unknowingly frozen. For example, the operator leaves the workstation and returns without realizing that the Alarm Control is still frozen.

The Alarm Control unfreezes automatically if one of the following changes:

- Alarm Mode
- Alarm Query
- Filter

Sort alarms

Like InTouch alarm controls, you can sort the alarms in ascending or descending direction for selected columns.

The Alarm Control supports alarm sorting for up to four columns at design time and run time. At run-time, the operator can configure sorting of even more columns by clicking on the column headers of the Alarm Control while pressing the Shift key.

Support for a redundant Historian server

A Historian may be configured to have a symmetrical "partner" Historian that can be used as a backup if the primary, or main, historian is not available. This is known as a "redundant historian" setup. No control configuration is required to take advantage of a redundant historian.

When the primary historian is unavailable, the Alarm Control automatically switches over to the configured partner historian. The control remains connected to the partner historian, even when the primary historian becomes available again. The Alarm Control switches back to an available primary historian if it fails to connect to the partner or during a new attempt to connect to the primary historian. One example of this would be restarting the Trend Control, which initiates a new connection to the primary historian. For a redundant historian setup, both historians must be configured to store events to the same location, either to history blocks or the A2ALMDB database.

When the Alarm Control successfully connects to either the primary historian or its partner, the following columns are updated with the connected historian server name in the tags list of the selected tags:

- Server
- I/O Address

The historian name shown in the Tag Picker is always the name of the primary historian, even when the control is connected to the partner.

There is no automatic synchronization built in to the redundant historian setup; it is up to the historian server administrator to make sure that the two historians in the pair are symmetrical and synchronized.

If the SQL Server Service is running while Historian Service is not running, this is not recognized by the Alarm Control as a scenario in which the Historian Server is unavailable.

Status bar

The status bar of the Alarm Control resembles the status bars of the InTouch alarm controls, with the following differences:

- Alarm Control also shows the alarm client time zone
- Alarm Control querying the Alarm Database has a Requery button to more easily retrieve data from the Alarm Database
- Alarm Control shows the current client mode as an icon

Configure the Alarm Client Control

This section shows you how to place an Alarm Client Control onto the canvas and configure it. You can configure it either with the Edit Animations dialog box, or by changing individual properties in the Properties Editor.

About Alarm Control configuration

After placing the Alarm Control onto the canvas, you can configure the:

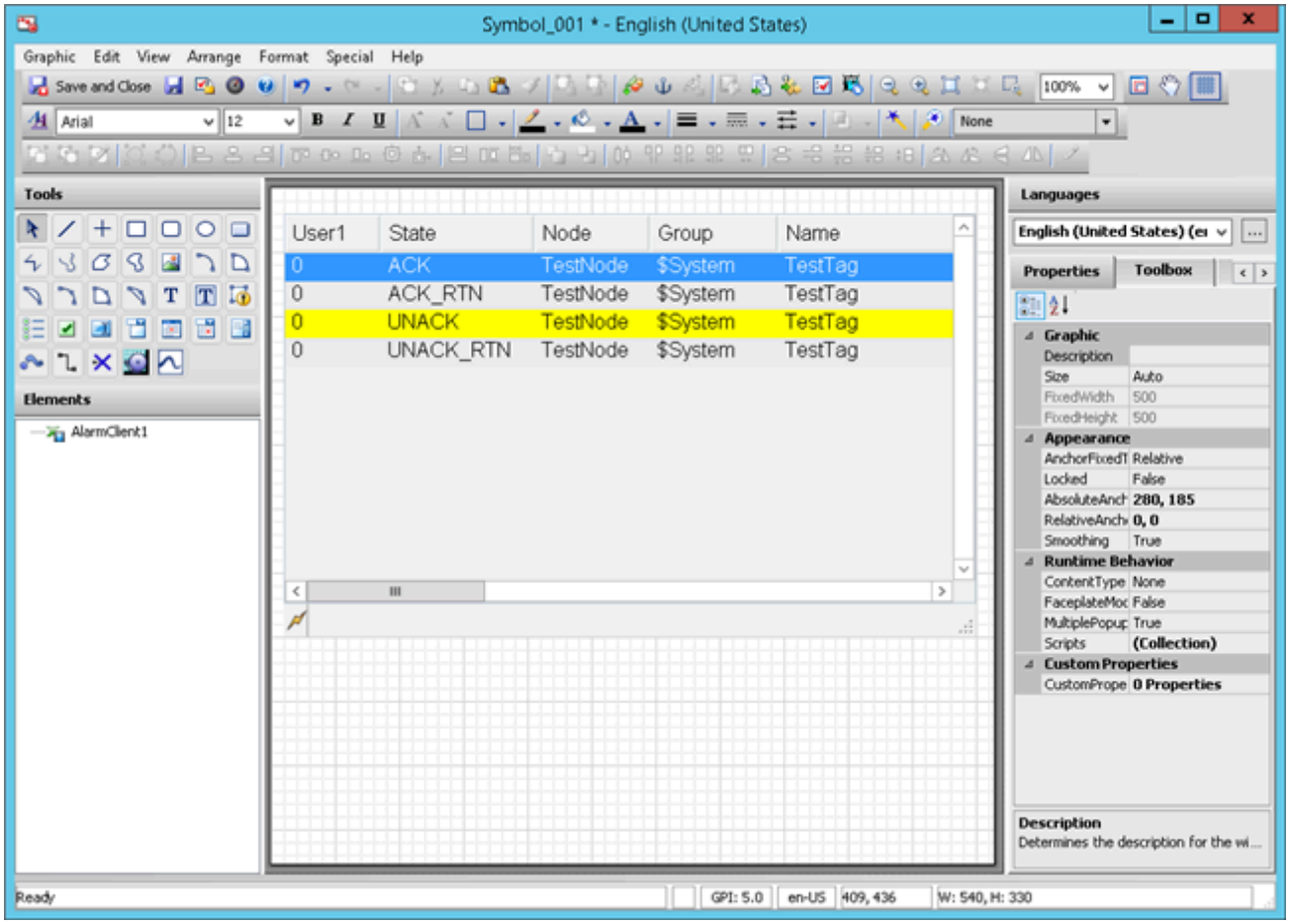
- Client Mode to show current alarms, recent alarms and events, or historical alarms and/or events.
- Colors for the Alarm Control grid, window, heading, and alarm records.
- Order and width of the grid columns and their headers.
- Sorting order of alarm records.
- Filtering for alarm records and save the filters as favorites for re-use.
- Time format and zone for the alarm record time stamps.
- Run-time behavior for the Alarm Control, such as:
  - If the operator can resize columns or select multiple records at run time.
  - Access to specified options of the shortcut menu at run time.

Place the Alarm Control into an industrial graphic

You can easily place the Alarm Control into an industrial graphic by placing it onto the canvas.

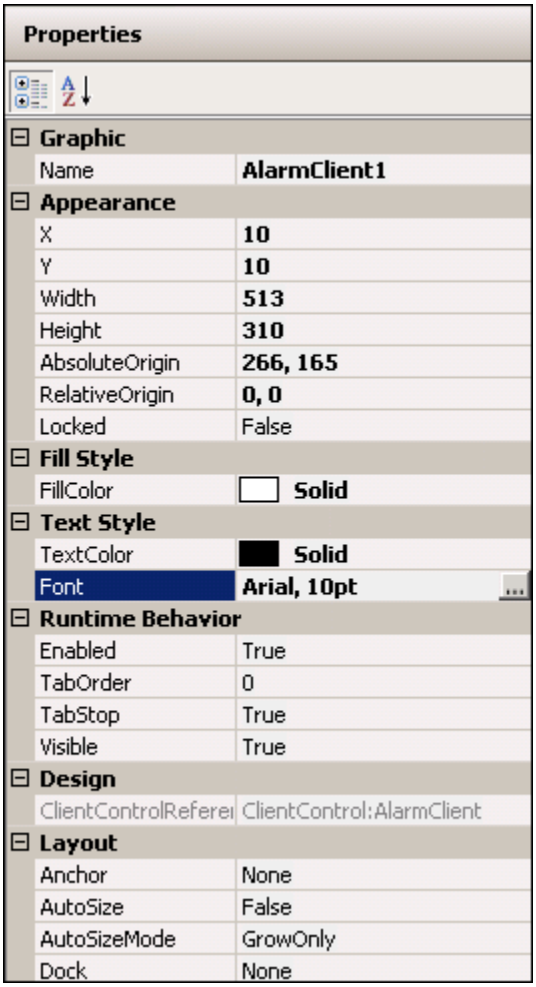
To place the Alarm Control into an industrial graphic

1. Open the Industrial graphic in the **Industrial Graphic Editor**.
2. On the **Tools** panel, click the **Alarm Client** icon. The cursor appears in insert mode.
3. Click the canvas where you want to place the Alarm Control.



Set Alarm Control properties

Like all other graphical objects in the Industrial Graphic Editor, you can set some of the properties of the selected Alarm Control directly in the Properties Editor.



We recommend you configure the Alarm Control with the **Edit Animations** dialog box and only use the **Properties Editor** to edit the configuration afterward.

Show current alarms or recent alarms and events

You can set the Alarm Control to show either of the following:

- Current alarms
- Recent alarms and events

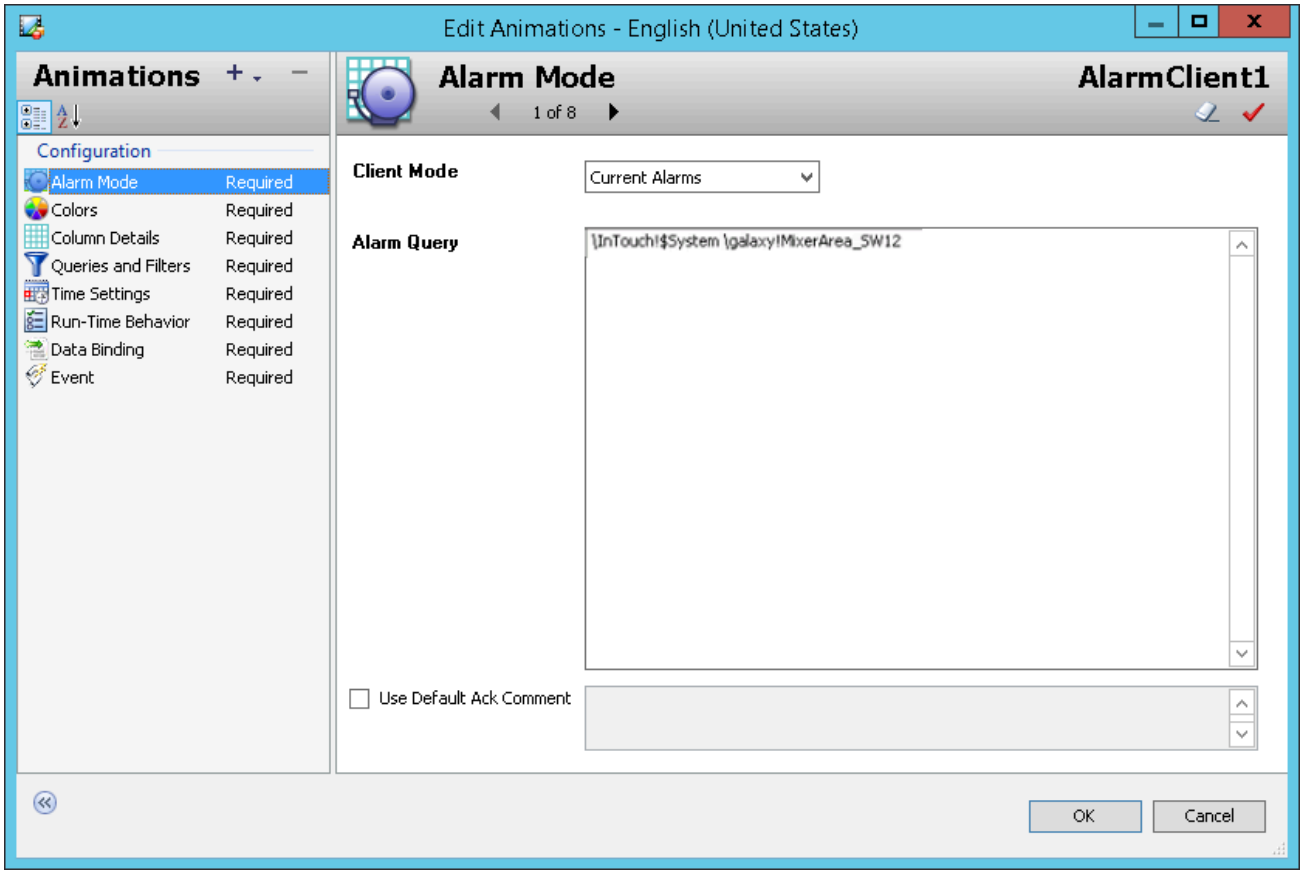
You use the ClientMode Property integer property in scripting to switch the Alarm Control to show current alarms or recent alarms and events at run time.

You can also configure a comment to use when alarms are acknowledged at run time. Use the AckComment.UseDefault Property Boolean property and AckComment.DefaultValue Property string property in scripting to use a default acknowledgement comment at run time.

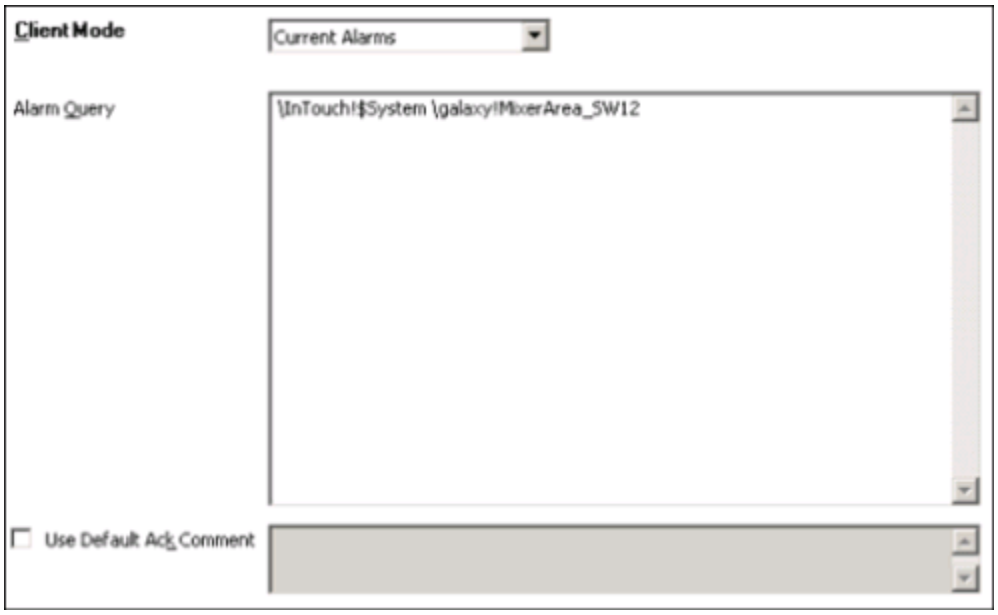
Note: When the Alarm Client Control is in the Recent Alarms and Events mode, the displayed alarms cannot be acknowledged.

To show current alarms

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.



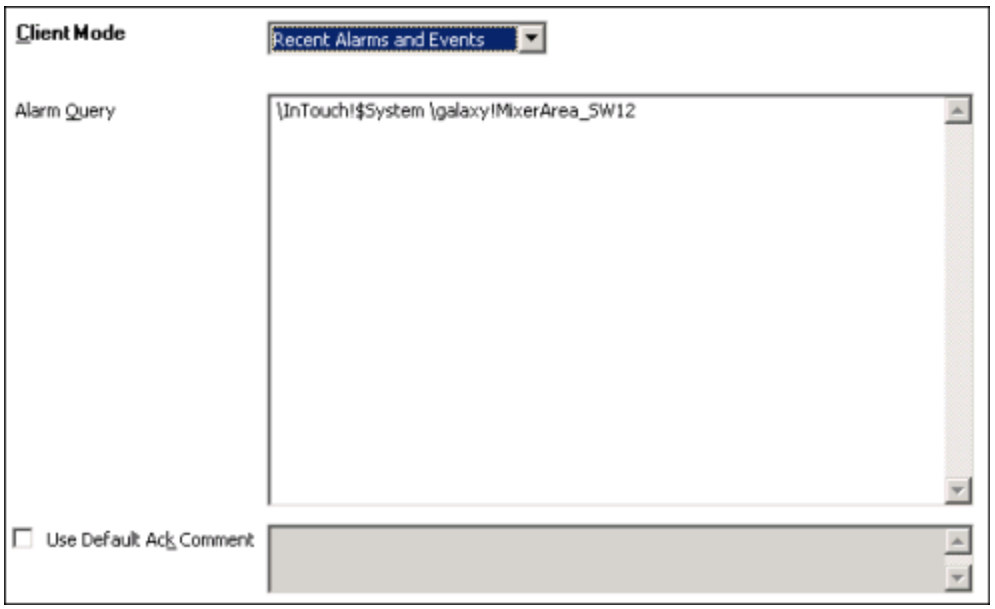
2. If necessary, click **Alarm Mode**. The **Alarm Mode** page appears.



3. In the **Client Mode** list, click **Current Alarms**.
4. In the **Alarm Query** box, type the alarm query. To create a new line in the Alarm Query box, press **Ctrl + Enter**. For more information on the valid syntax, see Alarm queries.
5. If you want to use a default acknowledgement comment, select the **Use Default Ack Comment** check box and type a comment in the text box.
6. Click **OK**.

To show recent alarms and events

1. Double-click Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Alarm Mode**. The **Alarm Mode** page appears.
3. In the **Client Mode** list, click **Recent Alarms and Events**.



4. In the **Alarm Query** box, type the alarm query. To create a new line in the **Alarm Query** box, press **Ctrl + Enter**.

   The alarm query must follow one of the following syntax:

   - \\node\provider!group
   - \provider!group
   - HotBackupName

   For example:

   ```
   \galaxy!Area_001
   ```

   For Alarm Controls hosted by Automation Object templates or instances, you can specify one of the following alarm queries:

   - \galaxy!<myArea.Tagname> to retrieve alarms and events from the Area object hosting the Automation Object template or instance.
   - \galaxy!<me.Area>!<me.Tagname>.* to retrieve alarms and events from the Automation Object template or instance.

   For more information on alarm queries, see Alarm queries.

5. If you want to use a default acknowledgement comment, select the **Use Default Ack Comment** check box and type a comment in the text box.
6. Click **OK**.

Show historical alarms and/or events

You can set the Alarm Control to show one of the following:

- Historical alarms from the Alarm Database
- Historical events from the Alarm Database
- Historical alarms and events from the Alarm Database
- Historical events from History Blocks
- Historical alarms from History Blocks
- Historical alarms and events from History Blocks

When you configure the Alarm Control to show historical alarms and/or events, you also configure the following:

- Server name hosting the Alarm Database
- Authentication information to connect to the Alarm Database
- Maximum number of records to retrieve from the Alarm Database
- Time range or duration to show in the Alarm Control
- If the Alarm Control should update to the current client time

Alarm Control works with both the ArchestrA Database (A2ALMDB) and with History Blocks.

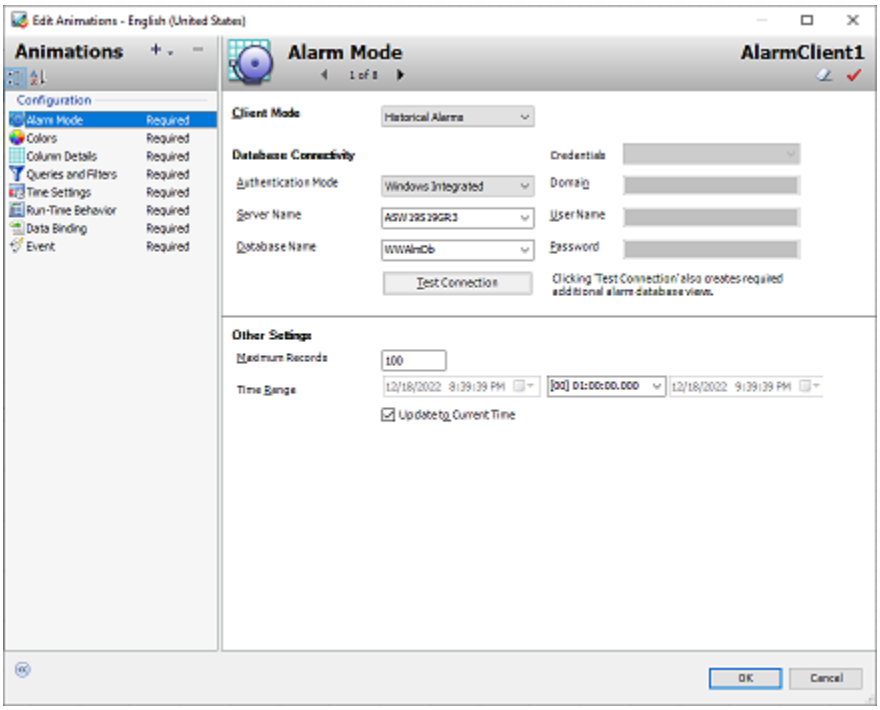
Use the following properties in scripting to switch the client mode and configure the database connection, such as:

- ClientMode property
- Database.Authentication property
- Database.Name property
- Database.Password property
- Database.ServerName property
- Database.UserID property
- Domain property

To show historical alarms and/or events

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Alarm Mode**. The **Alarm Mode** page appears.
3. In the **Client Mode** list, click:
   - **Historical Alarms** to only show alarms from the Alarm Database. No events are shown
   - **Historical Events** to only show events from the Alarm Database. No alarms are shown

- **Historical Alarms and Events** to show both alarms and events from the Alarm Database



4. In the **Authentication Mode** list, click one of the following:
   - **Windows Integrated** to use the authentication of the currently logged-on Windows user
   - **Windows Account** to use a given Windows user authentication
   - **SQL Server** to use SQL Server authentication mode
5. In the **Server Name** list, either select or type the name of the server hosting the Alarm Database.

   Provide the port number with the server name when connecting to a non-default port. For example: <servername>,23646. The port number can be changed in the Configurator.
6. In the **Database Name** box, type the name of the Alarm Database. For the ArchestrA Database, enter A2ALMDB, and for Historian block storage enter History Blocks.

   For the History Blocks option to function correctly, the REST Details > HTTP port (Default 32569) and HTTPS port (Default 32573) on the Historian Server needs to be opened in the firewall, and configured for both inbound and outbound traffic. For more information, see the Configuring Databases and Data File Locations in the System Platform Installation Guide.

   Note: For History Blocks, select Windows Integrated or Windows Account as the authentication mode.
7. From the **Credentials** drop-down, select the credentials for authentication.The Credentials field is enabled only when you select **Windows account** or **SQL Server** authentication type. The Windows Integrated authentication method uses the credentials of the user currently logged in, and disables the Credentials field.

   Note: For standalone InTouch applications, the credentials are retrieved from the Application Manager. For managed InTouch applications, the credentials are retrieved from the Credential Manager of the Application Server. For more information, see Work with Credential Manager in the AVEVA™ InTouch HMI Application Development Guide.

   You can also select **Use username and password** to enter the credentials.

   a. If you are using **Windows Account** authentication mode, type the domain, user name, and password in

the **Domain**, **User Name** and **Password** boxes.

b. If you are using **SQL Server** authentication mode, type user name and password in the **User Name** and **Password** boxes.

8. Click **Test Connection**. The connection to the Alarm Database is tested and a result message appears. If necessary, check your authentication information.

   If the Historian requires a secure connection and the client is not configured with the correct certificates, then an error message is displayed.

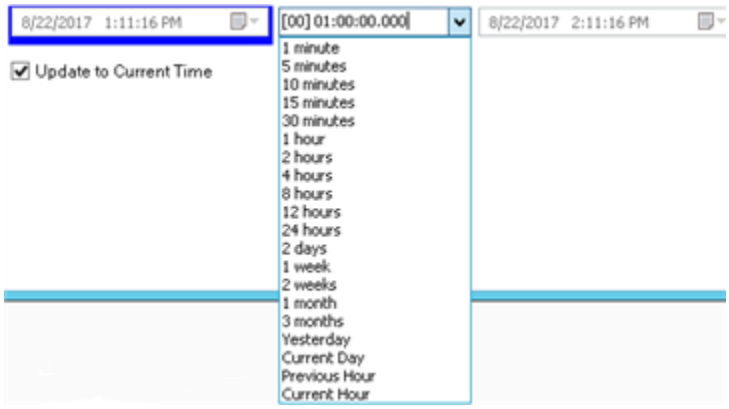
9. Click **OK**.

To set maximum records and time range

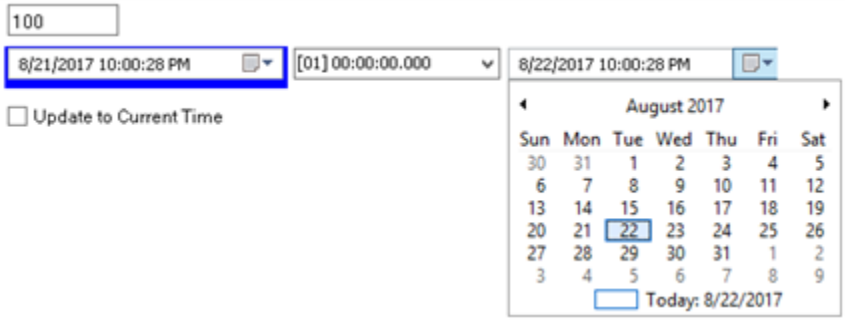
1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Alarm Mode**. The **Alarm Mode** page appears.
3. Make sure the **Client Mode** is set to **Historical Alarms**, **Historical Events**, or **Historical Alarms and Events**.
4. In the **Maximum Records** box, type the number of records to view from the control at one instance. The valid range of maximum records is from 1 to 32766.

   You can also use the MaxDatabaseRecords Property in scripting to set the maximum records at run time.

5. To use a pre-defined time interval, select an interval from the middle list of the **Time Range** pickers.



6. To use a specific start time and end time, clear **Update to Current Time**, and select the start time from the list at the left and the end time from the list at the right of the **Time Range** pickers.



   You can also use the **TimeSelector.*** methods and properties in scripting to set the start date, end date, or duration at run time. For more information, see TimeSelector property.

7. Click **OK**.

Set Alarm Control colors

You can show different types of alarm records with different colors to more easily identify certain types of alarms.

You can configure the Alarm Control with priority breakpoints to show alarm records within the resulting priority ranges in different colors.

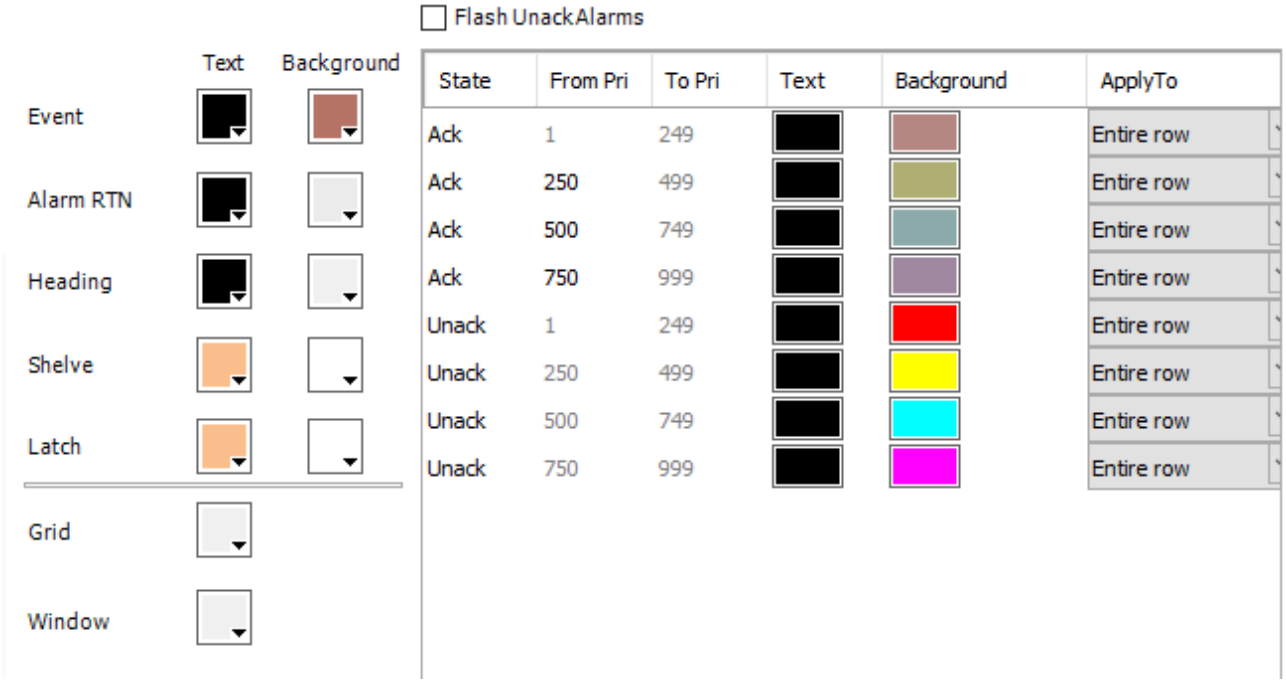
You can also configure the control background color, the grid color, and the heading colors.

Set event record colors

You can set text color and background color for event records. Use the EventColor.ForeGround Property and EventColor.BackGround Property properties in scripting to set the event alarm record text color and background color at run time.

To set text and background colors for event records

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click the **Colors**. The **Colors** page appears.



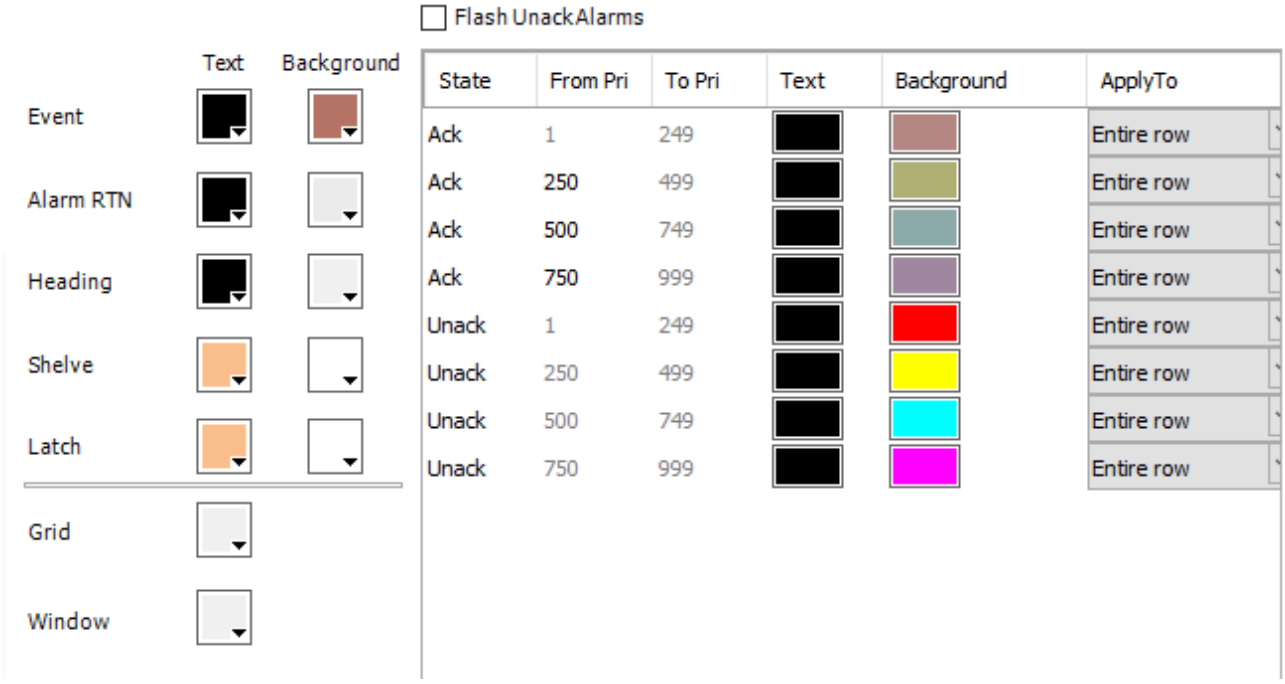
3. Configure the event record text color. Do the following:
   a. Click the color field next to **Event** and under **Text**. The color picker appears.
   b. Select a color and click **OK**.
4. Configure the event record background color. Do the following:
   a. Click the color field next to **Event** and under **Background**. The color picker appears.
   b. Select a color and click **OK**.
5. Click **OK**.

Set alarm return to normal record colors

You can set text color and background color for "return to normal" alarm records. Use the AlarmColor.Ack.RTN.ForeGround Property and AlarmColor.Ack.RTN.BackGround Property properties in scripting to set the "return to normal" alarm record text color and background color at run time.

To set text and background colors for "return to normal" records

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.



2. Click **Colors**. The **Colors** page appears.
3. Configure the "return to normal" record text color. Do the following:
   a. Click the color field next to **Alarm RTN** and under **Text**. The color picker appears.
   b. Select a color and click **OK**.
4. Configure the "return to normal" record background color. Do the following:
   a. Click the color field next to **Alarm RTN** and under **Background**. The color picker appears.
   b. Select a color and click **OK**.
5. Click **OK**.

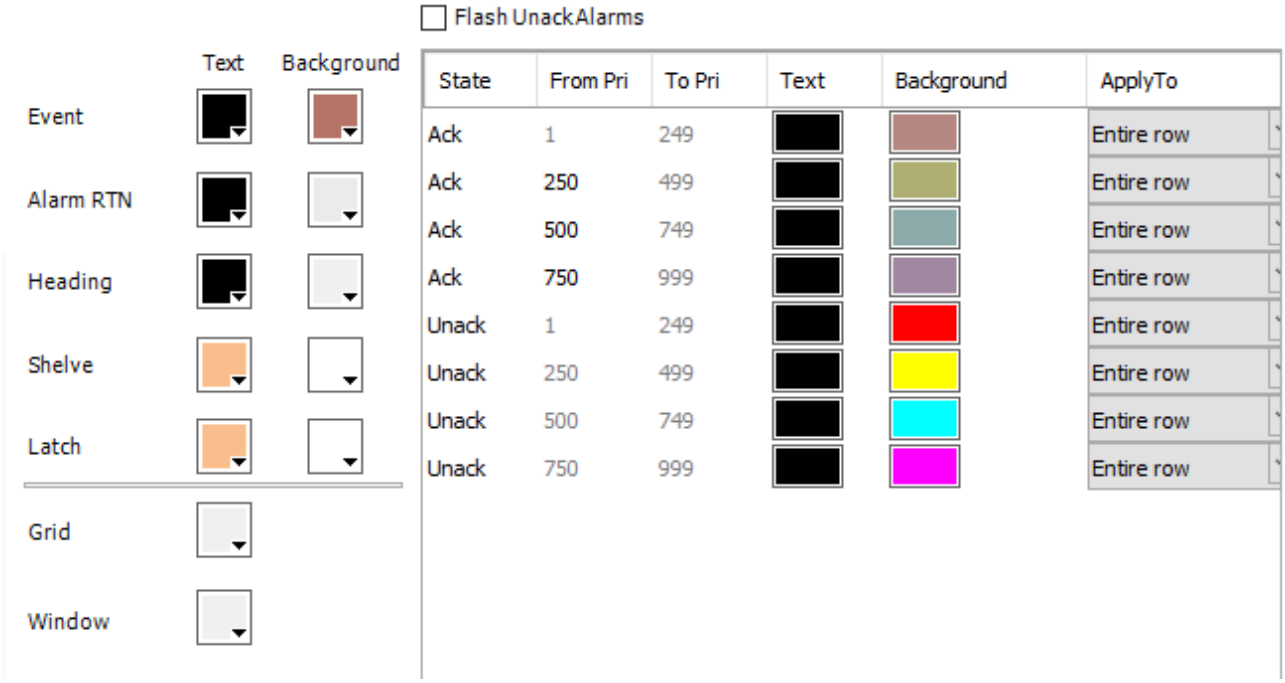
Set heading, grid, and window color

You can set text color and background color for the heading, the grid color, and the Alarm Control window color. Use the corresponding HeadingColor.ForeGround Property, HeadingColor.BackGround Property, GridColor Property, and WindowColor Property properties in scripting to set the colors for heading, grid, and window.

To set heading, grid, and window color for the Alarm Control

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.

2. Click **Colors**. The **Colors** page appears.



3. Do one of the following:
   a. Configure the heading text color by clicking the color box next to **Heading** and under **Text**. If the color box does not open, you need to select the **Show Heading** option on the **Run-Time Behavior** page first.
   b. Configure the heading background color by clicking the color box next to **Heading** and under **Background.** If the color box does not open, you need to select the **Show Heading** option on the **Run-Time Behavior** page first.
   c. Configure the grid color by clicking the color box next to **Grid**. If the color box does not open, you need to select the **Show Grid** option on the **Run-Time Behavior** page first.
   d. Configure the window color by clicking the color box next to **Window.**

Set shelved alarm colors

You can set text color and background color for alarms that are temporarily shelved.

To set alarm shelved colors

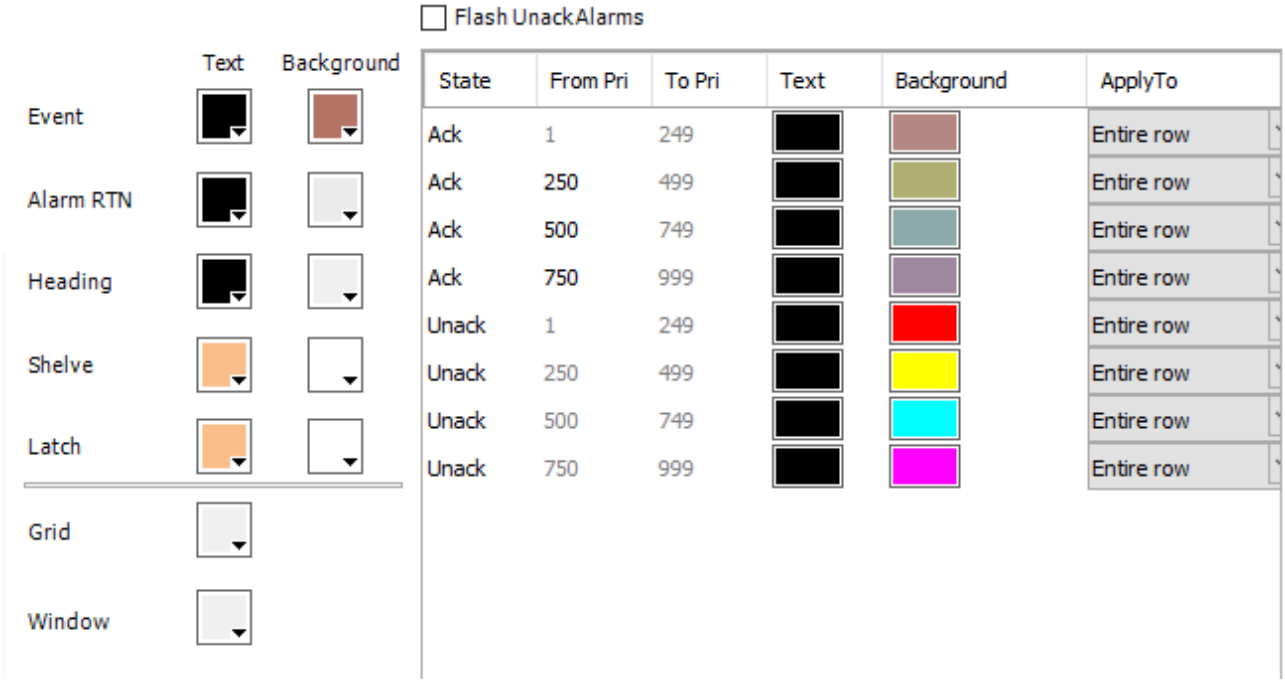
1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Colors**. The **Colors** page appears.
3. Configure the shelve record text color. Do the following:
   a. Click the color field next to **Shelve** and under **Text**. The color picker appears.
   b. Select a color and click **OK**.
4. Configure the shelve record background color. Do the following:
   a. Click the color field next to **Shelve** and under **Background**. The color picker appears.
   b. Select a color and click **OK**.
5. Click **OK**.

Set LATCHED state record colors

You can set text color and background color for the LATCHED alarm records. Use the LatchColor.ForeGround Property and LatchColor.BackGround Property properties in scripting to set the LATCHED state alarm record text color and background color at run time.

To set text and background colors for LATCHED alarm records

1. Double-click the **Alarm Control** on the canvas. The **Edit Animations** dialog box appears.
2. Click the **Colors**. The **Colors page** appears.



3. Configure the LATCHED state alarm record text color. Do the following:
   a. Click the color field next to **Latch** and under **Text**. The color picker appears.
   b. Select a color and click **OK**.
4. Configure the Latched State alarm record background color. Do the following:
   a. Click the color field next to **Latch** and under **Background**. The color picker appears.
   b. Select a color and click **OK**.
5. Click **OK**.

Set priority ranges for alarm records

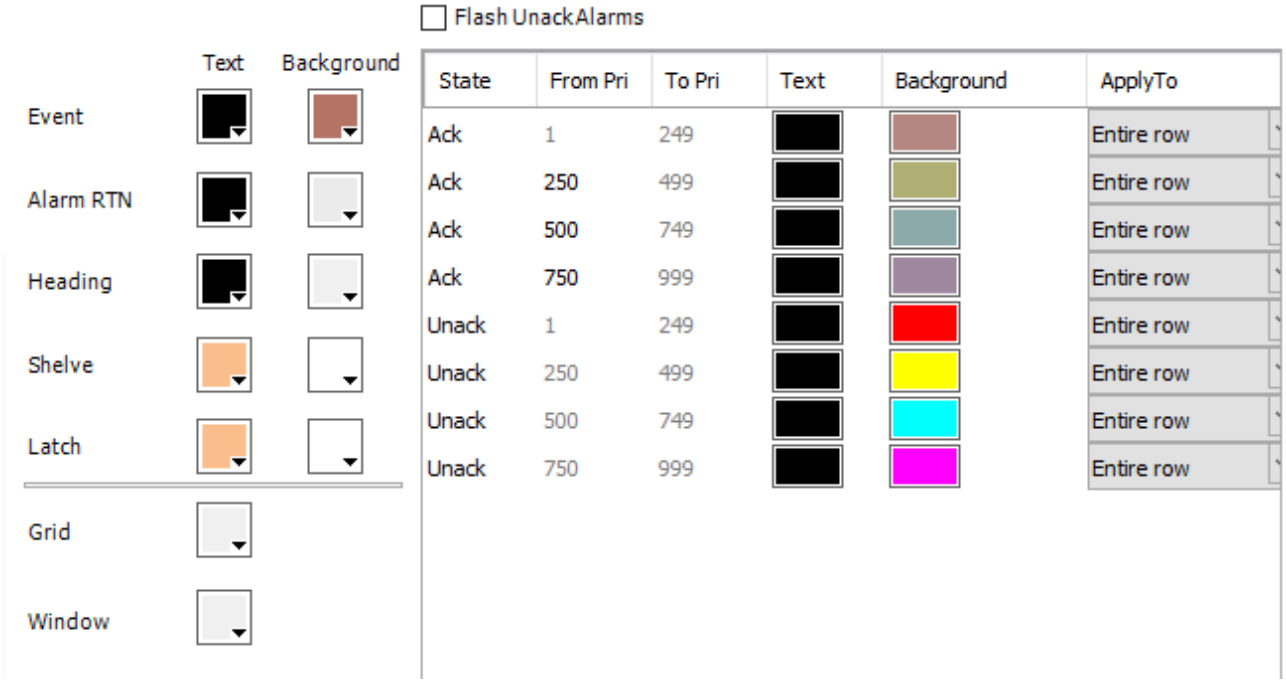
You can use alarm priority ranges to filter alarms. The Alarm Control can show alarms within a given range with a different text and background color. Use the AlarmColor.Range Property group in scripting to set the breakpoints at run time.

The Alarm Control supports four alarm ranges defined by three breakpoints:

1 < breakpoint 1 < breakpoint 2 < breakpoint 3 < 999

To set priority ranges for alarm records

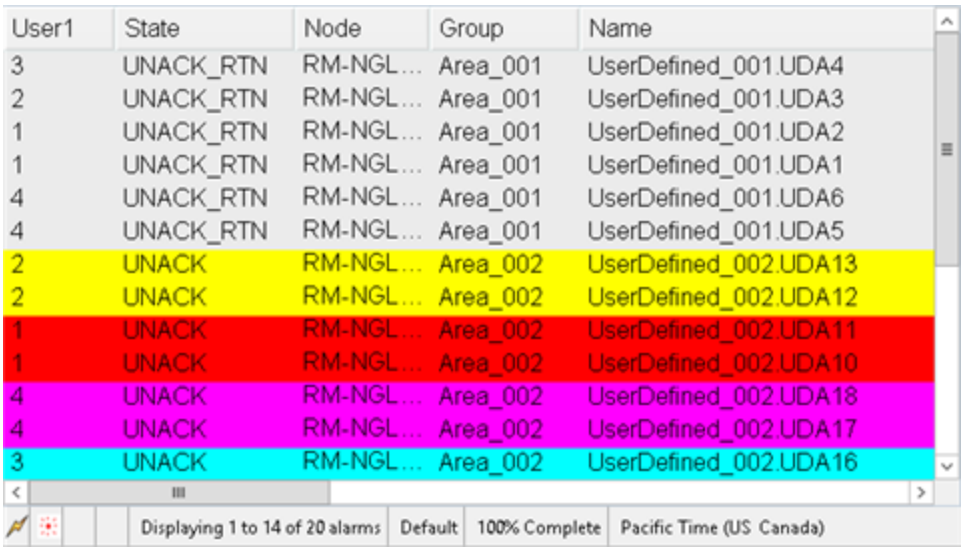
1. Double-click the Alarm Control on the canvas. The Edit Animations dialog box appears.
2. Click **Colors**. The **Colors** page appears.
3. In the **From Pri** column in the list at the right, locate the break point you want to change. These are values except 1 or 999.
4. Click the value and type a new value in the range between the previous breakpoint and the next breakpoint.



5. Press **Enter**. All priority values in the list are updated.
6. Click **OK**.

Example

If you use the color configuration in the procedure above, the Alarm Control at run time could have following appearance:

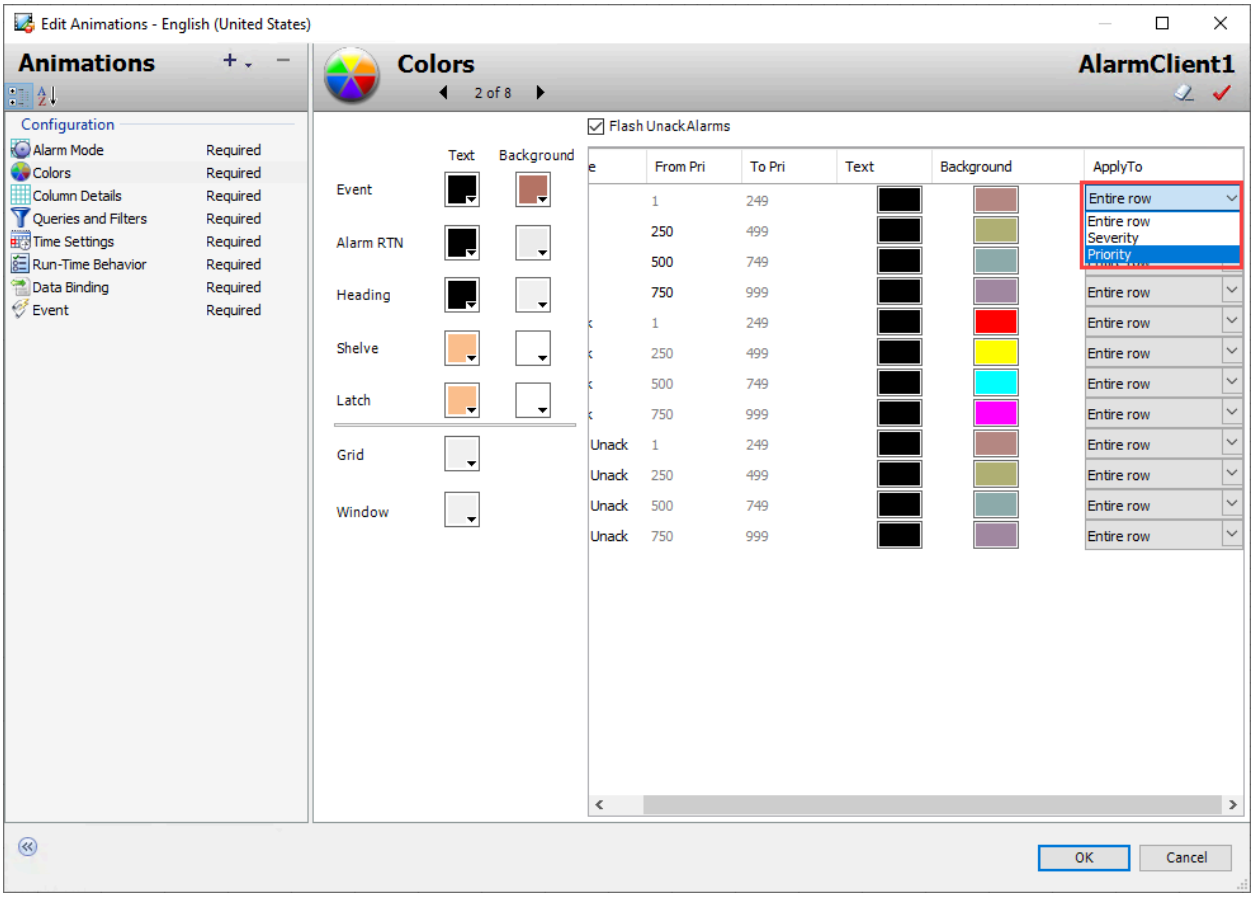


Set colors for acknowledged alarms

You can set the text and background colors for records of acknowledged alarms. For each of the priority ranges, you can set a text color and a background color. Use the AlarmColor.Ack.ForeGround Property and AlarmColor.Ack.BackGround Property property groups in scripting to set the text color and background color for acknowledged alarms in each priority range at run time.

To set colors for acknowledged alarm records

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Colors**. The **Colors** page appears.
3. In the list at the right, locate the **Ack** record and priority range for which you want to change the text or background color.
4. Click the color box in the **Text** or **Background** column of the line. The color picker appears.
5. Select a color and click **OK**.
6. In the **ApplyTo** column of the line, click the dropdown arrow to select one of the following options:
   - **Entire row** - highlights the entire row.
   - **Severity** - highlights only the Severity column.
   - **Priority** - highlights only the Priority column.



By default **Entire row** is selected. Note that Severity column is available only in Historical Mode with History Blocks.

7. Click **OK**.

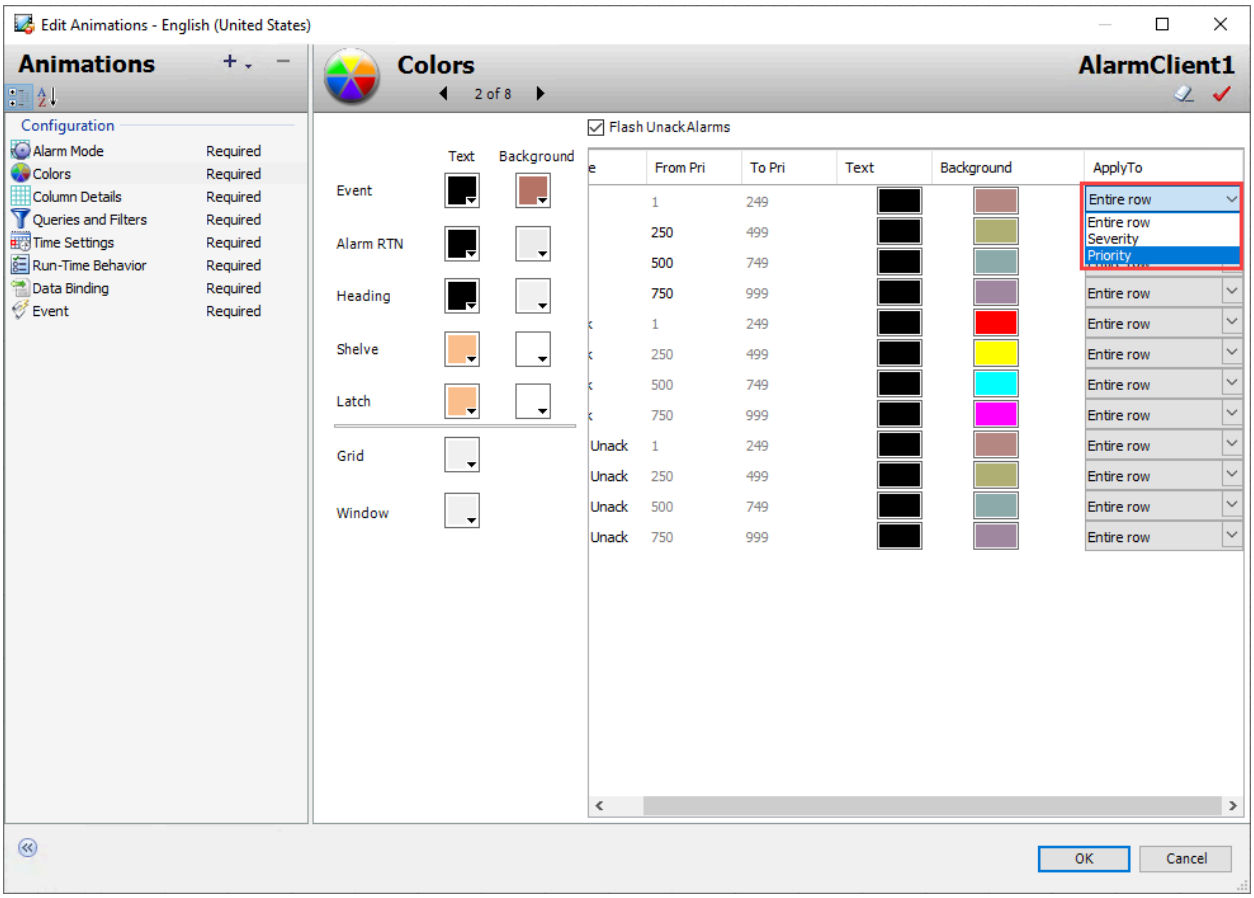
Set colors for unacknowledged alarms

You can set the text and background colors for records of unacknowledged alarms. For each of the priority ranges, you can set a text color and a background color. Use the AlarmColor.UnAck.ForeGround Property and AlarmColor.UnAck.BackGround Property property groups in scripting to set the text color and background color for unacknowledged alarms in each priority range at run time.

To set colors for unacknowledged alarm records

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Colors**. The **Colors** page appears.
3. In the list at the right, locate the **Unack** record and priority range for which you want to change the text or background color.
4. Click the color box in the **Text** or **Background** column of the line. The color picker appears.
5. Select a color and click **OK**.
6. In the **ApplyTo** column of the line, click the dropdown arrow to select one of the following options:
   - **Entire row** - highlights the entire row.

- **Severity** - highlights only the Severity column.
- **Priority** - highlights only the Priority column.



By default **Entire row** is selected. Note that Severity column is available only in Historical Mode with History Blocks.

7. Click **OK**.

Set unacknowledged alarms to flash

Instead of showing unacknowledged alarm records in predefined constant text and background color, you can configure the Alarm Control to flash unacknowledged alarms in another text and background colors.

Note: The Flash Unack Alarms setting is checked by default for instances of the Alarm Control that are embedded in Situational Awareness Library symbols.

The unacknowledged alarm records flash between the colors of the Unack alarms and the colors of the Flash Unack alarms. Use the FlashUnAckAlarms Property Boolean property in scripting to set unacknowledged alarm records to flash at run time.

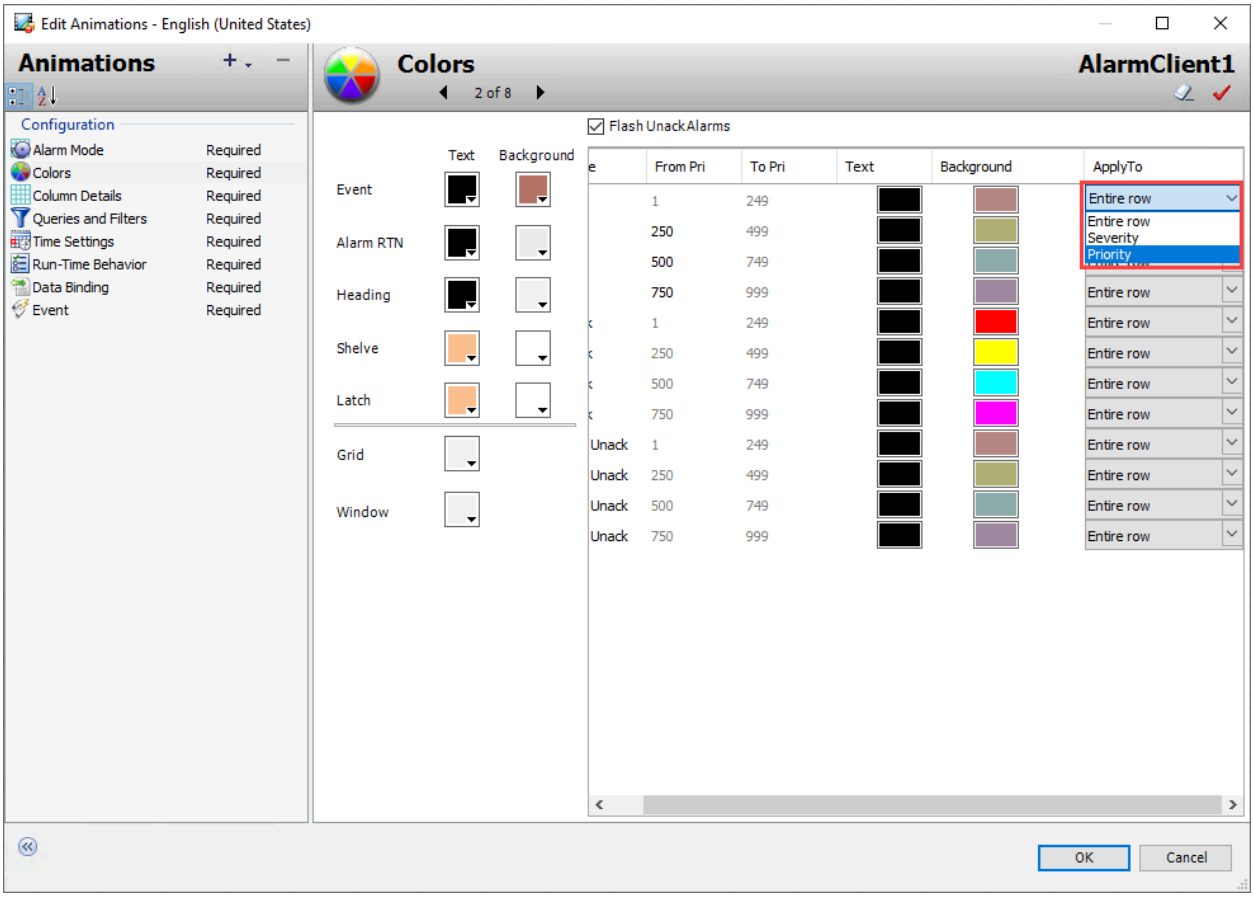
Use the AlarmColor.UnAck.Flash.ForeGround Property and AlarmColor.UnAck.Flash.BackGround Property property groups in scripting to set the text color and background color for flashing unacknowledged alarms in each priority range at run time.

To set flashing and colors for unacknowledged alarm records

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.

2. Click **Colors**. The **Colors** page appears.
3. Select the **Flash Unack Alarms** check box.

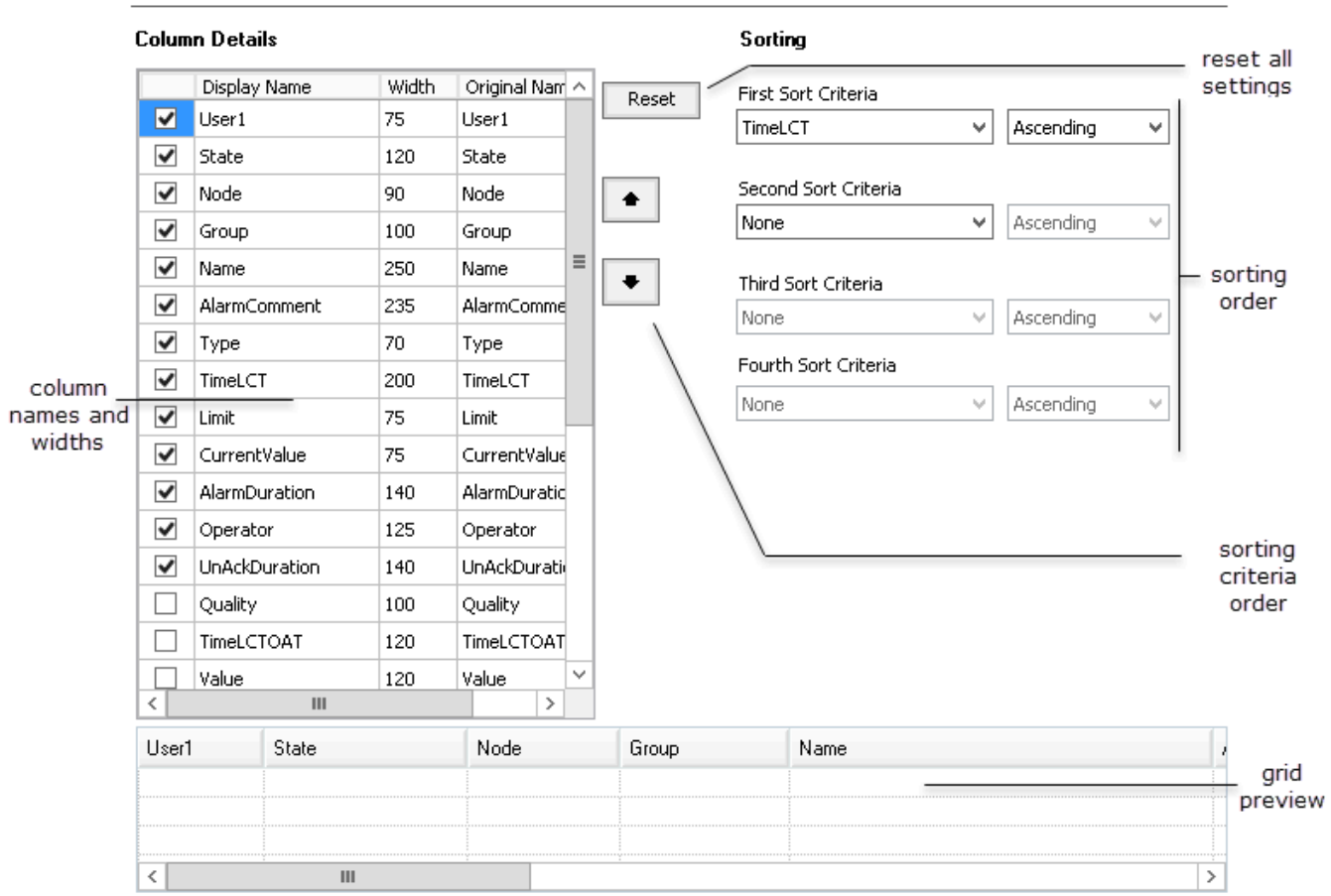
   Note: You cannot select the **Flash UnAck Alarms** check box if the client mode is set to one of the historical modes.
4. In the list on the right, locate the **Unack** record and priority range for which you want to change the text or background color. Do the following:
   a. Click the color box in the **Text** or **Background** column of the line. The color picker appears.
   b. Select a color and click **OK**.
5. Locate the **Flash Unack** record and priority range for which you want to change the text or background color. Do the following:
   a. Click the color box in the **Text** or **Background** column of the line. The color picker appears.
   b. Select a color and click **OK**.
6. In the **ApplyTo** column of the line, click the dropdown arrow to select one of the following options:
   - **Entire row** - highlights the entire row.
   - **Severity** - highlights only the Severity column.
   - **Priority** - highlights only the Priority column.



   By default **Entire row** is selected. Note that Severity column is available only in Historical Mode with History Blocks.
7. Click **OK**.

Rename, resize, and reorder column headers

You can rename, resize, and change the order of column headers in the Alarm Control.



All changes you make in the Column Details list are shown in the grid preview.

You can also use the grid preview to resize columns or change their order with the pointer.

Column headers can be localized along with other symbol text when you export, translate, and reimport language files. The translated language files must be imported to the InTouch HMI for run-time language switching. For further information, see *Working with Languages* in the *Application Server User's Guide*.

Important: If you rename or reorder column headers, you must repeat the symbol text translation procedures. If you do not, your changes will not be available for run-time language switching.

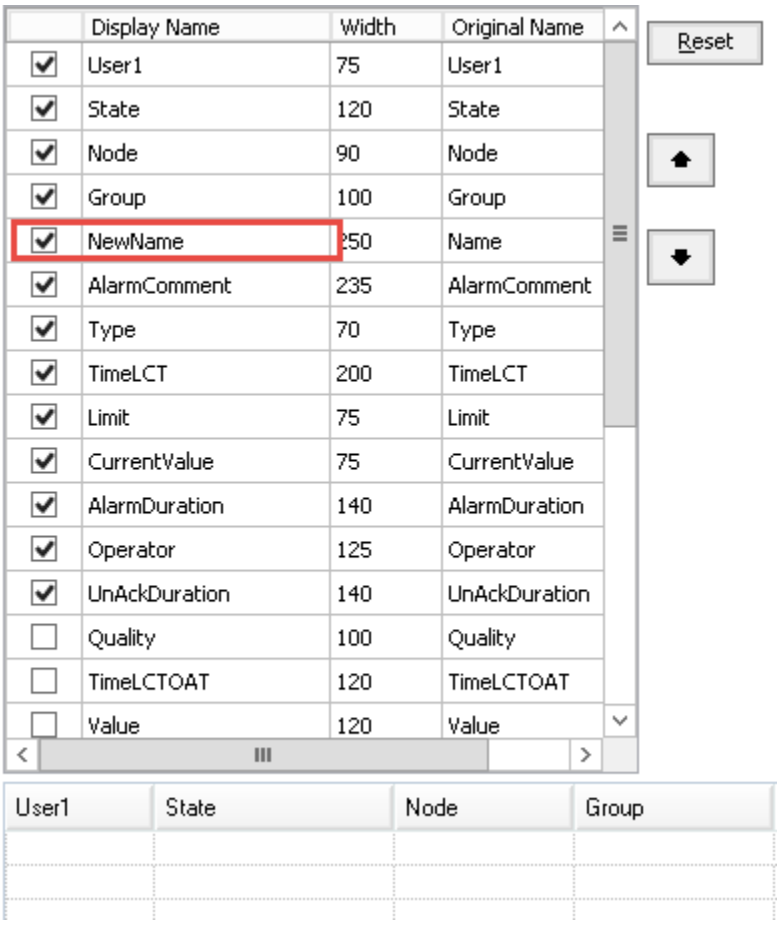
Rename column headers

You can rename the column headers in the Alarm Control.

To rename column headers

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Column Details** tab. The **Column Details** page appears.
3. In the **Column Details** list, locate the column header you want to rename and click on it.

4. Type a new name and press **Enter**. The **Column Details** list and the grid preview are updated with the new name.



5. Click **OK**.

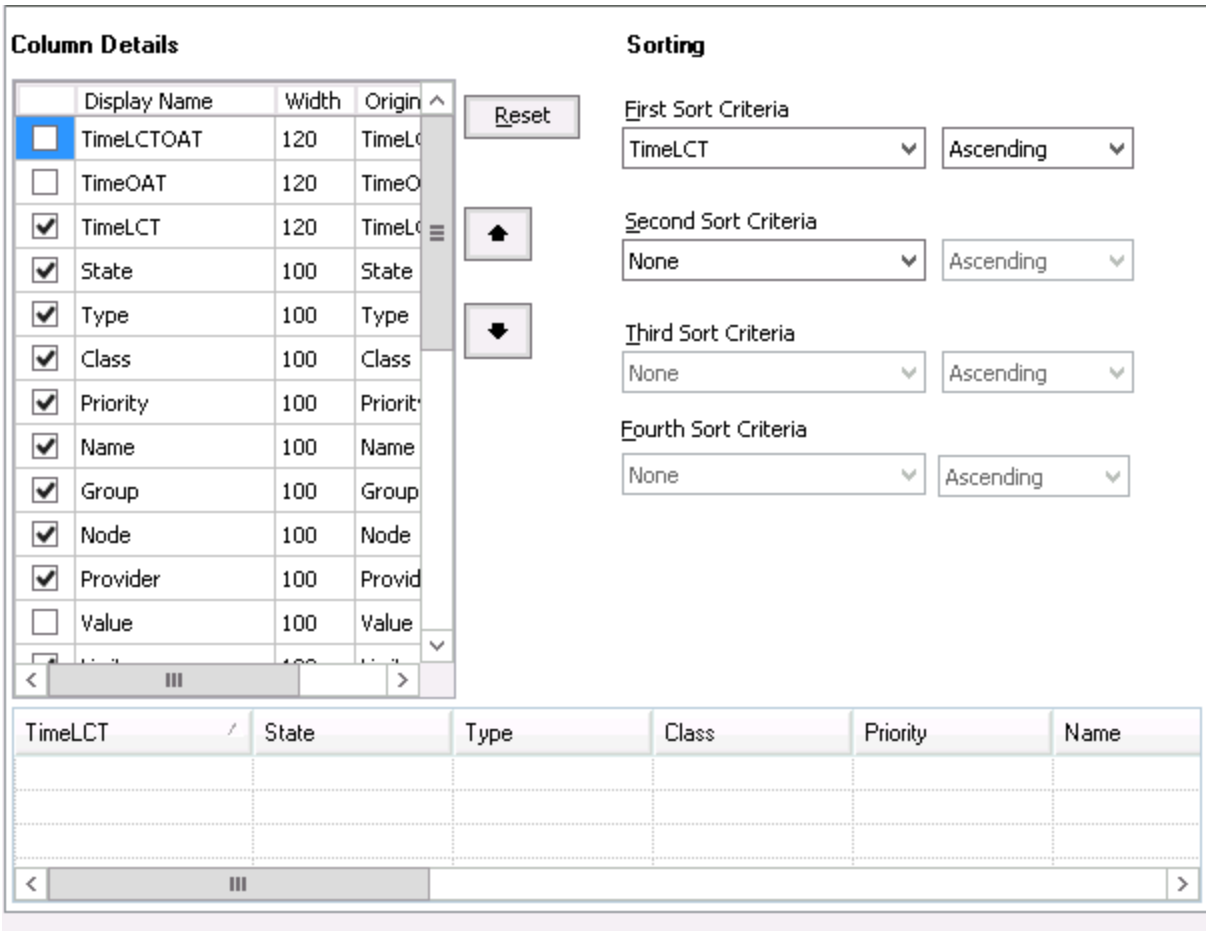
Resize columns

You can resize the column headers in the Alarm Control either by:

- Typing in a numeric value
- Dragging the column header boundary width with the pointer in the grid preview

To resize the column numerically

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Column Details**. The **Column Details** page appears.
3. In the **Column Details** list, locate the name of the column you want to resize and click on the **Width** value in the row.
4. Type a new width in pixels and press **Enter**. The **Column Details** list and the grid preview are updated.



5. Click **OK**.

To resize the column graphically

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Column Details**. The **Column Details** page appears.
3. In the grid preview, locate the column you want to resize and drag the column boundary to resize the column. The width value of the **Column Details** list is updated.
4. Click **OK**.

Change the order of columns

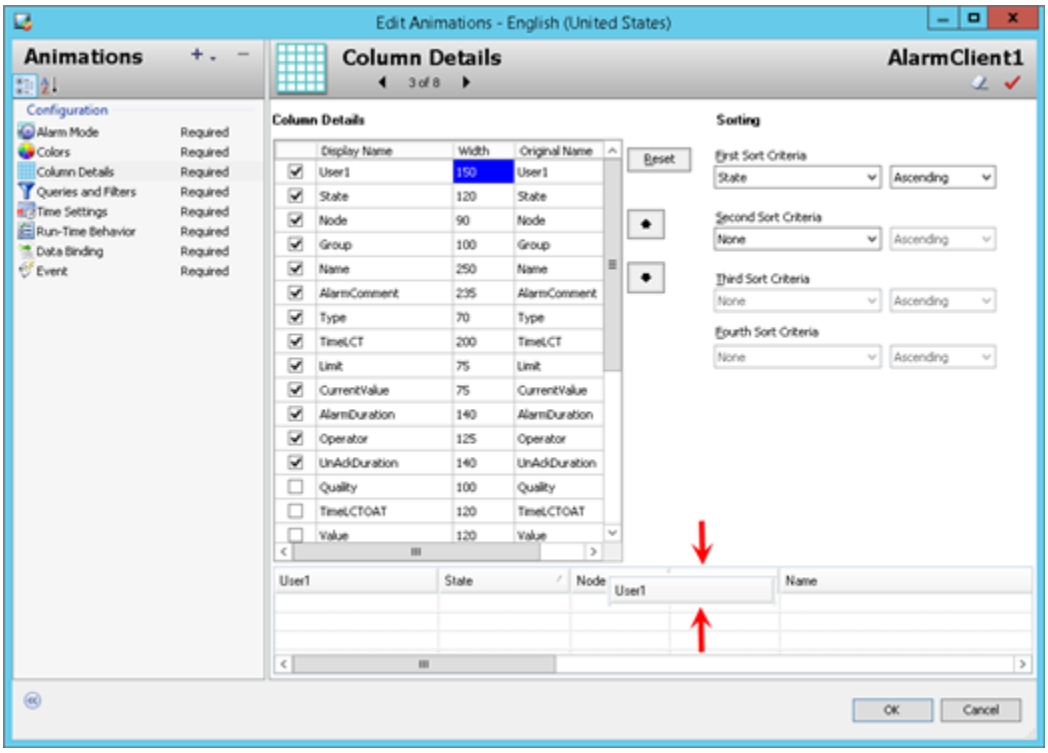
You can change the order of the columns in the Alarm Control by:

- Moving column names up and down in the **Column Details** list using buttons
- Dragging the column header with the pointer in the grid preview

You also can reset the column widths and order to their default values. Resetting the column widths and order also resets the names to their default values.

To change the column order

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Column Details**. The **Column Details** page appears.
3. Do one of the following:
   - Click arrow up and arrow down to reposition the columns
   - In the grid preview, drag the name of the column you want to reposition and drop it to the left of another column to reposition it



   The grid preview and the **Column Details** list shows the new column order.
4. Click **OK**.

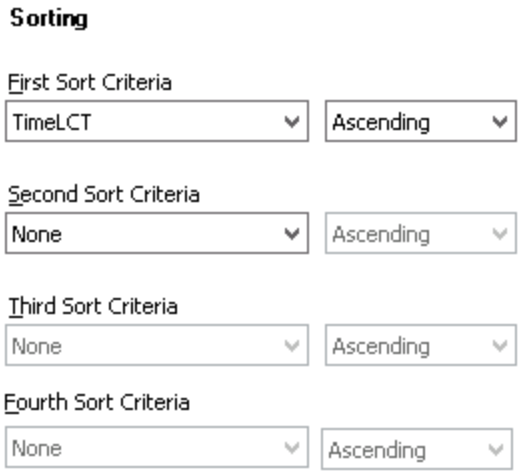
To reset column widths and order

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Column Details**. The **Column Details** page appears.
3. Click **Reset**. The column widths, names, and order are reset to their default values.
4. Click **OK**.

Configure alarm sorting

You can configure how the Alarm Control sorts alarm records at run time. By default, the Alarm Control lists alarm records by time in ascending order.

You can sort alarm records in ascending or descending order based on a first sort criteria, an optional second sort criteria, an optional third sort criteria, and and optional fourth sort criteria.



You can configure the sorting columns and directions either in lists or with the grid preview. Use the SortColumn.First Property, SortColumn.Second Property, and SortColumn.Third Property properties in scripting to set the columns to be sorted at run time. Use the SortOrder.First Property, SortOrder.Second Property, and SortOrder.Third Property properties in scripting to set the sort direction for each at run time.

To set sorting columns and directions with lists

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Column Details**. The **Column Details** page appears.
3. In the **Sorting** area, do the following:
   a. Select the primary sort column in the **First Sort Column** list and a sorting direction in the list to its right.
   b. Optionally, select the secondary sort column in the **Second Sort Column** list and a sorting direction in the list to its right.
   c. If you set the **Second Sort Column**, optionally select the tertiary sort column in the **Third Sort Column** list and a sorting direction in the list to its right.

   The grid preview is updated and shows arrows for the sorted columns and their sort directions.

To set sorting columns and directions with the grid preview

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Column Details**. The **Column Details** page appears.
3. In the grid preview, click on a column to select it for sorting. An arrow appears on the column header and the change is also shown in the **Sorting** area lists.
4. To change the sorting direction, click on the column header again. The arrow changes on the column header and the change is also shown in the **Sorting** area lists.

Note: If you click on a column header after releasing the **Shift** key, all sorting information is lost and the

selected column is the new primary sorting criteria.

5. To set second, third, and fourth sorting, hold the **Shift** key and repeat from step 3.
6. Release the **Shift** key.
7. Click **OK**.

Filter alarms

You can filter current and historical alarms by using queries and filters. Queries and filters are collections of filter criteria in a logical construct.

For example, you can filter alarms by defining a query or a filter that only shows alarms with priorities larger than 500 and smaller than 750.

You can re-use the filters and queries you define for historical alarms for current alarms and vice versa. You can also re-use filters and queries you define at design time at run time and vice versa.

Important: Queries and filters for current alarms and recent alarms and events require at least a **Provider** and **Group** as filter criteria. These must use the equals sign or a message asking for those fields will appear.

When you use TimeLCT, TimeOAT, or TimeLCTOAT as filter criteria for historical alarm modes, you need make sure that the TimeSelector.StartDate and TimeSelector.EndDate properties do not limit the query. Otherwise the Alarm Control can possibly not return all alarm and event records.

Set the TimeSelector.StartDate property earlier than any time filtering requirement, and the TimeSelector.EndDate later than any time filtering requirement.

Wildcards in queries

In current alarm queries, you can use wildcards only in the Tagname part of the query and not in the Provider, Group, or Node part of the query. A valid example is:

```
\galaxy!Mixing!RotorBlade*
```

In query filters that are used for current queries, the same restrictions apply.

In query filters that are used for historical queries, you must convert the operator and wildcard to SQL syntax according to the following table:

	Current Query	**Historical Query**
Operator	=	Like
Wildcard	*	%

For example:

```
Provider = ‘galaxy’ AND Group = ‘Mixing’
AND Name Like ‘RotorBlade%’
```

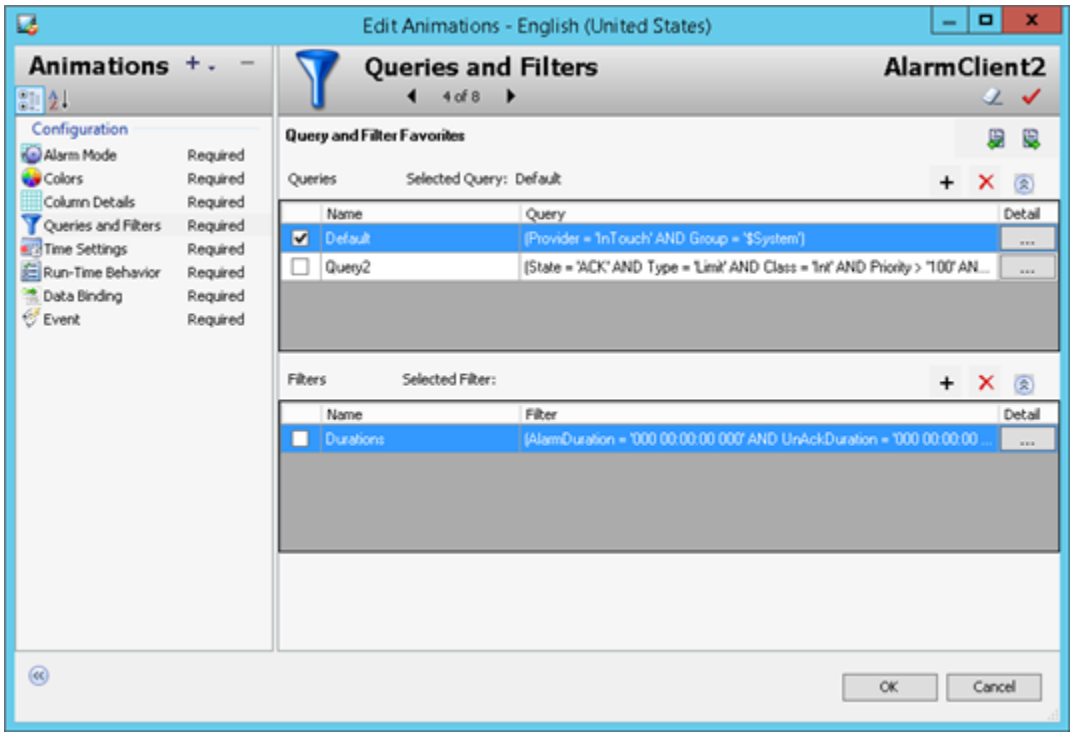
If you want to use a query filter containing a wildcard for a current query and a historical query, create two separate query filters. If a column contains null data, it cannot be retrieved using the Like operator.

Use an existing query or filter

You can use an existing query filter to filter the alarms shown in the Alarm Client Control. You can also use the Favorite Property string property in scripting to switch to an existing query filter at run-time.

To use an existing query filter

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Queries and Filters**. The **Query and Filter Favorites** page appears.
3. In the **Query and Filter Favorites** list, select a query or filter by clicking the check box before each query or filter name.



4. Click **OK**.

Note: In Current Alarms mode, if you try to proceed from **Queries and Filters** page with no query or filter selected, you will be prompted with a message to confirm you want to continue. Upon click of Yes the default query will be selected and you can continue. Upon click of No, you will be returned to the Queries and Filters page. If a query is selected but no filter, no message will display.

In Historical Alarms mode, if you proceed without a query selected but have selected a filter, the filter will be applied. If neither query nor filter is selected, you will be prompted with the same message.

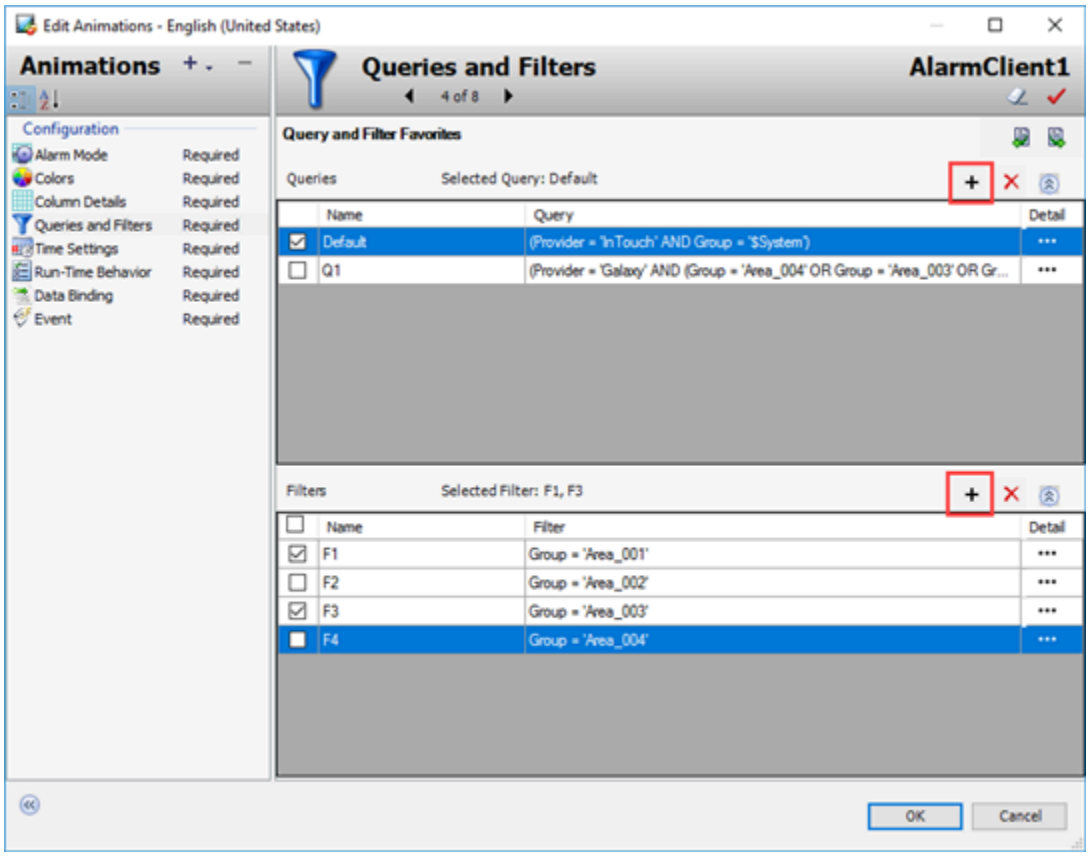
Add a query or filter

You can define a new query filter to filter the alarms shown in the Alarm Client Control. The new query filter is saved as a favorite in the **Query Filter Favorites** list.

To add a new query filter

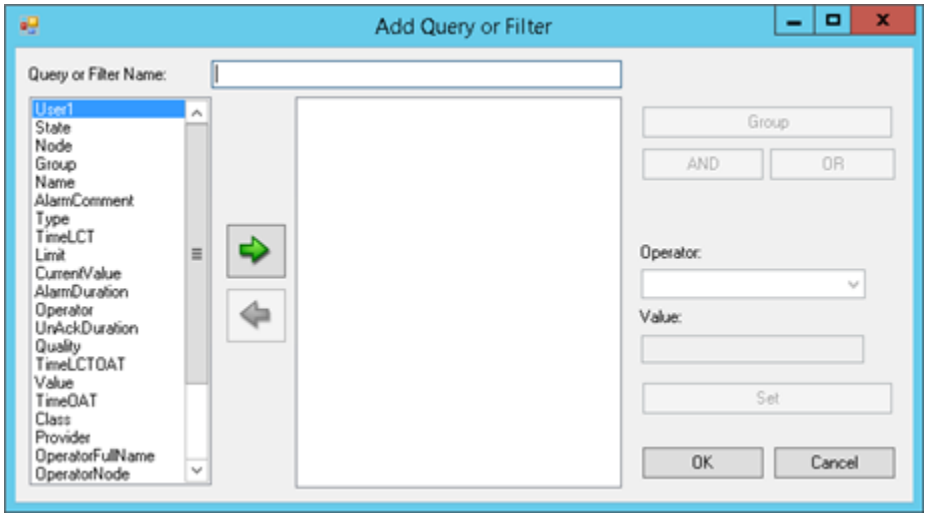
1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.

2. Click **Queries and Filters**. The **Query and Filter Favorites** page appears.



3. Click the **Add New Query** button above the saved query favorites box or click the **Add New Filter** button above the saved filter favorites box.

   The **Add Query or Filter** dialog box appears.



4. Construct queries or filters as needed.

   For more information, see Construct filters

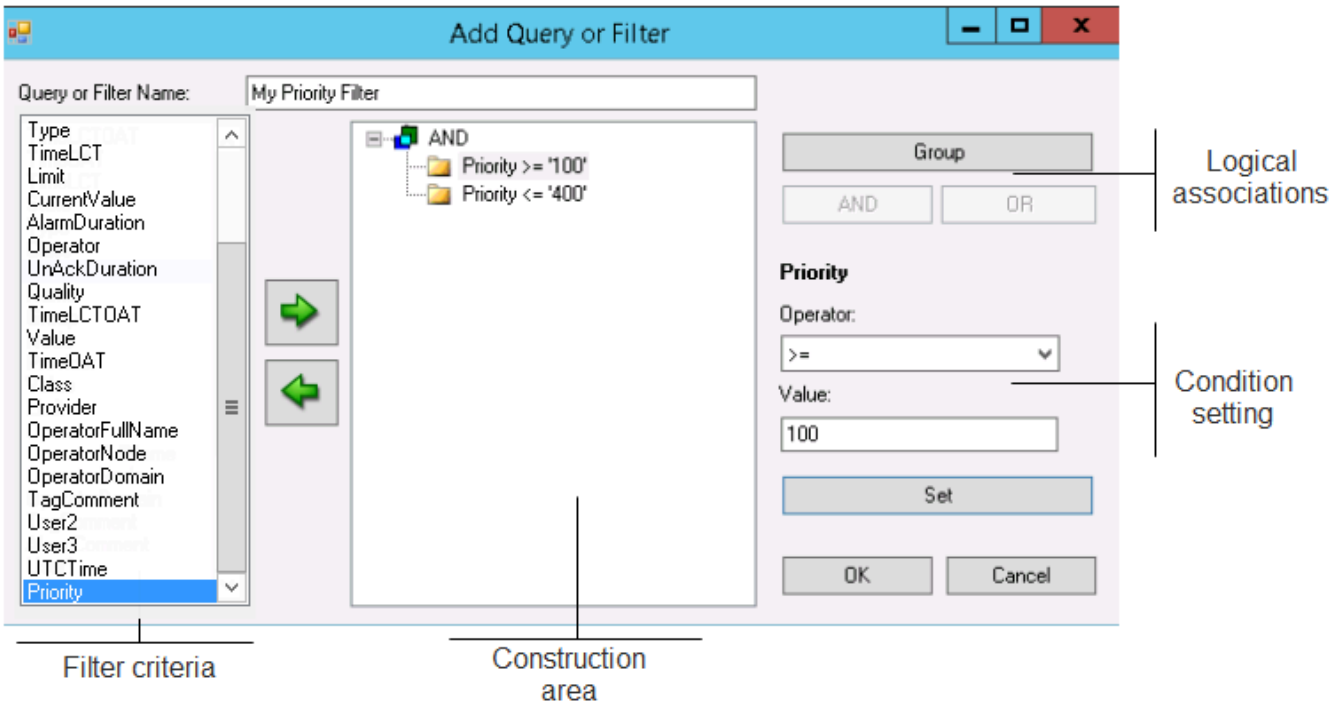
5. You can configure and select more than one filter for any selected query. For example, “F1” and “F3”.

   The “Selected Filter” label will display the selected filters using comma separator.

6. To select all filters, click the **Select All** checkbox. If the Select All checkbox is unchecked, all filters will be unselected.

Construct filters

You use the **Add Query or Filter** dialog box to create or edit a filter graphically.



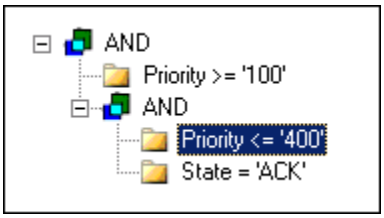
To construct a filter

1. If you want to change the filter name, type a new unique name in the **Filter Name** box.
2. Add filter criteria to the construction area by selecting a column name on the left and clicking the right arrow button. When you add filter criteria to the construction area, they are automatically logically connected by AND.

3. If necessary, remove filter criteria by selecting them in the filter construction area and clicking the left arrow button.
4. To change the logical operator, select it in the filter construction area, and then either:
   - Click **AND** or **OR**
   - Right-click and select **AND** or **OR** from the shortcut menu
5. To group filter criteria logically, either:
   - Drag a filter criteria in the construction area over another filter criteria
   - Select one filter criteria, click **Group**, and then click the other filter criteria

   By default, the filter criteria are logically grouped with AND. If necessary, you can select the **AND** item in

the tree and click **OR** to change it to an OR grouping.

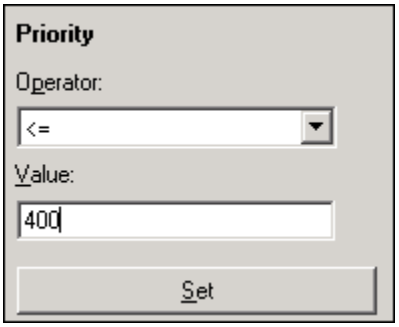


6. Assign values to filter criteria.

   Note: If you are using the Value column as a filter criteria, you may get unexpected results at run time. The items in the Value column are sorted alphabetically, not numerically. This is because the Value column can contain strings.

   Do the following:

   a. Select a filter criteria in the construction area.
   b. Select an operator from the **Operator** list.
   c. Type or select a value in the **Value** box.



   d. Click **Set**. The filter criteria is updated in the construction area.

7. To cut, copy, or paste individual filter criteria or filter criteria branches, right-click on the filter criteria and select the appropriate option from the shortcut menu.
8. When you are done, click **OK**.

Modify a query or filter

You can modify an existing query or filter using the **Modify Query or Filter** dialog box.

To modify a query or filter

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Queries and Filters**. The **Queries and Filters** page appears.
3. Select an existing query or filter in the **Query and Filter Favorites** list by clicking on the check box before each query or filter name.
4. Click the ellipsis button. The **Modify Query or Filter** dialog box appears. For more information, see Construct filters.
5. Click **OK**.

Delete a query or filter favorite

You can delete any non-default query or filter favorite.

To delete a query or filter favorite

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Queries and Filters**. The **Queries and Filters** page appears.
3. Select an existing query or filter in the **Queries and Filter Favorites** list by clicking on the check box before each query or filter name.
4. Click the **Delete selected query** button or click the **Delete selected filter** button.
5. When a message appears, click **Yes**.

Export query and filter favorites

You can export the query and filter favorites lists to an XML file. The XML file containing the query and filter favorites can be imported to other Alarm Control in design time or run time. Do not edit this file directly. The default query is also exported to the XML file.

To export the query and filter favorites lists

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Queries and Filters**. The **Queries and Filters** page appears.
3. Select an existing query or filter in the **Query and Filter Favorites** list by selecting the check box before each query or filter name.
4. Click the **Export** button. Select a location and a name for the XML file and click **Save**.

Import query and filter favorites

You can import the query filter favorites list from an XML file.

To import the query filter favorites list

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Queries and Filters**. The **Queries and Filters** page appears.
3. Click **Import**. Navigate to the XML file you want to import.
4. In the **Import Option** list, click either:
   - **Append** to append the imported query filters to the existing query filters. If query filter names in the imported XML conflict with existing query filters, you are prompted to confirm the import for each filter
   - **Overwrite** to replace all existing query filters with the imported query filters
5. Select the XML file and click **Open**.

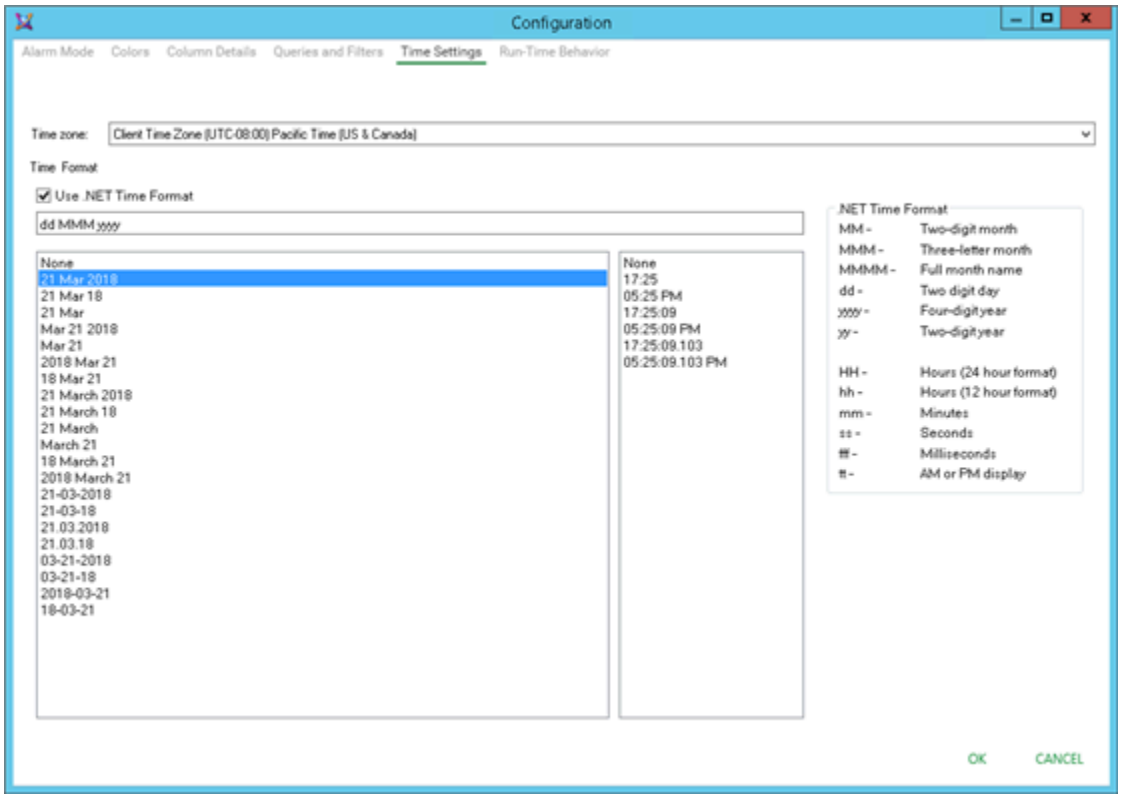
Set time zone and format

You can set the time zone in which the client shows the alarm and event records. By default, the time zone is set

to the client computer's current time zone at design time. Use the TimeZone.TimeZone Property, Time.Type Property, and Time.Format Property properties in scripting to set the time zone, time type, and time format at run time.

You can also set the time format of the alarm and event records. You can select between two different time format sets:

- Default Time Format: same as the InTouch Alarm Viewer control and InTouch Alarm DB View control of InTouch version 10.0 and later
- .NET Time Format: defined by Microsoft .NET Framework time format conventions



Set the time zone

You can set the time zone in which the Alarm Control shows the alarm and event records.

You can either set the time display to a predefined time zone, or to the client time zone. The client time zone is the time zone of the computer on which the Alarm Control is running.

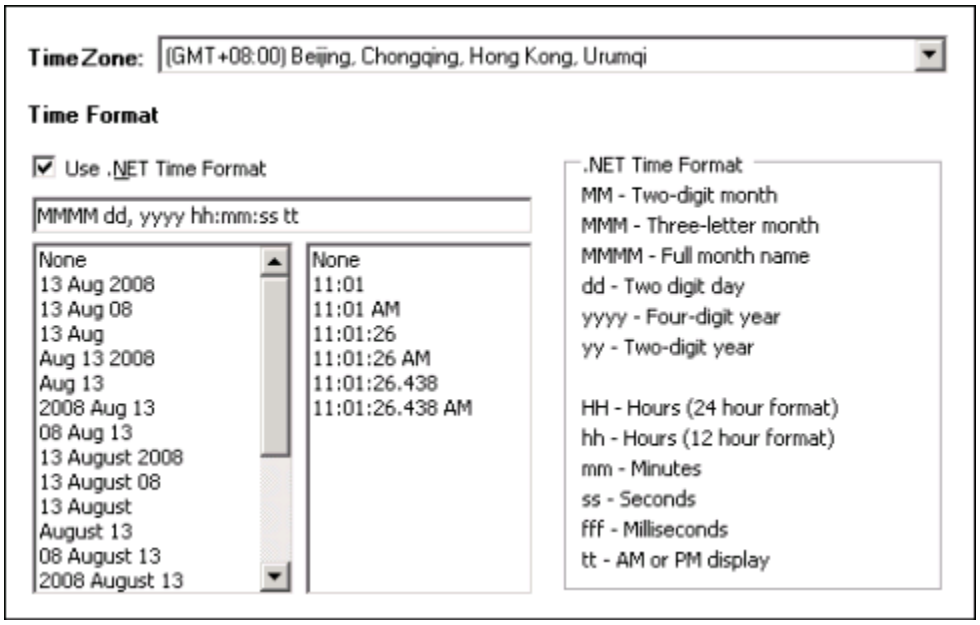
The **Client Time Zone** setting is useful if you are deploying an application using the Alarm Control to a different time zone.

For example, if you develop your application in the Pacific Time zone and deploy it to two computers in the time zones Central Time and Eastern Time, you can ensure the Alarm Control shows the local time for each deployment by setting the time zone to **Client Time Zone**.

To set the time zone

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Time Settings**. The **Time Settings** page appears.

3. In the **Time Zone** list, select a time zone.



4. Click **OK**.

Set the time format

You can set the time format in which the Alarm Control shows the alarm and event records. You can either use a predefined datetime format, or compose one.

To set the time format

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Select **Time Settings**.
3. In the **Time Format** area, do the following:
   a. Make sure **Use .NET Time Format** is cleared.
   b. Select date format codes from the list at the right. The equivalent date format code appears in the box above.
   c. Select time format codes from the list at the right. The equivalent time format code is appended to the date format string.
4. If you want to customize the datetime format, modify the codes in the box as follows:

Code	Purpose	Example
%m	Two-digit month	03
%b	Three-letter month	Mar
%B	Full month name	March
%d	Two-digit day	17

Code	Purpose	Example
%Y	Four-digit year	2008
%y	Two-digit year	08
%#x	Full day and date	Tuesday, March 11, 2008
%H	Hours in 24 hour format	14
%I	Hours in 12 hour format	2
%M	Minutes	55
%S	Seconds	34
%s	Milliseconds	223
%p	AM or PM	PM

5. Click **OK**.

Set a .NET DateTime format

You can set the .NET DateTime format in which the Alarm Control shows the alarm and event records. You can either use a predefined DateTime format, or compose one. The predefined date format is based on the short date format setting of the operating system and may vary from computer to computer.

To set the .NET DateTime format

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Time Settings**. The **Time Settings** page appears.
3. In the **Time Format** area, do the following:
   a. Select the **Use .NET Time Format** check box.
   b. Select date format codes from the list at the right. The equivalent date format code appears in the box above.
   c. Select time format codes from the list at the right. The equivalent time format code is appended to the time format string.
4. If you want to customize the DateTime format, modify the codes in the box as in the table below. For more information, see the Microsoft Knowledge database on .NET DateTime formats.

Code	Purpose	Example
M	Single-digit month	9
MM	Two-digit month	09
MMM	Three-letter month	Sep

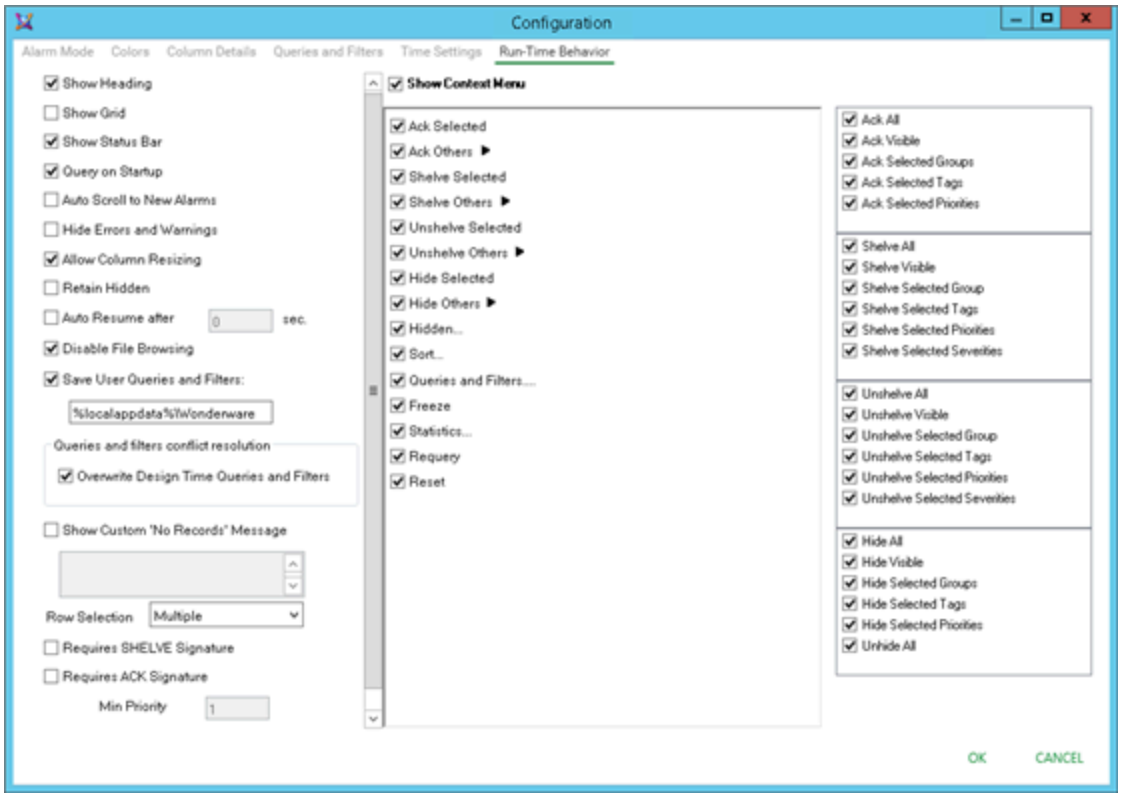
Code	Purpose	Example
MMMM	Full month name	September
d	Single-digit day	8
dd	Two-digit day	08
ddd	Abbreviated day of the week	Mon.
dddd	Day of the week	Monday
yyyy	Four-digit year	2008
yy	Two-digit year	08
HH	Hours in 24 hour format	14
hh	Hours in 12 hour format	2
mm	Minutes	55
ss	Seconds	34
fff	Milliseconds	223
tt	AM or PM	PM

5. Click **OK**.

Configure runtime behavior

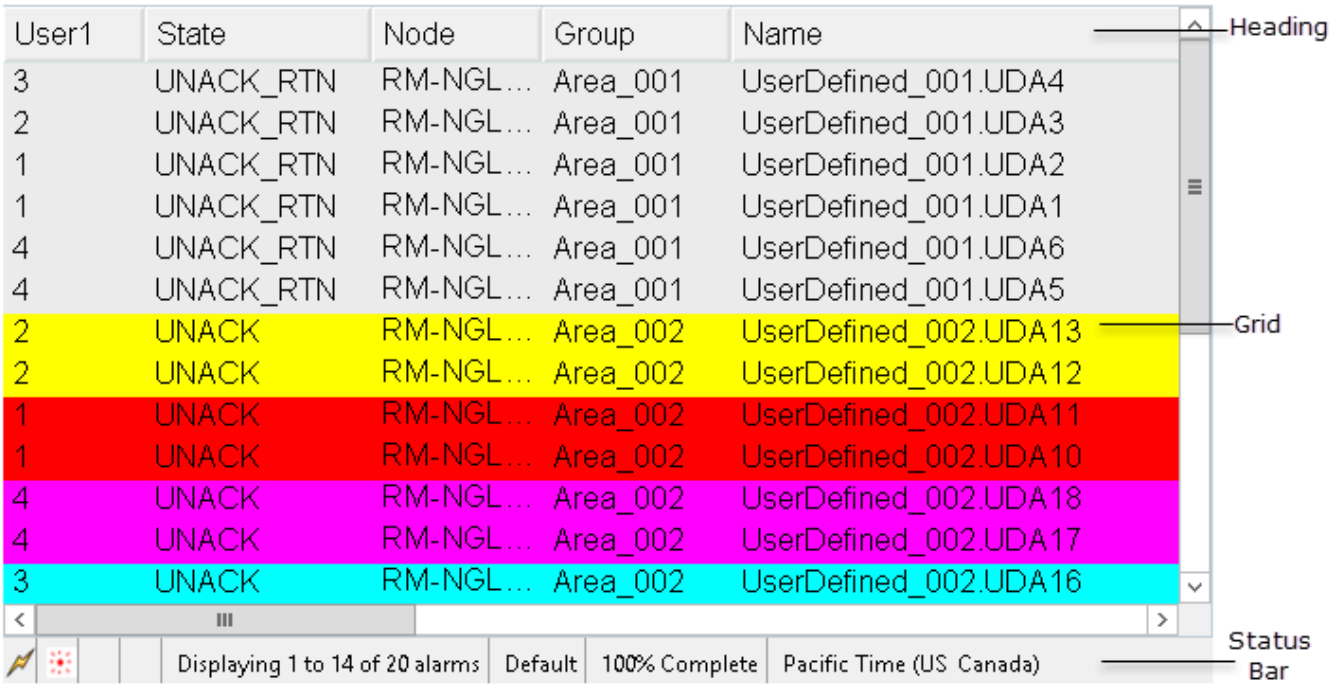
You can configure the behavior and appearance of the Alarm Control at run time, for example:

- Showing and Hiding parts of the Alarm Control
- Specifying if the Alarm Control queries the alarm database when it starts up
- Scrolling to new alarms
- Hiding warnings, errors, and messages
- Restricting operator access to parts of the Alarm Control
- Specifying Alarm Control freeze behavior
- Customizing the "no records" message
- Configuring alarms to require an ACK signature
- Configuring alarms to require a SHELVE signature
- Customizing the run-time shortcut menu



Show heading, grid, or status bar

You can show and hide parts of the Alarm Control at run time, such as the heading, grid, or status bar. Use the ShowHeading Property, ShowGrid Property, and ShowStatusBar Property properties in scripts to show or hide the heading, grid, and status bar at run time.



To show the heading, grid, or status bar at run time

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Run-Time Behavior**. The **Run-Time Behavior** page appears.
3. To show or hide parts of the Alarm Control during run time, do any of the following:
   - Select the **Show Heading** check box to show the heading at run time, or clear it to hide the heading at run time.
   - Select the **Show Grid** check box to show the grid at run time, or clear it to hide the grid at run time.
   - Select the **Show Status Bar** check box to show the status bar at run time, or clear it to hide the status bar at run time.

     Caution: If you hide the status bar, you will not be able to see important indicators, such as the New Alarms, Hidden Alarms, and Frozen Grid indicators.
4. Click **OK**.

Automatic query for alarms on startup

You can configure the Alarm Control to automatically query the Alarm Database when the control starts up at run time. Use the QueryStartup Property in scripts to control query behavior when the Alarm Control starts.

By default, current alarms and recent alarms and events are automatically queried when the Alarm Control starts at run time. You can disable the automatic query if the Alarm Control is:

- Configured to mainly use query filters
- Controlled mainly by scripts

To query the Alarm Database automatically on startup

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Run-Time Behavior**. The **Run-Time Behavior** page appears.
3. Select the **Query on Startup** check box.
4. Click **OK**.

Scroll automatically to new alarms

If an operator views multiple pages of alarms, new alarms may go unnoticed. You can configure the Alarm Control to scroll automatically to show new alarms. Use the AutoScroll Boolean property in scripts to scroll automatically to new alarms.

However, if the Alarm Control scrolls automatically to new alarms, it may be hard for the operator to view and analyze older alarms if new alarms occur. If the Alarm Control is frozen, it will not scroll automatically to new alarms.

To scroll automatically to new alarms

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Run-Time Behavior**. The **Run-Time Behavior** page appears.
3. Select the **Auto Scroll to New Alarms** check box.
4. Click **OK**.

Hide errors, warnings, and status messages

You can prevent a message dialog box from opening when errors, warnings, or status messages occur in the Alarm Control. Even if you hide errors, warnings and status messages are sent to the Logger. Use the HideErrors property in scripts to hide error, warning, and status messages at run time.

To hide error and warning messages

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Run-Time Behavior**. The **Run-Time Behavior** page appears.
3. Select the **Hide Errors and Warnings** check box.
4. Click **OK**.

Restrict user access to rows and columns

You can prevent the operator from:

- Resizing columns
- Selecting rows
- Selecting multiple rows

Use this feature for interfaces where it is easy to accidentally resize columns or select rows. For example, if the

Alarm Control is running on a small display, use the AllowColumnResize Property and RowSelection Property properties in scripting to control the ability to resize columns and select rows at run time.

To prevent the operator from resizing columns

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Run-Time Behavior**. The **Run-Time Behavior** page appears.
3. Clear the **Allow Column Resizing** check box.
4. Click **OK**.

To prevent the operator from selecting rows

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Run-Time Behavior**. The **Run-Time Behavior** page appears.
3. In the **Row Selection** list, click:
   - **No** to prevent operator from selecting rows
   - **Single** to allow operator to only select one row
   - **Multiple** to allow operator select multiple rows
4. Click **OK**.

Retain hiding when changing the alarm query filter

You can configure the Alarm Control to hide alarms even if the alarm query filter changes. Use the RetainHidden property in scripts to retain the hiding of alarms at run time.

To retain hiding when change the alarm query filter

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Run-Time Behavior**. The **Run-Time Behavior** page appears.
3. Select the **Retain Hidden** check box.
4. Click **OK**.

Override the frozen grid

You can configure the Alarm Control to unfreeze the grid after a given time in seconds. Use this option to make sure that new alarms appear on the grid after a specified time. Use the AutoResumeDuration Property in scripts to unfreeze the Alarm Control after a certain duration at run time.

The Alarm Control also unfreezes if you change one of the following:

- Alarm Mode
- Alarm Query
- Query Filter

To override the frozen grid

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Run-Time Behavior**. The **Run-Time Behavior** page appears.
3. Select the **Auto Resume after** check box and type the number of seconds after which the grid unfreezes.
4. Click **OK**.

Customize the "no records" message

You can customize the message that appears when there are no records to show in the grid. Use the NoRecordsMessage.Enabled Property and NoRecordsMessage.Message Property in scriptw to customize the "no records" message at run time.

To customize the "no records" message

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Run-Time Behavior**. The **Run-Time Behavior** page appears.
3. Select the **Show Custom 'No Records' Message** check box and type a message you want to show in the Alarm Control when there are no alarm records.
4. Click **OK**.

Change the language of the "no records" message

You can change the language of the message that appears when there are no records to show in the Alarm Control grid.

To change the language of the "no records" message

1. Right-click on the canvas and click **Scripts**. The **Edit Scripts** dialog box appears.
2. Click the Add icon and give the script a name, for example ChangeLanguage.
3. In the **Expression** box, type:

```
intouch:$Language
```

4. In the **Trigger** list, click **DataChange**.
5. In the script area, type the following:

```
If intouch:$Language == 1033 then ' Switch to English
AlarmClient1.NoRecordsMessage.Message = "No Records";
   else If intouch:$Language == 1031 then ' Switch to German
   AlarmClient1.NoRecordsMessage.Message = "Keine Einträge";
      else if intouch:$Language == 1036 then ' Switch to French
      AlarmClient1.NoRecordsMessage.Message = "Aucun enregistrement";
      endif;
   endif;
endif;
```

6. Click **OK**.

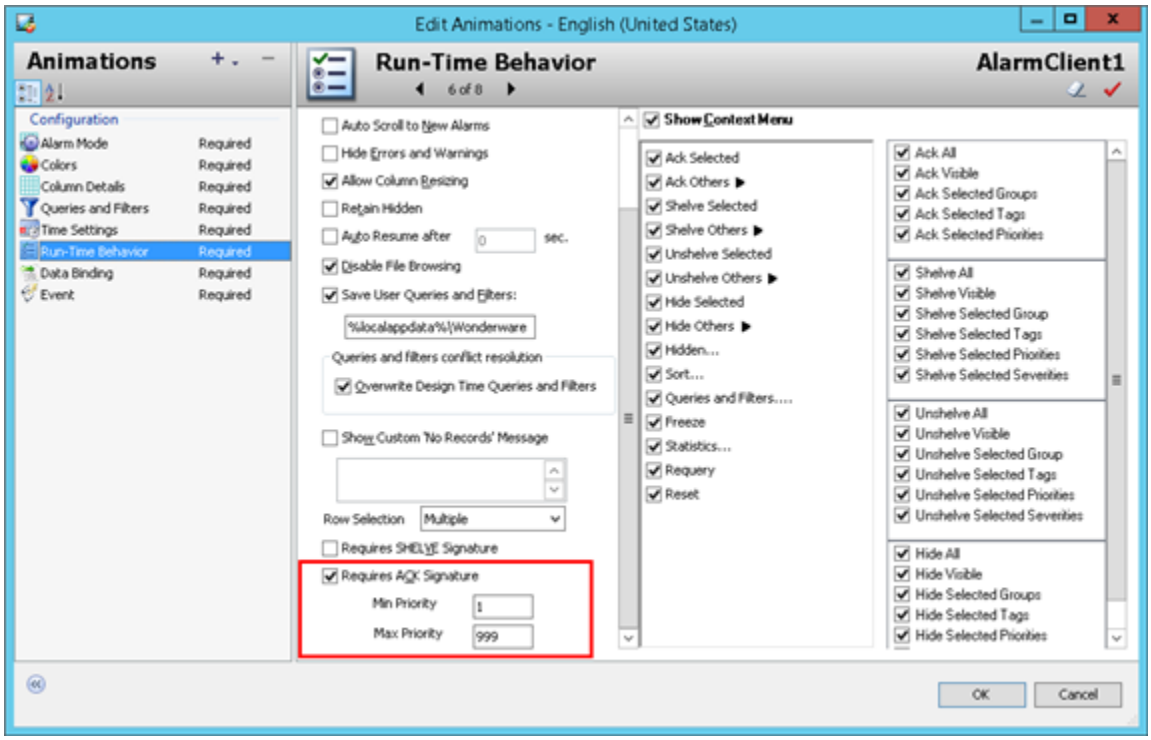
Configure the Alarm Control to require an ACK signature

Even if you are logged into the InTouch application, acknowledging the alarms that fall within a specific priority range may need to be authenticated by you if required by your company or industry. You can configure the Alarm Control to provide such functionality by selecting the **Requires ACK Signature** check box in the **Object Properties** dialog of the Alarm Control.

After the Alarm Control has been configured to require a signature for acknowledgement, you need to set its minimum and maximum priority values. The valid range of the minimum and maximum priority values is 1 to 999.

To configure the Alarm Control to require ACK signature

1. Place an Alarm Control in the drawing canvas.
2. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.



3. Click **Run-Time Behavior**. The **Run-Time Behavior** page appears.
4. Select the **Requires ACK Signature** check box. The **Min Priority** and **Max Priority** boxes are enabled.
5. Enter the minimum and maximum priority values for the alarm range that will require authentication.
6. Click **OK**.

You can configure the alarm signature requirement as well as the minimum and maximum values in run time using the **Object Editor**. For more information about using the **Object Editor**, see the Application Server User's Guide, Working with Objects.

For more information about configuring the Alarm Control in run time to require an alarm acknowledgement signature, see the "SignedAlarmAck() Applied Examples" topic in the Creating and Managing Industrial Graphics User's Guide.

Note: If your computer supports Smart Cards, you can use them for alarm authentication at run time. You need

to have the Smart Card mapped to your user account in the domain. If the Smart Card is already in the reader and you enter the correct PIN, the system will allow you to acknowledge the alarms.

Configure the Alarm Control to require a SHELVE signature

Operators can temporarily shelve selected alarms from the list of an Alarm Control's active alarms. A shelved alarm is suppressed and removed from the list of active alarms. Typically, operators shelve lower severity nuisance alarms because they provide little diagnostic value and interfere with the operator's ability to manage a plant process.

An alarm is shelved for a specified period. After the period ends, alarms are automatically unshelved and appear again in the list of active alarms. Operators can also manually unshelve an alarm before the end of the specified shelved period.

By default, Medium and Low severity alarms are enabled for shelving. Critical and High severity alarms are not because of the potential risk of shelving and ignoring alarms that represent serious operating states. For more information about enabling shelving based on alarm severity, see "Configuring Alarm Severity to Priority Mapping" in the *Application Server User Guide*.

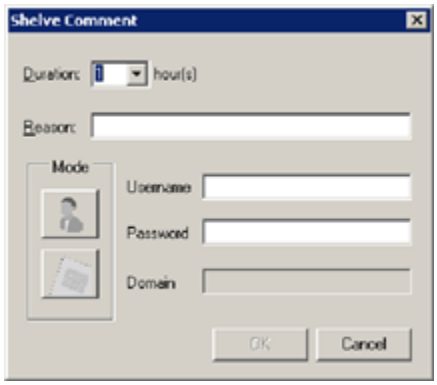
The Alarm Control provides a configuration option to restrict alarm shelving only to operators authorized by application security. When an operator selects one or more alarms and attempts to shelve them, the Alarm Control verifies if a shelve signature is required.

- If no SHELVE signature is required

  The Alarm Control shows a simple **Shelve Comment** dialog box with a **Duration** field to select the number of hours (1, 2, 4, 8, 12) to shelve the alarm and a **Reason** field to enter a mandatory comment.

- If a SHELVE signature is required and application security is active

  The Alarm Control shows a **Shelve Comment** dialog box with fields for the operator to enter user credentials (name, password, domain) in addition to selecting a shelve duration and entering a mandatory comment.



  By default, the logged-in user appears in the **Username** field. If the application security type is **ArchestrA,** then ArchestrA appears in the **Domain** field and cannot be edited. If the credentials are valid, the Alarm Control attempts to shelve the selected alarms.

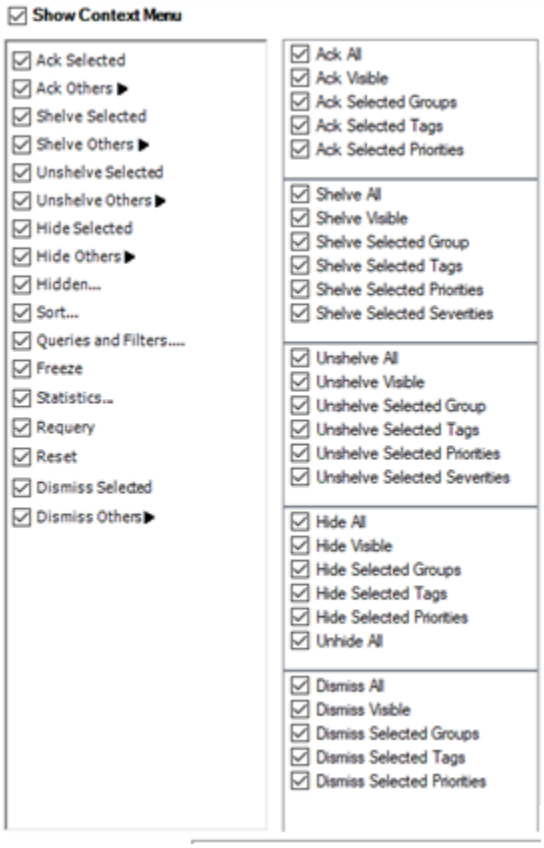
The Alarm Control shows an error message if the operator enters invalid credentials. When the operator clicks **OK** on the error message, the **Shelve Comment** dialog appears again with the entered user name, comment, and duration. The Password (or PIN) is blank. The operator can attempt to authenticate again or cancel.

To configure the Alarm Control to require a SHELVE signature

1. Place an Alarm Control in the drawing canvas.
2. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
3. Click **Run-Time Behavior**. The **Run-Time Behavior** page appears.
4. Select or clear **Requires SHELVE Signature** based on whether operators need to enter their credentials to shelve alarms or not.
5. Click **OK**.

Configure the runtime shortcut menu

You can configure the Alarm Control's shortcut menu to show only selected options at run time. The shortcut menus showing historical alarms (or events) and current alarms (or recent alarms and events) are different.



For the current alarms Context menu, you can also show or hide entire shortcut submenus. Use the ContextMenu and the ShowContextMenu Property in scripting to control if shortcut menu items appear or not at run time.

To hide the Context menu

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Run-Time Behavior**. The **Run-Time Behavior** page appears.
3. Clear the **Show Context Menu** check box.
4. Click **OK**.

To show or hide Context menu options

1. Double-click the Alarm Control on the canvas. The **Edit Animations** dialog box appears.
2. Click **Run-Time Behavior**. The **Run-Time Behavior** page appears.
3. Make sure the **Show Context Menu** check box is selected.In the shortcut menu lists, do the following:
   a. Select the options you want to appear on the run-time Context menu (if applicable for the selected client mode).
   b. Clear the options you want to hide from the operator on the run-time Context menu.
4. Click **OK**.

Use the Alarm Control at runtime

During runtime, users can show a shortcut menu containing options to monitor and manage alarms. Also, other options enable users to sort and filter alarms shown by the Alarm Control.

Refresh the Alarm Control grid

You can refresh the Alarm Control grid at run time. The Alarm Control retrieves alarm record data based on the Alarm Database time range settings.

To refresh the Alarm Control

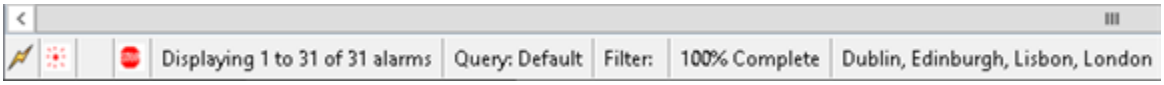
1. Right-click the Alarm Control grid at run time. The shortcut menu appears.
2. Click **Requery**.

View status bar information

The status bar shows you information about the current Alarm Control grid. Depending on the client mode, the status bar information shows different information.

Status bar information for current or recent alarms

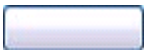
If the Alarm Control is showing current alarms or recent alarms and events, the status bar shows the following:



Element	Icon(s)	Description
Client Mode		Indicates the Alarm Control is showing current alarms (or recent alarms and events).
New Alarms		Appears if new alarms have occurred. If you move the pointer over the indicator, the tooltip shows how many alarms are unacknowledged.
Hidden Alarms		Appears if any alarms are currently hidden. If you move the pointer over the indicator, the tooltip shows how many alarms are hidden.
Frozen Grid		Appears if the Alarm Control is currently frozen.
Alarm Records		Displaying 1 to 13 of 28 alarms Shows the total number of alarm records and which alarms are currently shown.
Query Filter		Default Shows the name of the current query filter favorite.
Retrieval		100% Complete Shows the percentage of alarms retrieved from all alarm providers. If this percentage is less than 100%, not all alarm providers are providing alarm data. Use the **Alarm Statistics** dialog box to detect which alarm providers are not providing alarm data.
Time Zone		Beijing, Chongqing, Hong Kong, Urumqi Shows the current time zone of the Alarm Control. Move the pointer over the time zone to show the full information in a tool tip.

Status bar information for historical alarms

If the Alarm Control shows historical alarms or events, the status bar shows the following fields:

Element	Description
Client Mode	 Indicates the Alarm Control is showing historical alarms and/or events.
Alarm Records	Displaying 1 to 13 of 28 alarms Shows the total number of alarm records and the number of alarms shown by the Alarm Control.
Alarm Database	localhost - WWAlmDb Shows the name of the server hosting the Alarm Database and the Alarm Database name.
Connection Status	Connected Shows the connection status to the Alarm Database.
Time Zone	Beijing, Chongqing, Hong Kong, Urumqi Shows the current time zone of the Alarm Control. Move the pointer over the time zone to show complete time zone information in a tool tip.
Requery	 Click this button to retrieve latest alarm records from the Alarm Database.

Acknowledge alarms

You can configure the Alarm Control to require an alarm to be acknowledged even if the condition causing the alarm has passed. This ensures that an operator is aware of events that caused a temporary alarm state, but have returned to normal. You acknowledge alarms at run time using a shortcut menu or by script methods.

You can acknowledge alarm records directly from the Alarm Control. You can acknowledge:

- One or more selected alarms
- All alarms, including alarms not visible due to the limited space of the Alarm Control
- All visible alarms
- All alarms with common values, such as provider names, group names, priority ranges, and tag names. You can simplify alarm acknowledgement for the operator by using methods in scripting. For more information, see the Ack.All() method.

  Note: When the Alarm Client Control is in the Recent Alarms and Events mode, the displayed alarms cannot

be acknowledged.

To acknowledge selected alarms using the Alarm Control grid

1. Select one or more alarms in alarm state.
2. Right-click on an alarm shown in the Alarm Control and click **Ack Selected**.

   If no default acknowledgement statement is configured for the Alarm Control, the **Ack Comment** dialog box appears.
3. Type an alarm acknowledgement comment and click **OK**.

To acknowledge other alarms using the Alarm Control grid

1. Select one or more alarms in alarm state.
2. Right-click the Alarm Control grid, point to **Ack Others**, and click one of the following:
   - **Ack All** to acknowledge all alarms in alarm state
   - **Ack Visible** to acknowledge all visible alarms
   - **Ack Selected Group** to acknowledge alarms with the same provider names and group names of one or more selected alarms in alarm state.
   - **Ack Selected Tag** to acknowledge alarms with the same provider names, group names, and tag names within the priority ranges of one or more selected alarms in alarm state.
   - **Ack Selected Priority** to acknowledge alarms with the same provider names, group names, and within the priority ranges of one or more selected alarms in alarm state.
3. If no default acknowledgement statement is configured for the Alarm Control, the **Ack Comment** dialog box appears.
4. Type an alarm acknowledgement comment and click **OK**.

Provide a signature to acknowledge alarms

You must provide your signature in the **Ack Alarms** dialog box to acknowledge alarms. A signature is required when the **Require ACK Signature** check box is selected and any of the selected alarms falls within the configured priority range, or if no one is logged on to the InTouch application. You are not required to be logged on to the InTouch application to acknowledge alarms. However, if you are logged on, the **Username** box displays the username.

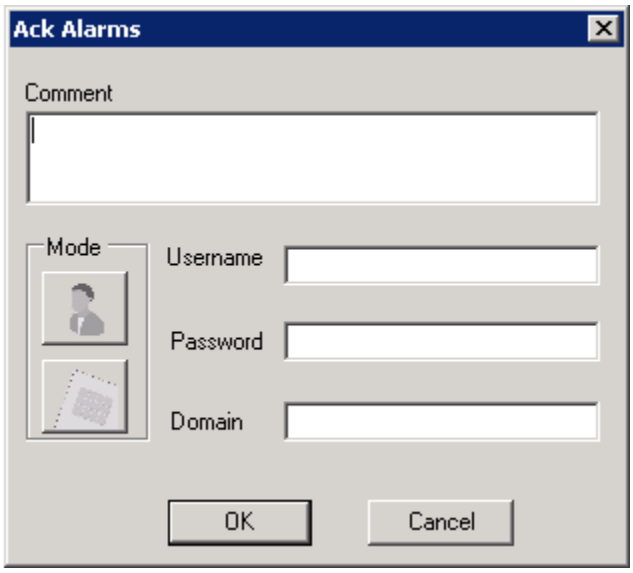
When you select the **Require ACK Signature** check box, the acknowledgment comment is prefixed in the updated Alarm Record. If one or more of the selected alarms falls within the configured priority range, the comment is prefixed with "Signed ACK -" indicating that it is a signed acknowledgment. Otherwise, it is prefixed with "Std ACK -" indicating that it is a standard acknowledgment.

Note: If the selected alarms do not require a signature, then the **Ack Alarms** dialog box displays the Comment box. You can enter a comment before acknowledging the alarm.

To provide a signature to acknowledge alarms with user name

1. Select one or more alarms in the alarm state.
2. Right-click the **Alarm Control** grid and click **Ack Selected**.
3. If the selected alarms require a signature, or if you are not logged on to InTouch, then the **Ack Alarms** dialog

box appears. If the Smart Card authentication system is not configured on your computer, the following dialog appears:



4. In the **Comment** box, enter or modify the alarm comment.
5. In the **Username** box, enter your user name.
6. In the **Password** box, enter your password.
7. In the **Domain** box, enter the domain and click **OK**.

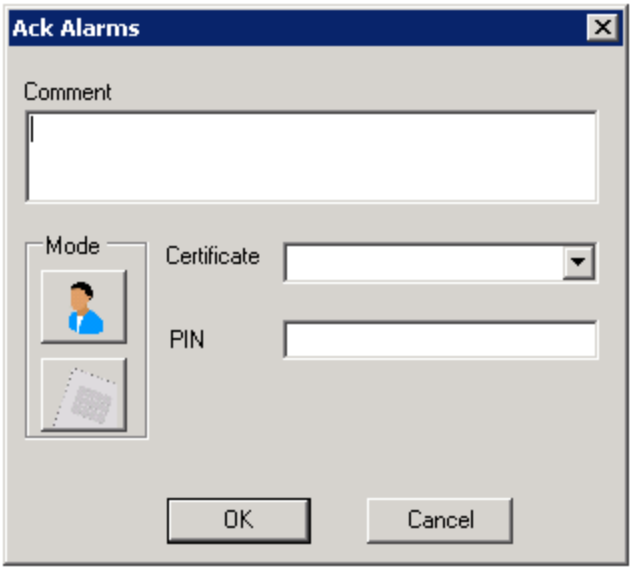
Note: If you enter invalid credentials, the Alarm Control displays an error message. When you click **OK**, the **ACK Alarms** dialog box appears again with the user name and the comment you had provided. You must enter the correct password.

Acknowledge alarms with Smart Cards

You can use a Smart Card to provide the authentication to acknowledge alarms if your computer supports Smart Card authentication. In that case, the Alarm Control displays the **Ack Alarms** dialog box with the Smart Card authentication dialog. You must have the Smart Card inserted in the Smart Card reader attached to your computer.

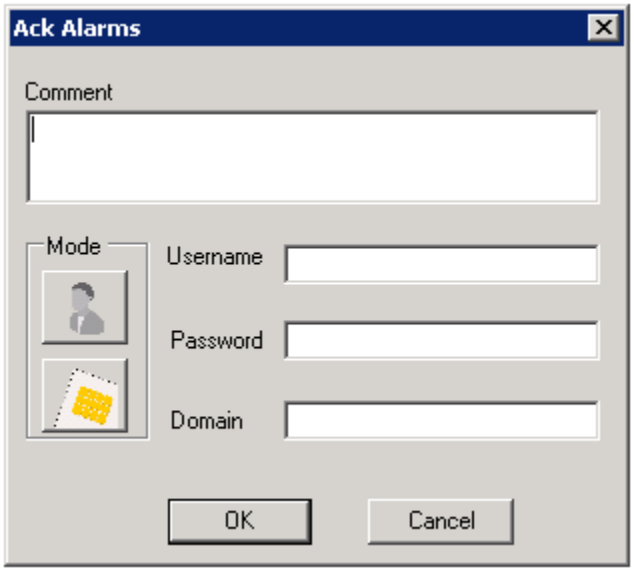
To provide a signature to acknowledge alarms using Smart Cards

1. Select one or more alarms in an alarm state.
2. Right-click the Alarm Control grid and click **Ack Selected**.
3. If the selected alarms are configured to require a signature and if Smart Card Authentication is selected in InTouch, the **Ack Alarms** dialog box appears where the **Smart Card** button under **Mode** is disabled.



4. In the **Comment** box, enter or modify the comment.
5. From the **Certificate** list, select the Smart Card currently inserted into the reader in your system.

   Note: If a card is newly-inserted or removed from the reader, you can update the list of cards by selecting the appropriate Smart Card from the **Certificate** list.
6. In the **PIN** box, enter the personal identification number and click **OK**.
   a. If you enter an invalid **PIN**, the system displays an error message. When you click **OK**, the **ACK Alarms** dialog box appears again with the user credential and the comment you had provided. You must enter the correct **PIN**.
   b. If you need to provide your log on credentials instead of the Smart Card details to verify your signature, click the **User Name Authentication** button under **Mode**. A different version of the **Ack Alarms** dialog box appears where the **Smart Card** button under **Mode** is enabled.



Dismiss alarms at runtime

When an alarm occurs, the run time operator (or system) can acknowledge the alarm to indicate that they are aware of the alarm. An acknowledged alarm that has returned to a normal value will continue to be displayed if the latching feature has been enabled for the Galaxy. With latching enabled, an acknowledged alarm that has returned to normal condition will be placed in the LATCHED alarm state.

LATCHED alarms can be displayed in the current alarms mode to show that the alarms did occur. Alarms go to the LATCHED state if:

- You ACK an alarm from UNACK_RTN state.

  Or
- You return an alarm from ACK state.

To view the LATCHED state, you have to enable the LATCHED state under the global settings. For more information, see the "Enable a Latched State" in the *AVEVA OMI* help.

You can dismiss the LATCHED alarms to remove the LATCHED alarms from the current mode of the Alarm Client Control grid. The dismissed LATCHED alarms would be visible in the recent mode of the Alarm Client Control.

To dismiss selected alarms

1. Select one or more alarms in **LATCHED** state.
2. Right-click anywhere on the grid and select **Dismiss Selected**.

The selected alarm is removed from the grid.

To dismiss other alarms

1. Select one or more alarms in **LATCHED** state.
2. Right-click anywhere on the grid, point to **Dismiss Others**, and select one of the following options:
   - **Dismiss All** to dismiss all alarms in alarm state
   - **Dismiss Visible** to dismiss all visible alarms
   - **Dismiss Selected Groups** Groups to dismiss alarms with the same provider names and group names of one or more selected active alarms
   - **Dismiss Selected Tags** to dismiss alarms with the same provider names, group names, and tag names of one or more selected active alarms

The relevant alarms are removed from the grid.

Shelve and unshelve alarms at runtime

Operators can shelve alarms to temporarily suppress them for a fixed period. Shelving an alarm means temporarily removing it from the Alarm Control's main alarm list and placing it on a shelved list. Shelving is normally controlled by an operator to handle irrelevant nuisance alarms that have not been caught by filtering or alarm suppression mechanisms.

When shelving an alarm from the Alarm Control, operators set an associated time period in which the alarm remains shelved and enter a mandatory comment. Operators can select from a list of Alarm Control Context

commands during run time to:

- Shelve one or more selected alarms
- Shelve all alarms
- Shelve only those alarms visible in the Alarm Control
- Shelve all alarms within the same alarm group as an alarm selected from the Alarm Control
- Shelve alarms by selected tags or attributes
- Shelve alarms by selected alarm priorities
- Shelve all alarms that have the same severity as an alarm selected from the Alarm Control

When application security is used, alarms can be shelved and unshelved only by operators with proper authorization. For more information about setting shelving authorization, see Configure the Alarm Control to require a SHELVE signature.

Shelved alarms are automatically unshelved at the end of the specified time period.Operators can also manually unshelve alarms and return them to an active state. Alarms are unshelved automatically at the end of the shelving time period. An unshelved alarm reappears in the Alarm Control active list and resumes its state at the time it was shelved. Operators can manually unshelve a shelved alarm before the end of the shelved period and enter an optional comment.

Shelve alarms

You can shelve alarms directly from the Alarm Control by selecting commands from the Alarm Control's Context menu.

To shelve selected alarms using the Alarm Control

1. Select one or more alarms in alarm state.
2. Right-click the Alarm Control grid and click **Shelve Selected** from the shortcut menu.

   The **Shelve Comment** dialog box appears to set a shelf duration and enter a comment.

   Note: If the application runs under security and the Alarm Control has been configured to require a SHELVE signature, operators must authenticate themselves. The **Shelve Comment** dialog box includes additional fields for an operator to enter a username, password, and domain.
3. Select an alarm shelving duration from the **Duration** field.
4. Type a mandatory alarm shelving comment in the **Reason** field and click **OK**.

To shelve other alarms using the Alarm Control

1. Select one or more alarms in alarm state.
2. Right-click the Alarm Control, point to **Shelve Others**, and click one of the following shortcut commands:

- **Shelve All** to shelve all alarms in alarm state
- **Shelve Visible** to shelve all visible alarms
- **Shelve Selected Groups** to shelve alarms with the same provider names and group names of one or more selected active alarms
- **Shelve Selected Tags** to shelve alarms with the same provider names, group names, and tag names of one or more selected active alarms
- **Shelve Selected Priorities** to shelve alarms with the same provider names, group names, and within the same priority ranges of one or more selected active alarms
- **Shelve Selected Severities** to shelve alarms with the same provider names, group names, and within the same severity ranges of one or more selected active alarms

The **Shelve Comment** dialog box appears with fields to set a shelf duration and enter a comment.

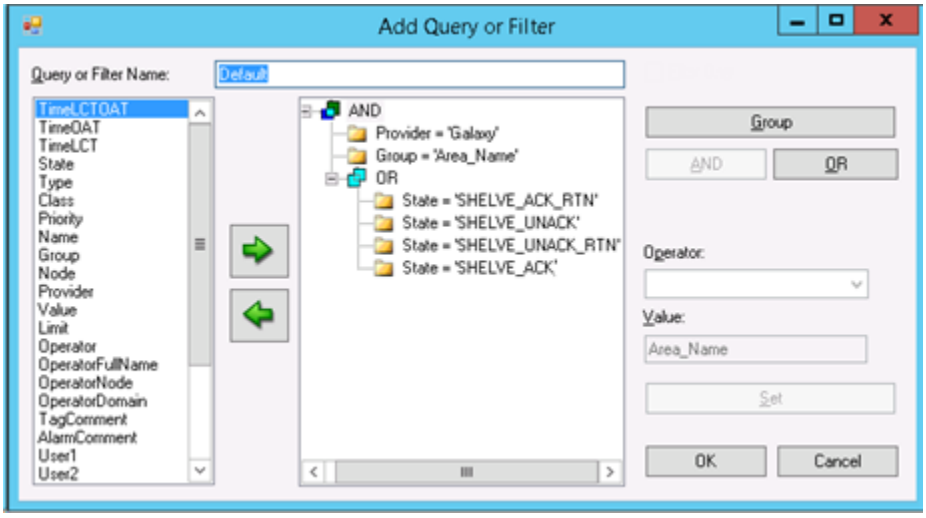
3. Select an alarm shelving duration from the **Duration** field.
4. Type an alarm shelving comment in the **Reason** field and click **OK**.

Show shelved alarms

After alarms are shelved, they no longer appear as active alarms in an Alarm Control. Filters only filter active alarms, so applying only a filter will not return shelved alarms. To show shelved alarms, you must use a query and apply a filter that is configured for shelved alarms.

For example:

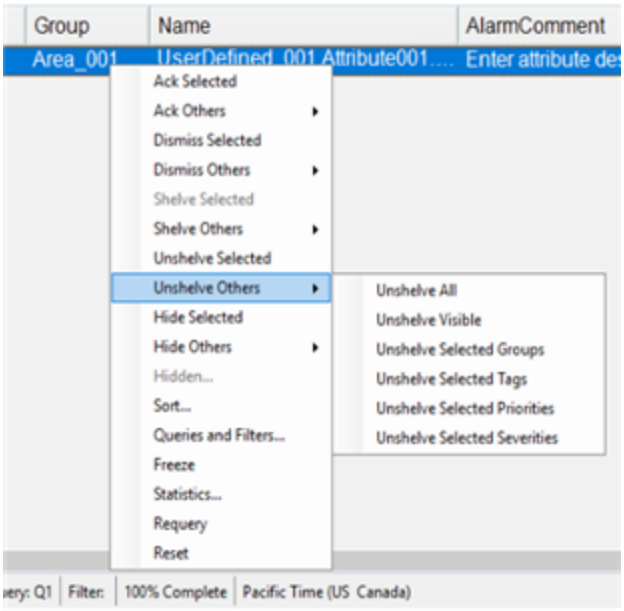
(Provider = 'Galaxy' AND Group = 'Area_Name' AND (State = 'SHELVE_ACK_RTN' OR State = 'SHELVE_UNACK' OR State = 'SHELVE_UNACK_RTN' OR State = 'SHELVE_ACK'))"



For more information about configuring queries to filter alarms, see Filter alarms.

Unshelve Alarms

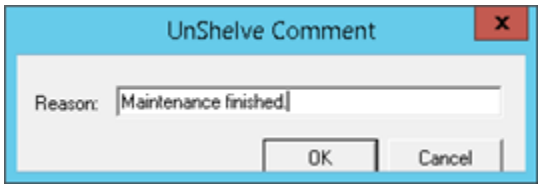
Manually unshelving alarms follows a similar sequence of steps to shelve alarms using Context menu commands. You should have created a query that shows the current shelved alarms.



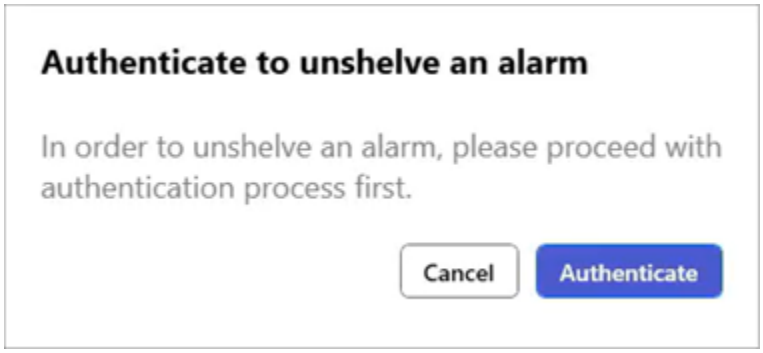
To unshelve selected alarms using the Alarm Control

1. Select one or more shelved alarms.
2. Right-click the Alarm Control grid and click **Unshelve Selected** from the Context menu.

   If **Operations Control connected experience** is not enabled in the Configurator, the **Unshelve Comment** dialog box appears.



   If **Operations Control connected experience** is enabled in the Configurator, then the **Authenticate to unshelve an alarm** dialog box is displayed.

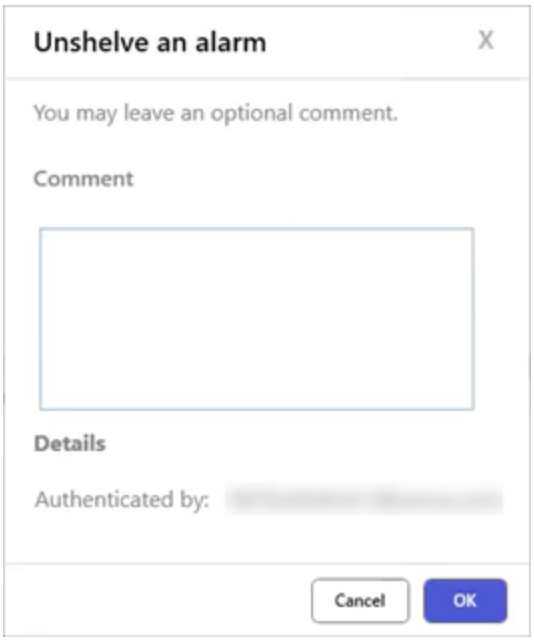


   a. Select **Authenticate**.

      AVEVA Identity Manager login page appears.

   b. Authenticate using the registered CONNECT account credentials. For more information refer to the "Authenticate using AVEVA Identity Manager" topic of InTouch HMI help.

      The **Unshelve an alarm** dialog box appears.



3. Type an optional alarm unshelving comment in the **Reason** field and click **OK**.

To unshelve other alarms using the Alarm Control

1. Select one or more alarms in alarm state.
2. Right-click the Alarm Control, point to **Unshelve Others**, and click one of the following shortcut commands:
   - **Unshelve All** to unshelve all shelved alarms
   - **Unshelve Visible** to unshelve all visible shelved alarms
   - **Unshelve Selected Groups** to unshelve alarms with the same provider names and group names of one or more selected shelved alarms
   - **Unshelve Selected Tags** to unshelve alarms with the same provider names, group names, and tag names of one or more selected shelved alarms
   - **Unshelve Selected Priorities** to unshelve alarms with the same provider names, group names, and within the same priority ranges of one or more selected shelved alarms
   - **Unshelve Selected Severities** to unshelve alarms with the same provider names, group names, and within the same severity ranges of one or more selected shelved alarms

   If **Operations Control connected experience** is not enabled in the Configurator, the **Unshelve Comment** dialog box appears.

   If **Operations Control connected experience** is enabled in the Configurator, then the **Authenticate to unshelve an alarm** dialog box is displayed.

   a. Select **Authenticate**.

      AVEVA Identity Manager login page appears.

   b. Authenticate using the registered CONNECT account credentials. For more information refer to the "Authenticate using AVEVA Identity Manager" topic of InTouch HMI help.

      The **Unshelve an alarm** dialog box appears.
3. Type an optional alarm unshelving comment in the **Reason** field and click **OK**.

Sort alarms at runtime

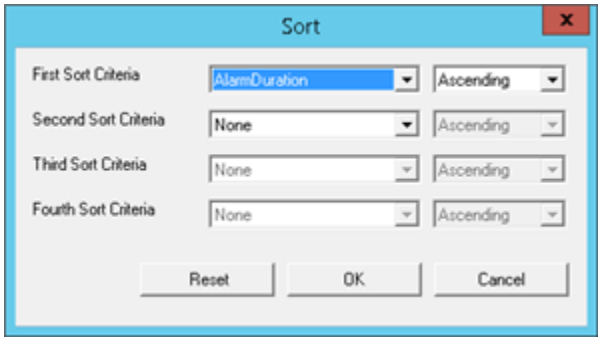
The Alarm Control supports alarm sorting for up to three columns at design time and runtime. At runtime, operators can configure sorting of even more columns by clicking on the column headers of the Alarm Control. You can sort alarms in ascending or descending direction for selected columns.

You can sort alarms at runtime in similar way as design time. Any changes you make to the sorting at runtime are lost when you switch back to design time.

Note: If you are sorting by the Value column, the items in the column are sorted alphabetically, not numerically. This is because the Value column can contain strings.

To set sorting columns and directions with lists at runtime

1. Right-click the **Alarm Control** grid and click **Sort**. The **Sort** dialog box appears.



2. In the **First Sort Criteria** list, select the first sort column and a sorting direction in the list to its right.
3. Optionally, select the second sort column in the **Second Sort Criteria** list and a sorting direction in the list to its right.
4. If you set the **Second Sort Column**, optionally select the third sort column in the **Third Sort Criteria** list and a sorting direction in the list to its right.
5. Click **OK**.

To set sorting columns and directions in the grid at runtime

1. In the Alarm Control grid, click on a column header to set sorting for the column. An arrow appears on the column header.
2. To change the sorting direction, click on the column header again. The arrow changes direction on the header.

   Note: If you click on a column header after releasing the **Shift** key, all sorting information is lost and the selected column is the new primary sorting criteria.
3. To set sorting for second and third columns, repeat step 3 while pressing the **Shift** key.
4. Release the **Shift** key.

Filter alarms at runtime

You can filter alarms at runtime by using the filters you defined at design time.

If you did not define a filter according to your needs at design time, you can still create new filters at runtime, or

modify existing filters.

If you saved filters to an XML file, you can load them from a file at runtime.

Filters you define at runtime are not saved for use at design-time. To re-use filters you create or modify at runtime, export the filter list to an XML file, and import the XML file at design time.

Naming conflicts may occur between the runtime and design time XML lists. To resolve naming conflicts between the XML files, leave the **Overwrite design time queries and filters** option checked in the **Run-Time Behaviors** tab of the **Edit Animations** dialog box. The runtime query or filter list will overwrite the design time list and resolve the naming conflict.

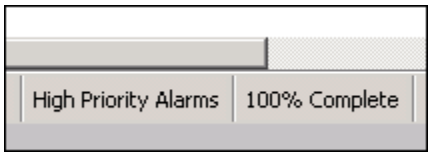
If you are showing historical alarms or events, you can use the filtering mechanism provided by the grid technology instead of using filter favorites.

Use an existing query filter

At run time, you can use any filter you defined at design time, regardless if you defined it for the current modes or historical modes. You can also use scripting to switch to an existing query filter. For more information, see Favorite Property.

To use an existing query filter

1. Right-click the Alarm Control grid and click **Queries and Filters**. The **Queries and Filters** dialog box appears.
2. Select the filter from the list and click **OK**. The alarm records are filtered and the current filter name appears in the status bar.



Add a query or filter at runtime

At runtime, you can create new queries or filters to limit the number of alarm records.

New queries or filters are saved and can be reused for future sessions.Upon close and restart of either the Alarm Control window or WindowViewer, added queries or filters will be available. For details, see Saving Run-Time Modifications to Queries and Filters.

To add a new query or filter at runtime

1. Right-click the Alarm Control grid and click **Queries and Filters**. The **Queries and Filters** dialog box appears.
2. The configuration is the same as in design time. For more information, see Add a New Query or Filter.

Modify a query or filter at runtime

At runtime, you can modify a query filter.

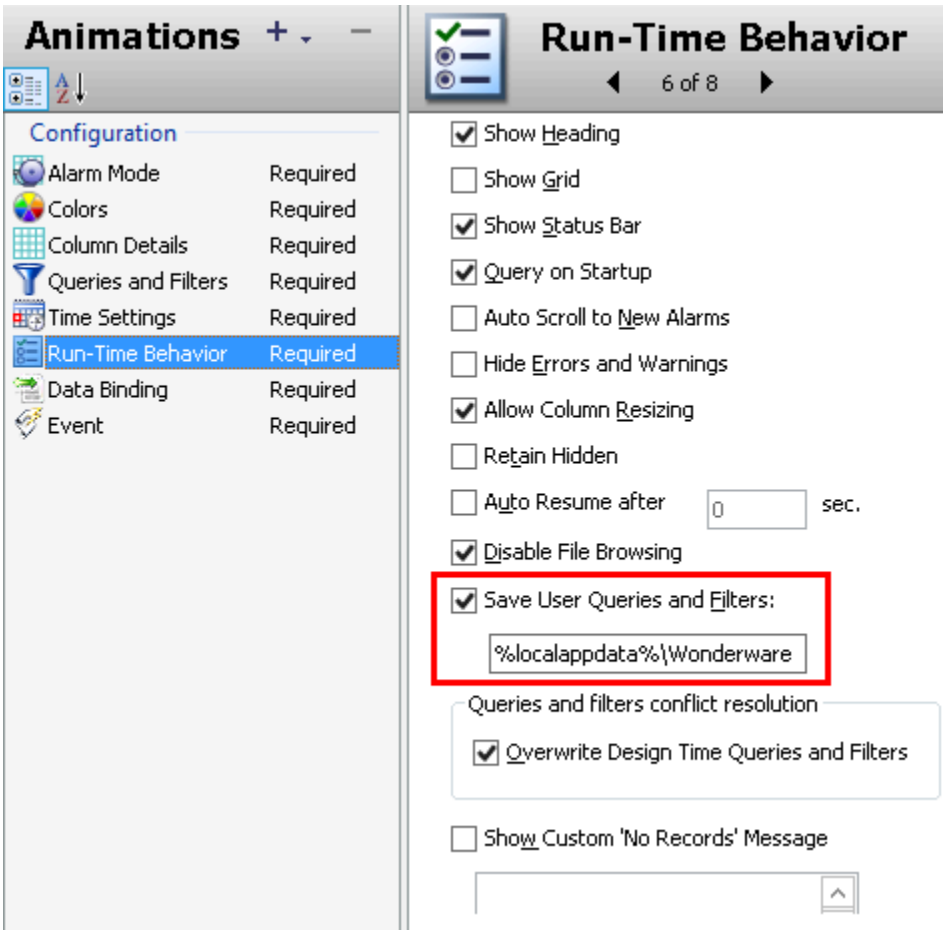
The modification of query filters is not saved for future use and is only valid for the current session. If you want to save the modifications, you must also export the query filters to an XML file. For more information, see Export query and filter favorites.

To modify an existing query filter

1. Right-click the Alarm Control grid and click **Query Filters**. The **Query Filters** dialog box appears.
2. The configuration is the same as in design time. For more information, see Modify a query or filter.

Save runtime modifications to queries and filters

You can save modifications made to queries and filters at runtime. There is an option to save user queries and filters in the **Run-Time Behavior** tab of the **Edit Animations** dialog box. It is selected by default, as shown below:



When this option is selected, modifications made to queries or filters at runtime are saved on a per user and alarm control instance to a XML file. For example, a saved query will save as UserQueryFilter_001.xml.

Modifications are saved in the following default directory:

```
C:\Users\<OS Login Name>\AppData\Local\Wonderware
```

The map between the Alarm Control instance and the above XML file is saved in a map file, "InstancesMap.dat".

Modified queries and filters will be available upon closing and restarting the alarm control window or WindowViewer.

If you uncheck the **Save User Queries and Filters** option, updates made to queries and filters during run-time will not be saved.

Note: When the **Save User Queries and Filter** option is checked, the **Overwrite Design Time Queries and Filters** option is enabled and checked by default. When checked, this option resolves naming conflicts between design-

time queries and filters and runtime queries and filters loaded from the XML file. Naming conflicts will be resolved by overwriting the design time queries and filters.

Delete a query or filter at runtime

At runtime, you can delete a query filter.

After you delete a query or filter at runtime, it is only deleted for the current session. If you want to save the list of query filters without the deleted query filter, you must export the query filters to an XML file. For more information, see Export query and filter favorites.

To delete an existing query or filter at runtime

1. Right-click the Alarm Control grid and click **Queries and Filters**. The **Queries and Filters** dialog box appears.
2. The configuration is the same as in design time. For more information, see Delete a query or filter favorite.

Import query and filter favorites at runtime

At runtime, you can import the list of query filters from an XML file.

To import query filter favorites

1. Right-click the Alarm Control grid and click **Query and Filters**. The **Query and Filters** dialog box appears.
2. The configuration is the same as in design time. For more information, see Import query and filter favorites.

Export query and filter favorites at runtime

At runtime, you can export the list of query filters to an XML file for future use. After exporting, you can import the query filter from the XML into design time.

Note: The default query filter favorite is not exported to the XML file.

To export query filter favorites

1. Right-click the Alarm Control grid and click **Query and Filters**. The **Query and Filters** dialog box appears.
2. The configuration is the same as in design time. For more information, see Export query and filter favorites.

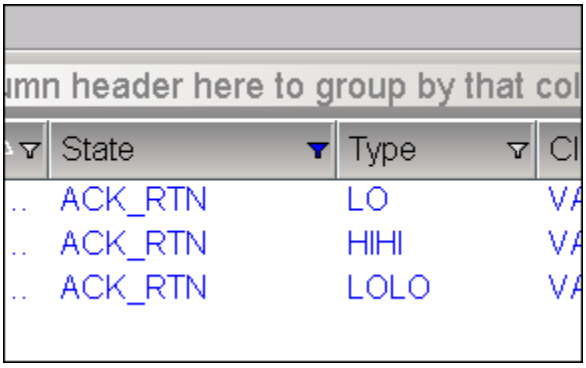
Filter alarms with client-based filtering

The grid technology used in the Alarm Control enables you to filter the grid contents after the data has been retrieved from the data source.

You can filter historical alarms and/or events in the following ways for any selected column:

Filter	Description
(All)	No filtering, all records are shown for the selected column.
(Custom)	Lets you configure a more complex filter for the selected column, for example a filter that can compare values of different columns.
(Blanks)	Filters by showing blank values only.
(NonBlanks)	Filters by showing non blank values only.
Values	Filters by the selected value.

If a filter is applied to any column in the Alarm Control, the filter icon in the column header appears in blue.

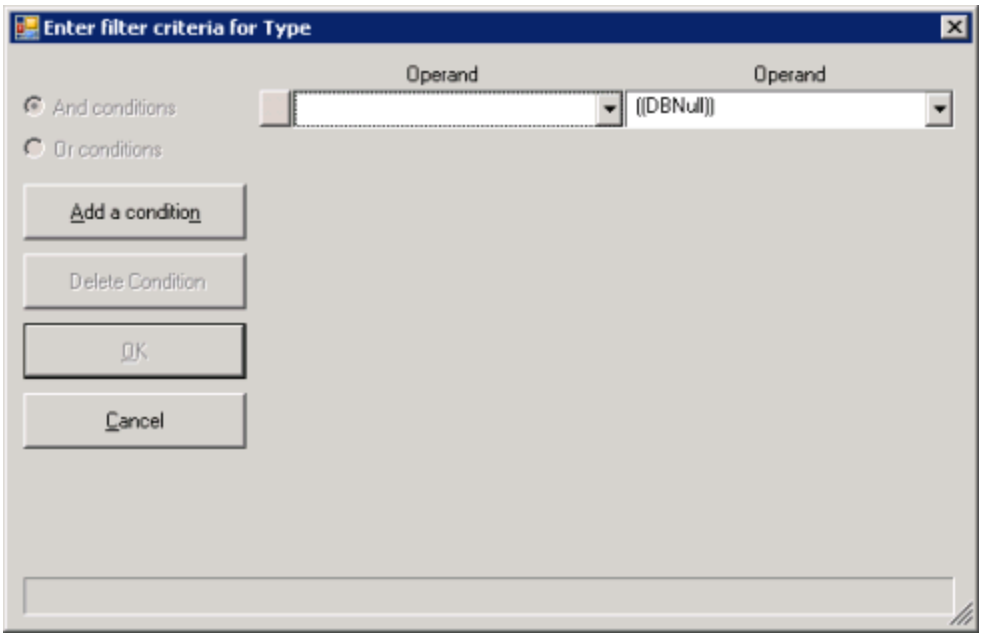


To filter alarms with client-based filtering

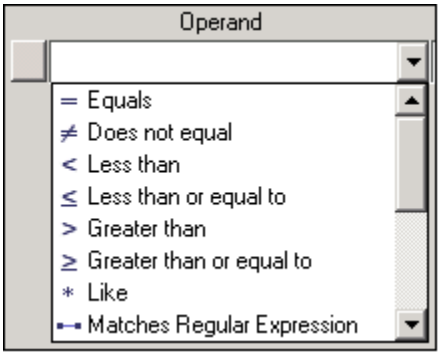
1. Click the filter icon on the column you want to filter by. A menu appears.

2. Select one of the following:
   - **(All)** to switch off filtering
   - **(Custom)** to define a more complex filter
   - **(Blanks)** to filter by blank values
   - **(NonBlanks)** to filter by non blank values
   - A value to filter by the value

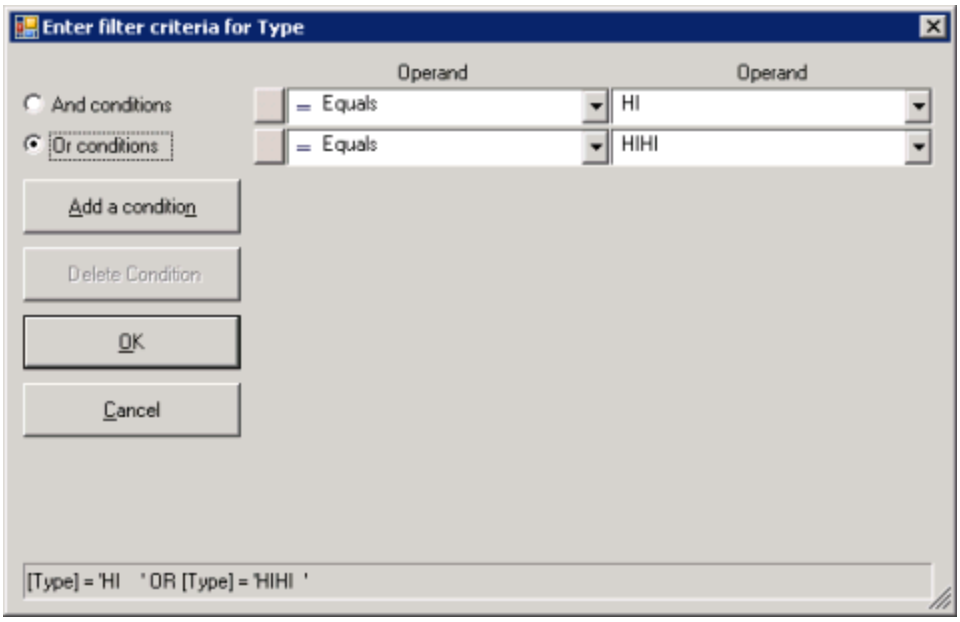
     If you selected **(Custom)**, a dialog box appears.



3. Do one of the following:
   - Select a different operator for the current condition



   - Type or select a different operand for the current condition. The operand can be a value, or the value of a different column in the same row
   - Click **Add a condition** to add more conditions to the filter



- Click **Delete Condition** to delete one or more selected conditions (You can mark the condition by clicking on the button to the left of each condition.)

4. Click **OK**.

Reset the grid

You can reset the Alarm Control's column widths, column order, and names to their last design-time values. When you reset the grid, the query filter is also reset to its default. You can also reset the grid by using the Reset() Method in scripting.

To reset the grid

- Right-click the Alarm Control grid and click **Reset**

Hide alarms

The "hiding" and "unhiding" of alarm records is known in the corresponding InTouch alarm controls as "suppressing" and "unsuppressing".

When the Alarm Control is hiding alarms, it ignores certain alarms. If an alarm matches the exclusion criteria, it is not visible.

The actual alarm generation is completely unaffected by hiding. Alarm records are still logged into the alarm history.

You can hide:

- All alarms, including alarms not visible due to the limited space of the Alarm Control
- All visible alarms
- One or more selected alarms
- All alarms with the same provider names and group names of one or more selected alarms

- All alarms with the same provider names, group names, and within the priority ranges of one or more selected alarms
- All alarms with the same provider names, group names, and tag names within the priority ranges of one or more selected alarms

You can also view which alarms are hidden and unhide them.

To hide all alarms

- Right-click the Alarm Control grid, point to **Hide Others**, and click **Hide All**

To hide all visible alarms

- Right-click the Alarm Control grid, point to **Hide Others**, and click **Hide Visible**

To hide selected alarms

1. Select one or more alarms in alarm state.
2. Right-click the Alarm Control grid and click **Hide Selected**.

To hide alarms with common parameters

1. Select one or more alarms.
2. Right-click the Alarm Control grid, point to **Hide Others**, and click one of the following:
   - **Hide Selected Groups** to hide alarms with the same provider names and group names of one or more selected alarms
   - **Hide Selected Tags** to hide alarms with the same provider names, group names, and tag names within the priority ranges of one or more selected alarms
   - **Hide Selected Priorities** to hide alarms with the same provider names, group names, and within the priority ranges of one or more selected alarms

To unhide alarms

1. Right-click the Alarm Control grid and click **Hidden**. The **Hidden Alarms** dialog box appears.
2. Select the alarms you want to unhide and click **Unhide**.
3. Click **Close**.

Freeze and unfreeze the Alarm Control grid

You can freeze the Alarm Control to prevent the Alarm control tree from being updated with any further changes.

For example, if new alarms occur while the Alarm Control is frozen, the new alarms are only shown after you unfreeze the Alarm Control.

You can configure a time period after which the Alarm Control automatically unfreezes to avoid the Alarm Control being unknowingly frozen. For example, the operator leaves the workstation and returns without realizing that the Alarm Control is still frozen.

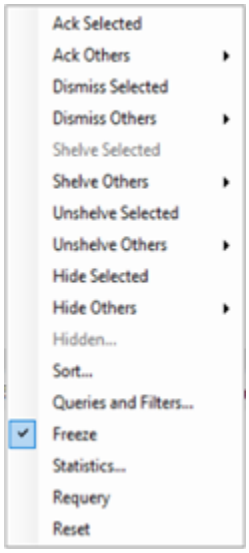
The Alarm Control unfreezes automatically if one of the following changes:

- Alarm Mode
- Alarm Query
- Query Filter

After you unfreeze the Alarm Control, the grid updates with the new alarm records and any other updates while the grid was frozen. You can also use scripting to freeze and unfreeze the Alarm Control grid at run time. For more information, see FreezeDisplay() method.

To freeze or unfreeze the Alarm Control grid

1. Right-click the Alarm Control grid. The shortcut menu appears.



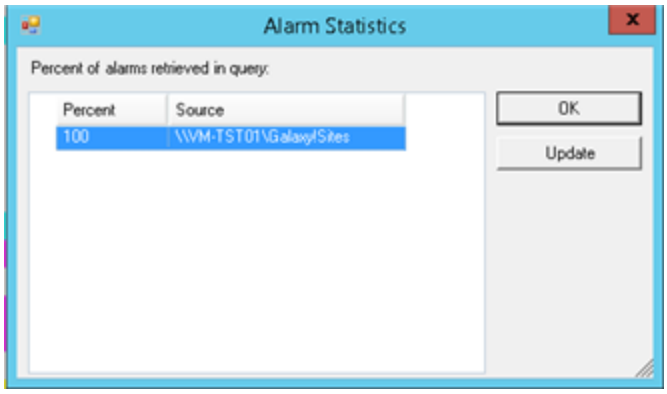
   A check mark next to the **Freeze** option indicates if the grid is currently frozen.
2. Click **Freeze**. The Alarm Control grid is either frozen or unfrozen.

Show alarm statistics

You can view alarm statistics at run time to see which alarm providers are providing the alarm data. For more information, see Show.Statistics() method.

To show alarm statistics

1. Right-click the Alarm Control grid and click **Statistics**. The **Alarm Statistics** dialog box appears.



2. If you use an Alarm Hotbackup name as alarm query, you can expand the Hotbackup name in the **Alarm Statistics** dialog box to show the individual percentages of retrieval for the configured primary and backup alarm provider.
3. Click **Update** to update the statistics.
4. Click **Close**.

Switch between client modes

You can switch between client modes at run time by changing the Alarm Control ClientMode property. The easiest way to do this, is to configure an application server script to interact with the Alarm Control ClientMode property at design time.

To switch between client modes

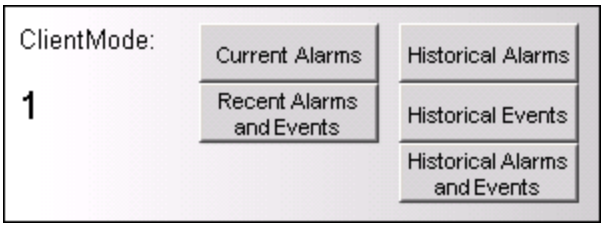
1. Place the Alarm Control on the Industrial graphic.
2. Paste a button on the canvas and change its caption to **Current Alarms**.
3. Double-click the button and configure it with the following action script:

```
AlarmControlGrid1.ClientMode = 1;
```

4. Click **OK**.
5. Repeat steps 2 to 4 for the following buttons:

Button Caption	Action script
Recent Alarms and Events	`AlarmControlGrid1.ClientMode = 2;`
Historical Alarms	`AlarmControlGrid1.ClientMode = 3;`
Historical Events	`AlarmControlGrid1.ClientMode = 4;`
Historical Alarms and Events	`AlarmControlGrid1.ClientMode = 5;`

6. Save and close the Industrial graphic.
7. Create a new managed InTouch application and open it in WindowMaker.
8. Place the Industrial graphic on a new InTouch window.
9. Switch to WindowViewer to test your application.



10. Click **Historical Alarms** to show historical alarms instead of current alarms.

By default, the Alarm Control tries to connect to the alarm database (either the InTouch Database called WWALMDB or the ArchestrA Database called A2ALMDB) on the local computer using the currently logged on user. If you are using a different configuration, you can use value input links or action script to set the following properties:

- Database.ServerName Property
- Database.UserID Property
- Database.Password Property
- Domain Property
- Database.Name Property
- Database.Authentication Property

Switch runtime languages

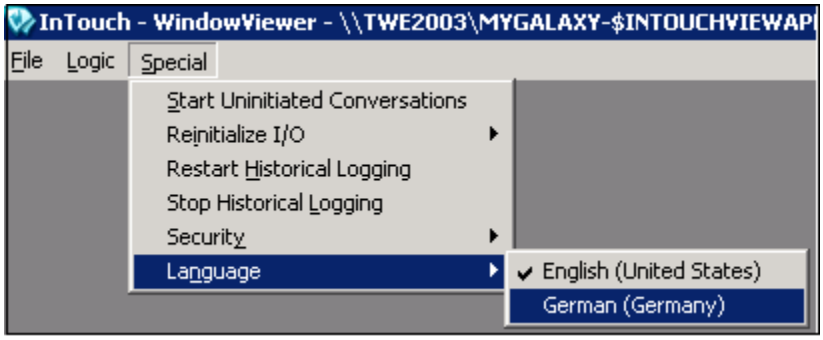
You can switch the language of the Alarm Control in the same way as other parts of your InTouch application. When you switch language, the alarm state, alarm class, alarm type, the various alarm comment fields, the column headers, and the status bar messages are switched to the selected language.

You can also run queries and scripts, which return results in the selected language

To switch the language

Do one of the following:

- In WindowViewer on the **Special** menu, point to **Languages**, and then click the language you want to switch to



- In WindowMaker, use the InTouch QuickScript **SwitchDisplayLanguage** in a button action script to switch the language. At run time, click the button to switch the language
- In WindowMaker, use the system tag $Language in a button action script and assign it to the language code you want to switch to. At run time, click the button to switch the language

For more information about run-time language switching, see *Working with Languages*, in the *Application Server User's Guide*.

Important: If you rename or reorder column headers, you must repeat the symbol text translation procedures. If you do not, your changes will not be available for run-time language switching.

Script the Alarm Control

This section describes the script properties, methods, and events of the Alarm Control.

Alarm Control properties

This section describes the properties available for scripting in the Alarm Control.

AckComment.DefaultValue property

The AckComment.DefaultValue property is a read-write string property that gets or sets the default acknowledgement comment when the AckComment.UseDefault property is TRUE.

Syntax

```
result = AlarmClient.AckComment.DefaultValue;
AlarmClient.AckComment.DefaultValue = ackComment;
```

Example

```
AlarmClient1.AckComment.UseDefault = 1;
AlarmClient1.AckComment.DefaultValue = "This alarm is acknowledged by John Smith";
```

Remarks

For more information, see Show current alarms or recent alarms and events.

AckComment.UseDefault property

The AckComment.UseDefault property is a read-write Boolean property that gets or sets the usage of the default acknowledgement comment.

Syntax

```
result = AlarmClient.AckComment.UseDefault;
AlarmClient.AckComment.UseDefault = useComment;
```

Example

```
AlarmClient1.AckComment.UseDefault = 1;
AlarmClient1.AckComment.DefaultValue = "This alarm is acknowledged by John Smith";
```

Remarks

For more information, see Show current alarms or recent alarms and events.

AckSignature.MaxPriority property

The AckSignature.MaxPriority property is a read-write integer property, which gets or sets the maximum priority value for alarms that require a signature to be acknowledged. The value must range between 1 and 999, and must be greater than or equal to the AckSignature.MinPriority value. The default value of AckSignature.MaxPriority is 999.

Syntax

```
To Set:
<EAC instance name>.<Property Name> = <integer value>;
```

Example

```
AlarmClient1.AckSignature.MaxPriority = 500;
```

Syntax:

```
To Get:
integer <variable name> = <EAC instance name>.<Property Name>;
```

Example

```
dim iMax as integer;
iMax = AlarmClient1.AckSignature.MaxPriority;
```

Remarks

For more information, see Provide a signature to acknowledge alarms.

AckSignature.MinPriority property

The AckSignature.MinPriority property is a read-write integer property, which gets or sets the minimum priority value for alarms that require a signature to be acknowledged. The value must range between 1 and 999, and must be less than or equal to the AckSignature.MaxPriority value. The default value of AckSignature.MinPriority is 1.

Syntax

```
To Set:
<EAC instance name>.<Property Name> = <integer value>;
```

Example

```
AlarmClient1.AckSignature.MinPriority = 100;
```

Syntax

```
To Get:
To Get:
integer <variable name> = <EAC instance name>.<Property Name>;
```

Example

```
dim iMin as integer;
iMin = AlarmClient1.AckSignature.MinPriority;
```

Remarks

For more information, see Provide a signature to acknowledge alarms.

AckSignature.Required property

The AckSignature.Required property is a read-write Boolean property, indicating whether a signature is required for acknowledging the alarms. The default value of AckSignature.MinPriority is True.

Syntax

```
To Set:
<EAC instance name>.<Property Name> = <boolean value>;
```

Example

```
AlarmClient1.AckSignature.Required = true;
```

Syntax

```
To Get:
boolean <variable name> = <EAC instance name>.<Property Name>;
```

Example

```
dim bIsACKSigned as boolean;
bIsACKSigned = AlarmClient1.AckSignature.Required;
```

Remarks

For more information, see Configure the Alarm Control to require a SHELVE signature

AlarmColor.Ack.BackGround property

The AlarmColor.Ack.BackGround property is an array of read-write integer properties that get or set the background colors of all acknowledged alarm records.

Index	Purpose
0	Sets the background color of all acknowledged alarm records in all priority ranges.
1	Gets or sets the background color of acknowledged alarm records in the priority range 1 to AlarmColor.Range[1].
2	Gets or sets the background color of acknowledged alarm records in the priority range AlarmColor.Range[1] to AlarmColor.Range[2].
3	Gets or sets the background color of acknowledged alarm records in the priority range AlarmColor.Range[2] to AlarmColor.Range[3].
4	Gets or sets the background color of acknowledged alarm records in the priority range AlarmColor.Range[3] to 999.

Syntax

```
Color = AlarmClient.AlarmColor.Ack.BackGround[n];
AlarmClient.AlarmColor.Ack.BackGround[n] = Color;
```

Parameters

n

Index from 0 to 4.

Color

Color of background.

Examples

```
AlarmClient1.AlarmColor.Ack.BackGround[0] = Color.Red;
AlarmClient1.AlarmColor.Ack.BackGround[1] = Color.FromARGB(0,128,0);
AlarmClient1.AlarmColor.Ack.BackGround[2] = Color.Grey;
```

```
AlarmClient1.AlarmColor.Ack.BackGround[3] = Color.Yellow;
AlarmClient1.AlarmColor.Ack.BackGround[4] = Color.Black;
```

Remarks

Color is a .NET Framework data type. You can use various Color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName().

For a list of the .NET color names and the hexadecimal codes, see .NET colors.

For more information on the color methods, see the online Microsoft documentation for .NET Framework Development.

AlarmColor.Ack.ForeGround property

The AlarmColor.Ack.ForeGround property is an array of read-write integer properties that get or set the text colors of all acknowledged alarm records.

Index	Purpose
0	Sets the text color of all acknowledged alarm records in all priority ranges.
1	Gets or sets the text color of acknowledged alarm records in the priority range 1 to AlarmColor.Range[1].
2	Gets or sets the text color of acknowledged alarm records in the priority range AlarmColor.Range[1] to AlarmColor.Range[2].
3	Gets or sets the text color of acknowledged alarm records in the priority range AlarmColor.Range[2] to AlarmColor.Range[3].
4	Gets or sets the text color of acknowledged alarm records in the priority range AlarmColor.Range[3] to 999.

Syntax

```
Color = AlarmClient.AlarmColor.Ack.ForeGround[n];
AlarmClient.AlarmColor.Ack.ForeGround[n] = Color;
```

Parameters

n

Index from 0 to 4.

Color

Color of text.

Examples

```
AlarmClient1.AlarmColor.Ack.ForeGround[0] = Color.Black;
AlarmClient1.AlarmColor.Ack.ForeGround[1] = Color.Blue;
AlarmClient1.AlarmColor.Ack.ForeGround[2] = Color.Green;
AlarmClient1.AlarmColor.Ack.ForeGround[3] = Color.Yellow;
AlarmClient1.AlarmColor.Ack.ForeGround[4] = Color.FromARGB(0,128,0);
```

Remarks

Color is a .NET Framework data type. You can use various Color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName().

For a list of the .NET color names and the hexadecimal codes, see .NET colors.

For more information on the color methods, see the online Microsoft documentation for .NET Framework Development.

AlarmColor.Ack.RTN.BackGround property

The AlarmColor.Ack.RTN.BackGround property is a read-write color property that gets or sets the background color of acknowledged alarm records that "return to normal" (ACK_RTN).

Syntax

```
Color = AlarmClient.AlarmColor.Ack.RTN.BackGround;
AlarmClient.AlarmColor.Ack.RTN.BackGround = Color;
```

Parameters

Color

Color of background.

Return Value

Returns the background color of acknowledged alarms that "return to normal".

Example

```
AlarmClient1.AlarmColor.Ack.RTN.BackGround = Color.Blue;
```

Remarks

Color is a .NET Framework data type. You can use various Color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName().

For a list of the .NET color names and the hexadecimal codes, see .NET colors.

For more information on the color methods, see the online Microsoft documentation for .NET Framework

Development.

AlarmColor.Ack.RTN.ForeGround property

The AlarmColor.Ack.RTN.ForeGround property is a read-write color property that gets or sets the text color of acknowledged alarm records that "return to normal" (ACK_RTN).

Syntax

```
Color = AlarmClient.AlarmColor.Ack.RTN.ForeGround;
AlarmClient.AlarmColor.Ack.RTN.ForeGround = Color;
```

Parameters

Color

Color of text.

Example

```
AlarmClient1.AlarmColor.Ack.RTN.ForeGround = Color.Black;
```

Remarks

Color is a .NET Framework data type. You can use various Color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName().

For a list of the .NET color names and the hexadecimal codes, see .NET colors.

For more information on the color methods, see the online Microsoft documentation for .NET Framework Development.

AlarmColor.Range property

The AlarmColor.Range property is an array of read-write integer properties that get or set the boundaries of the priority ranges.

You can use priority ranges to classify, group, and emphasize alarms and events belonging to a certain priority range.

The boundaries must fulfill the following condition:

1 < Range[1] < Range[2] < Range[3] < 999

By default, the boundaries are set as follows:

• AlarmColor.Range[1]	250
• AlarmColor.Range[2]	500

• AlarmColor.Range[3]	750

Syntax

```
RangeN = AlarmClient.AlarmColor.Range[N];
AlarmClient.AlarmColor.Range[1] = RangeN;
```

Parameters

N

Range index 1, 2, or 3.

Example

The following example defines four priority ranges (1 to 50, 51 to 600, 601 to 800, and 801 to 999):

```
AlarmClient1.AlarmColor.Range[1] = 50;
AlarmClient1.AlarmColor.Range[2] = 600;
AlarmClient1.AlarmColor.Range[3] = 800;
```

Remarks

For more information, see Set priority ranges for alarm records

AlarmColor.RTN.BackGround property

The AlarmColor.RTN.BackGround property is a read-write color property that gets or sets the background color of alarm records that "return to normal" (ACK_RTN and UNACK_RTN).

Syntax

```
Color = AlarmClient.AlarmColor.RTN.BackGround;
AlarmClient.AlarmColor.RTN.BackGround = Color;
```

Parameters

Color

Color of background.

Example

```
AlarmClient1.AlarmColor.RTN.BackGround = Color.Blue;
```

Remarks

For more information, see Set alarm return to normal record colors.

Color is a .NET Framework data type. You can use various Color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName().

For a list of the .NET color names and the hexadecimal codes, see .NET colors.

For more information on the color methods, see the online Microsoft documentation for .NET Framework Development.

AlarmColor.RTN.ForeGround property

The AlarmColor.RTN.ForeGround property is a read-write color property that gets or sets the text color of alarm records that "return to normal" (ACK_RTN and UNACK_RTN).

Syntax

```
Color = AlarmClient.AlarmColor.RTN.ForeGround;
AlarmClient.AlarmColor.RTN.ForeGround = Color;
```

Parameters

Color

Color of text.

Example

```
AlarmClient1.AlarmColor.RTN.ForeGround = Color.Yellow;
```

Remarks

For more information, see Set alarm return to normal record colors.

Color is a .NET Framework data type. You can use various Color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName().

For a list of the .NET color names and the hexadecimal codes, see .NET colors.

For more information on the color methods, see the online Microsoft documentation for .NET Framework Development.

AlarmColor.UnAck.BackGround property

The AlarmColor.UnAck.BackGround property is an array of read-write integer properties that get or set the background colors of all unacknowledged alarm records.

Index	Purpose
0	Sets the background color of all unacknowledged alarm records in all priority ranges.
1	Gets or sets the background color of unacknowledged alarm records in the priority range 1 to AlarmColor.Range[1].
2	Gets or sets the background color of unacknowledged alarm records in the priority range AlarmColor.Range[1] to AlarmColor.Range[2].
3	Gets or sets the background color of unacknowledged alarm records in the priority range AlarmColor.Range[2] to AlarmColor.Range[3].
4	Gets or sets the background color of unacknowledged alarm records in the priority range AlarmColor.Range[3] to 999.

Syntax

```
Color = AlarmClient.AlarmColor.UnAck.BackGround[n];
AlarmClient.AlarmColor.UnAck.BackGround[n] = Color;
```

Parameters

n

Index from 0 to 4.

Color

Color of background.

Example

```
AlarmClient1.AlarmColor.UnAck.BackGround[0] = Color.Blue;
AlarmClient1.AlarmColor.UnAck.BackGround[1] = Color.ARGB(223,113,76);
AlarmClient1.AlarmColor.UnAck.BackGround[2] = Color.Yellow;
AlarmClient1.AlarmColor.UnAck.BackGround[3] = Color.Green;
AlarmClient1.AlarmColor.UnAck.BackGround[4] = Color.White;
```

Remarks

Color is a .NET Framework data type. You can use various Color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName().

For a list of the .NET color names and the hexadecimal codes, see .NET colors

For more information on the color methods, see the online Microsoft documentation for .NET Framework

Development.

AlarmColor.UnAck.Flash.BackGround property

The AlarmColor.UnAck.Flash.BackGround property is an array of read-write color properties that get or set the background colors of all flashing unacknowledged alarm records.

Index	Purpose
0	Sets the background color of all flashing unacknowledged alarm records in all priority ranges.
1	Gets or sets the background color of flashing unacknowledged alarm records in the priority range 1 to AlarmColor.Range[1].
2	Gets or sets the background color of flashing unacknowledged alarm records in the priority range AlarmColor.Range[1] to AlarmColor.Range[2].
3	Gets or sets the background color of flashing unacknowledged alarm records in the priority range AlarmColor.Range[2] to AlarmColor.Range[3].
4	Gets or sets the background color of flashing unacknowledged alarm records in the priority range AlarmColor.Range[3] to 999.

Syntax

```
Color = AlarmClient.AlarmColor.UnAck.Flash.BackGround[n];
AlarmClient.AlarmColor.UnAck.Flash.BackGround[n] = Color;
```

Parameters

n

Index from 1 to 4.

Color

Color of background.

Example

```
AlarmClient1.AlarmColor.UnAck.Flash.BackGround[1] = Color.ARGB(223,113,76);
AlarmClient1.AlarmColor.UnAck.Flash.BackGround[2] = Color.Yellow;
AlarmClient1.AlarmColor.UnAck.Flash.BackGround[3] = Color.Green;
AlarmClient1.AlarmColor.UnAck.Flash.BackGround[4] = Color.White;
```

Remarks

Color is a .NET Framework data type. You can use various Color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName().

For a list of the .NET color names and the hexadecimal codes, see .NET colors

For more information on the color methods, see the online Microsoft documentation for .NET Framework Development.

AlarmColor.UnAck.Flash.ForeGround property

The AlarmColor.UnAck.Flash.ForeGround property is an array of read-write color properties that get or set the text colors of all flashing unacknowledged alarm records.

Index	Purpose
0	Sets the text color of all flashing unacknowledged alarm records in all priority ranges.
1	Gets or sets the text color of flashing unacknowledged alarm records in the priority range 1 to AlarmColor.Range[1].
2	Gets or sets the text color of flashing unacknowledged alarm records in the priority range AlarmColor.Range[1] to AlarmColor.Range[2].
3	Gets or sets the text color of flashing unacknowledged alarm records in the priority range AlarmColor.Range[2] to AlarmColor.Range[3].
4	Gets or sets the text color of flashing unacknowledged alarm records in the priority range AlarmColor.Range[3] to 999.

Syntax

```
Color = AlarmClient.AlarmColor.UnAck.Flash.ForeGround[n];
AlarmClient.AlarmColor.UnAck.Flash.ForeGround[n] = Color;
```

Parameters

n

Index from 1 to 4.

Color

Color of text.

Examples

```
AlarmClient1.AlarmColor.UnAck.Flash.ForeGround[1] = Color.ARGB(223,113,76);
AlarmClient1.AlarmColor.UnAck.Flash.ForeGround[2] = Color.Yellow;
AlarmClient1.AlarmColor.UnAck.Flash.ForeGround[3] = Color.Green;
AlarmClient1.AlarmColor.UnAck.Flash.ForeGround[4] = Color.White;
```

Remarks

Color is a .NET Framework data type. You can use various Color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName().

For a list of the .NET color names and the hexadecimal codes, see .NET colors

For more information on the color methods, see the online Microsoft documentation for .NET Framework Development.

AlarmColor.UnAck.ForeGround property

The AlarmColor.UnAck.ForeGround property is an array of read-write integer properties that get or set the text colors of all unacknowledged alarm records.

Index	Purpose
0	Sets the text color of all unacknowledged alarm records in all priority ranges.
1	Gets or sets the text color of unacknowledged alarm records in the priority range 1 to AlarmColor.Range[1].
2	Gets or sets the text color of unacknowledged alarm records in the priority range AlarmColor.Range[1] to AlarmColor.Range[2].
3	Gets or sets the text color of unacknowledged alarm records in the priority range AlarmColor.Range[2] to AlarmColor.Range[3].
4	Gets or sets the text color of unacknowledged alarm records in the priority range AlarmColor.Range[3] to 999.

Syntax

```
Color = AlarmClient.AlarmColor.UnAck.ForeGround[n];
AlarmClient.AlarmColor.UnAck.ForeGround[n] = Color;
```

Parameters

n

Index from 0 to 4.

Color

Color of text.

Example

```
AlarmClient1.AlarmColor.UnAck.ForeGround[0] = Color.Blue;
AlarmClient1.AlarmColor.UnAck.ForeGround[1] = Color.ARGB(223,113,76);
AlarmClient1.AlarmColor.UnAck.ForeGround[2] = Color.Yellow;
AlarmClient1.AlarmColor.UnAck.ForeGround[3] = Color.Green;
AlarmClient1.AlarmColor.UnAck.ForeGround[4] = Color.White;
```

Remarks

Color is a .NET Framework data type. You can use various Color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName().

For a list of the .NET color names and the hexadecimal codes, see .NET colors

For more information on the color methods, see the online Microsoft documentation for .NET Framework Development.

AlarmColor.UnAck.RTN.BackGround property

The AlarmColor.UnAck.RTN.BackGround property is a read-write color property that gets or sets the background color of unacknowledged alarm records that "return to normal" (UNACK_RTN).

Syntax

```
Color = AlarmClient.AlarmColor.UnAck.RTN.BackGround;
AlarmClient.AlarmColor.UnAck.RTN.BackGround = Color;
```

Parameters

Color

Color of background.

Example

```
AlarmClient1.AlarmColor.UnAck.RTN.BackGround = Color.Blue;
```

Remarks

Color is a .NET Framework data type. You can use various Color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName().

For a list of the .NET color names and the hexadecimal codes, see .NET colors

For more information on the color methods, see the online Microsoft documentation for .NET Framework

Development.

AlarmColor.UnAck.RTN.ForeGround property

The AlarmColor.UnAck.RTN.ForeGround property is a read-write color property that gets or sets the text color of unacknowledged alarm records that "return to normal" (UNACK_RTN).

Syntax

```
Color = AlarmClient.AlarmColor.UnAck.RTN.ForeGround;
AlarmClient.AlarmColor.UnAck.RTN.ForeGround = Color;
```

Parameters

Color

Color of text.

Example

```
AlarmClient1.AlarmColor.UnAck.RTN.ForeGround = Color.FromARGB(0,0,0);
```

Remarks

Color is a .NET Framework data type. You can use various Color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName().

For a list of the .NET color names and the hexadecimal codes, see .NET colors

For more information on the color methods, see the online Microsoft documentation for .NET Framework Development.

AlarmQuery property

The AlarmQuery property is a read-write string property that gets or sets the selected alarm query.

Syntax

```
result = AlarmClient.AlarmQuery;
AlarmClient.AlarmQuery = AlmQry;
```

Parameters

AlmQry

Alarm query string in format \\node\provider!group where node is optional.

Example

```
AlarmClient.AlarmQuery = "\intouch!GroupA";
```

Remarks

When a new query is selected the AlarmQuery property is updated with the selected query. The selected query will be updated if a new query string is written to the AlarmQuery property.

Note: Empty string is not allowed for AlarmQuery property when the Alarm Client Control client mode is configured as Current Alarms or Recent Alarms and Events.

AllowColumnResize property

The AllowColumnResize property is a read-write Boolean property that gets or sets the ability to resize the columns at run time.

Syntax

```
result = AlarmClient.AllowColumnResize;
AlarmClient.AllowColumnResize = allowColResizing;
```

AutoResumeDuration property

The AutoResumeDuration property is a read-write integer property that gets or sets the time in seconds after which the grid becomes unfrozen and resumes showing alarms.

Set this value to 0 to disable auto resume.

Syntax

```
result = AlarmClient.AutoResumeDuration;
AlarmClient.AllowColumnResize = timeout;
```

AutoScroll property

The AutoScroll property is a read-write Boolean property that gets or sets automatic scrolling to new alarms.

Syntax

```
result = AlarmClient.AutoScroll;
AlarmClient.AutoScroll = allowAutoscroll;
```

ClientMode property

The ClientMode property is a read-write integer property that gets or sets the client mode for the Alarm Control.

Use one of the following values:

Value	Client Mode
1	Current Alarms
2	Recent Alarms and Events
3	Historical Alarms
4	Historical Events
5	Historical Alarms and Events

Syntax

```
result = AlarmClient.ClientMode;
AlarmClient.ClientMode = clientMode;
```

Example

```
AlarmClient1.ClientMode = 2;
LogMessage("Alarm client set to Recent Alarms and Events");
```

Remarks

For more information, see Show current alarms or recent alarms and events.

ConnectStatus property

The ConnectStatus property is a read-only string property that gets the status of the connection to the Alarm Database.

Syntax

```
result = AlarmClient.ConnectStatus;
```

Return Value

Returns the status of the connection to the alarm database. Can be "Connected," "Not connected," or "In progress."

Example

```
alive = AlarmClient1.ConnectStatus;
if alive == "Connected" then
   LogMessage("The Alarm Control is currently connected to the Alarm Database");
```

```
else
   LogMessage("The Alarm Control is either currently connecting to the Alarm Database or
   not connected.");
endif;
```

ContextMenu.AckAll property

The ContextMenu.AckAll property is a read-write Boolean property that gets or sets the appearance of the **Ack All** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.AckAll;
AlarmClient.ContextMenu.AckAll = AckAllVis;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.AckOthers property

The ContextMenu.AckOthers property is a read-write Boolean property that gets or sets the appearance of the **Ack Others** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.AckOthers;
AlarmClient.ContextMenu.AckOthers = AckOthersVis;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.AckSelected property

The ContextMenu.AckSelected property is a read-write Boolean property that gets or sets the appearance of the **Ack Selected** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.AckSelected;
AlarmClient.ContextMenu.AckSelected = AckSelectedVis;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.AckSelectedGroups property

The ContextMenu.AckSelectedGroups property is a read-write Boolean property that gets or sets the appearance of the **Ack Selected Groups** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.AckSelectedGroups;
AlarmClient.ContextMenu.AckSelectedGroups = AckSelGrpsVis;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.AckSelectedPriorities property

The ContextMenu.AckSelectedPriorities property is a read-write Boolean property that gets or sets the appearance of the **Ack Selected Priorities** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.AckSelectedPriorities;
AlarmClient.ContextMenu.AckSelectedPriorities = AckSelPriVis;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.AckSelectedTags property

The ContextMenu.AckSelectedTags property is a read-write Boolean property that gets or sets the appearance of the **Ack Selected Tags** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.AckSelectedTags;
AlarmClient.ContextMenu.AckSelectedTags = AckSelTagsVis;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.AckVisible property

The ContextMenu.AckVisible property is a read-write Boolean property that gets or sets the appearance of the **Ack Visible** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.AckVisible;
AlarmClient.ContextMenu.AckVisible = AckVisVis;
```

Remarks

For more information, see Configure the runtime shortcut menu..

ContextMenu.Favorites property

The ContextMenu.Favorites property is a read-write Boolean property that gets or sets the appearance of the **Query and Filter Favorites** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.Favorites;
AlarmClient.ContextMenu.Favorites = FavsVis;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.Freeze property

The ContextMenu.Freeze property is a read-write Boolean property that gets or sets the appearance of the **Freeze** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.Freeze;
AlarmClient.ContextMenu.Freeze = FreezeVis;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.Hidden property

The ContextMenu.Hidden property is a read-write Boolean property that gets or sets the appearance of the **Hidden** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.Hidden;
AlarmClient.ContextMenu.Hidden = HiddenVis;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.HideAll property

The ContextMenu.HideAll property is a read-write Boolean property that gets or sets the appearance of the **Hide All** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.HideAll;
AlarmClient.ContextMenu.HideAll = HideAllVis;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.HideOthers property

The ContextMenu.HideOthers property is a read-write Boolean property that gets or sets the appearance of the **Hide Others** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.HideOthers;
AlarmClient.ContextMenu.HideOthers = HideOthersVis;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.HideSelected property

The ContextMenu.HideSelected property is a read-write Boolean property that gets or sets the appearance of the **Hide Selected** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.HideSelected;
AlarmClient.ContextMenu.HideSelected = HideSelVis;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.HideSelectedGroups property

The ContextMenu.HideSelectedGroups property is a read-write Boolean property that gets or sets the appearance of the **Hide Selected Groups** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.HideSelectedGroups;
AlarmClient.ContextMenu.HideSelectedGroups = HideSelGrpsVis;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.HideSelectedPriorities property

The ContextMenu.HideSelectedPriorities property is a read-write Boolean property that gets or sets the appearance of the **Hide Selected Priorities** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.HideSelectedPriorities;
AlarmClient.ContextMenu.HideSelectedPriorities = HideSelPrisVis;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.HideSelectedTags property

The ContextMenu.HideSelectedTags property is a read-write Boolean property that gets or sets the appearance of the **Hide Selected Tags** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.HideSelectedTags;
AlarmClient.ContextMenu.HideSelectedTags = HideSelTagsVis;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.HideVisible property

The ContextMenu.HideVisible property is a read-write Boolean property that gets or sets the appearance of the **Hide Visible** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.HideVisible;
AlarmClient.ContextMenu.HideVisible = HideVisVis;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.Requery property

The ContextMenu.Requery property is a read-write Boolean property that gets or sets the appearance of the **Requery** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.Requery;
AlarmClient.ContextMenu.Requery = RequeryVis;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.Reset property

The ContextMenu.Reset property is a read-write Boolean property that gets or sets the appearance of the **Reset** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.Reset;
AlarmClient.ContextMenu.Reset = ResetVis;
```

Remarks

For more information, see Configure the runtime shortcut menu

ContextMenu.ShelveAll property

CntextMenu.ShelveAll is a read-write Boolean property that gets or sets the appearance of the **Shelve All** option on the Alarm Control's shortcut menu.

Syntax

```
To Set:
AlarmClient.ContextMenu.ShelveAll = boolean_value;
```

Example

```
AlarmClient1.ContextMenu.ShelveAll = true;
```

Syntax

```
To Get:
boolean_name = AlarmClient.ContextMenu.ShelveAll;
```

Example

```
dim ShlvAll as boolean;
ShlvAll = AlarmClient1.ContextMenu.ShelveAll;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.ShelveOthers property

ContextMenu.ShelveOthers is a read-write Boolean property that gets or sets the appearance of the **Shelve**

Others option on the Alarm Control's shortcut menu.

Syntax

```
To Set:
AlarmClient.ContextMenu.ShelveOthers = boolean_value;
```

Example

```
AlarmClient1.ContextMenu.ShelveOthers = true;
```

Syntax

```
To Get:
boolean_name = AlarmClient.ContextMenu.ShelveOthers;
```

Example

```
dim ShlvOthrs as boolean;
ShlvOthrs = AlarmClient1.ContextMenu.ShelveOthers;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.ShelveSelected property

ContextMenu.ShelveSelected is a read-write Boolean property that gets or sets the appearance of the **Shelve Selected** option on the Alarm Control's shortcut menu.

Syntax

```
To Set:
AlarmClient.ContextMenu.ShelveSelected = boolean_value;
```

Example

```
AlarmClient1.ContextMenu.ShelveSelected = true;
```

Syntax

```
To Get:
boolean_name = AlarmClient.ContextMenu.ShelveSelected;
```

Example

```
dim ShlvSlctd as boolean;
ShlvSlctd = AlarmClient1.ContextMenu.ShelveSelected;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.ShelveSelectedGroups property

ContextMenu.ShelveSelectedGroups is a read-write Boolean property that gets or sets the appearance of the **Shelve Selected Groups** option on the Alarm Control's shortcut menu.

Syntax

```
To Set:
AlarmClient.ContextMenu.ShelveSelectedGroups = boolean_value;
```

Example

```
AlarmClient1.ContextMenu.ShelveSelectedGroups = true;
```

Syntax

```
To Get:
boolean_name = AlarmClient.ContextMenu.ShelveSelectedGroups;
```

Example

```
dim ShlvSlctdGrp as boolean;
ShlvSlctdGrp = AlarmClient1.ContextMenu.ShelveSelectedGroups;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.ShelveSelectedPriorities property

ContextMenu.ShelveSelectedPriorities is a read-write Boolean property that gets or sets the appearance of the **Shelve Selected Priorities** option on the Alarm Control's shortcut menu.

Syntax

```
To Set:
AlarmClient.ContextMenu.ShelveSelectedPriorities = boolean_value;
```

Example

```
AlarmClient1.ContextMenu.ShelveSelectedPriorities = true;
```

Syntax

```
To Get:
boolean_name = AlarmClient.ContextMenu.ShelveSelectedPriorities;
```

Example

```
dim ShlvSlctdPri as boolean;
ShlvSlctdPri = AlarmClient1.ContextMenu.ShelveSelectedPriorities;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.ShelveSelectedSeverities property

ContextMenu.ShelveSelectedSeverities is a read-write Boolean property that gets or sets the appearance of the **Shelve Selected Severities** option on the Alarm Control's shortcut menu.

Syntax

```
To Set:
AlarmClient.ContextMenu.ShelveSelectedSeverities = boolean_value;
```

Example

```
AlarmClient1.ContextMenu.ShelveSelectedSeverities = true;
```

Syntax

```
To Get:
boolean_name = AlarmClient.ContextMenu.ShelveSelectedSeverities;
```

Example

```
dim ShlvSlctdSev as boolean;
ShlvSlctdSev = AlarmClient1.ContextMenu.ShelveSelectedSeverities;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.ShelveSelectedTags property

ContextMenu.ShelveSelectedTags is a read-write Boolean property that gets or sets the appearance of the **Shelve Selected Tags** option on the Alarm Control's shortcut menu.

Syntax

```
To Set:
AlarmClient.ContextMenu.ShelveSelectedTags = boolean_value;
```

Example

```
AlarmClient1.ContextMenu.ShelveSelectedTags = true;
```

Syntax

```
To Get:
boolean_name = AlarmClient.ContextMenu.ShelveSelectedTags;
```

Example

```
dim ShlvSlctdTag as boolean;
ShlvSlctdTag = AlarmClient1.ContextMenu.ShelveSelectedTags;
```

Remarks

For more information, see Configure the runtime shortcut menu

ContextMenu.ShelveVisible property

ContextMenu.ShelveVisible is a read-write Boolean property that gets or sets the appearance of the **Shelve Visible** option appears in the Alarm Control's shortcut menu.

Syntax

```
To Set:
AlarmClient.ContextMenu.ShelveSelectedVisible = boolean_value;
```

Example

```
AlarmClient1.ContextMenu.ShelveSelectedVisible = true;
```

Syntax

```
To Get:
boolean_name = AlarmClient.ContextMenu.ShelveSelectedVisible;
```

Example

```
dim ShlvSlctdVis as boolean;
ShlvSlctdVis = AlarmClient1.ContextMenu.ShelveSelectedVisible;
```

Remarks

For more information, see Configure the runtime shortcut menu

ContextMenu.Sort property

The ContextMenu.Sort property is a read-write Boolean property that gets or sets the appearance of the **Sort** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.Sort;
AlarmClient.ContextMenu.Sort = SortVis;
```

Remarks

For more information, see Configure the runtime shortcut menu

ContextMenu.Statistics property

The ContextMenu.Statistics property is a read-write Boolean property that gets or sets the appearance of the **Statistics** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.Statistics;
AlarmClient.ContextMenu.Statistics = StatsVis;
```

Remarks

For more information, see Configure the runtime shortcut menu

ContextMenu.UnhideAll property

The ContextMenu.UnhideAll property is a read-write Boolean property that gets or sets the appearance of the **Unhide All** option on the shortcut menu.

Syntax

```
result = AlarmClient.ContextMenu.UnhideAll;
```

```
AlarmClient.ContextMenu.UnhideAll = UnhideAllVis;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.UnshelveAll property

ContextMenu.UnshelveAll is read-write Boolean property that gets or sets the appearance of the **Unshelve All** option on the Alarm Control's shortcut menu.

Syntax

```
To Set:
AlarmClient.ContextMenu.UnshelveAll = boolean_value;
```

Example

```
AlarmClient1.ContextMenu.UnshelveAll = true;
```

Syntax

```
To Get:
boolean_variable_name = AlarmClient.ContextMenu.UnshelveAll;
```

Example

```
dim ShlvAll as boolean;
ShlvAll = AlarmClient1.ContextMenu.UnshelveAll;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.UnshelveOthers property

ContextMenu.UnshelveOthers is read-write Boolean property that gets or sets the appearance of the **Unshelve Others** option on the Alarm Control's shortcut menu.

Syntax

```
To Set:
AlarmClient.ContextMenu.UnshelveOthers = boolean_value;
```

Example

```
AlarmClient1.ContextMenu.UnshelveOthers = true;
```

Syntax

```
To Get:
boolean_variable_name = AlarmClient.ContextMenu.UnshelveOthers;
```

Example

```
dim ShlvOthrs as boolean;
ShlvOthrs = AlarmClient1.ContextMenu.UnshelveOthers;
```

Remarks

For more information, see Configure the runtime shortcut menu.

ContextMenu.UnshelveSelected property

ContextMenu.UnshelveSelected is read-write Boolean property that gets or sets the appearance of the **Unshelve Selected** option on the Alarm Control's shortcut menu.

Syntax

```
To Set:
AlarmClient.ContextMenu.UnshelveSelected = boolean_value;
```

Example

```
AlarmClient1.ContextMenu.UnshelveSelected = true;
```

Syntax

```
To Get:
boolean_variable_name = AlarmClient.ContextMenu.UnshelveSelected;
```

Example

```
dim ShlvSlctd as boolean;
ShlvSlctd = AlarmClient1.ContextMenu.UnshelveSelected;
```

Remarks

For more information, see Configure the runtime shortcut menu

ContextMenu.UnshelveSelectedGroups property

ContextMenu.UnshelveSelectedGroups is read-write Boolean property that gets or sets the appearance of the **Unshelve Selected Groups** option on the Alarm Control's shortcut menu.

Syntax

```
To Set:
AlarmClient.ContextMenu.UnshelveSelectedGroups = boolean_value;
```

Example

```
AlarmClient1.ContextMenu.UnshelveSelectedGroups = true;
```

Syntax

```
To Get:
boolean_name = AlarmClient.ContextMenu.UnshelveSelectedGroups;
```

Example

```
dim UnshlvSlctdGrp as boolean;
UnshlvSlctdGrp = AlarmClient1.ContextMenu.UnshelveSelectedGroups;
```

Remarks

For more information, see Configure the runtime shortcut menu

ContextMenu.UnshelveSelectedPriorities property

ContextMenu.UnshelveSelectedPriorities is read-write Boolean property that gets or sets the appearance of the **Unshelve Selected Priorities** option on the Alarm Control's shortcut menu.

Syntax

```
To Set:
AlarmClient.ContextMenu.UnshelveSelectedPriorities = boolean_value;
```

Example

```
AlarmClient1.ContextMenu.UnshelveSelectedPriorities = true;
```

Syntax

```
To Get:
boolean_name = AlarmClient.ContextMenu.UnshelveSelectedPriorities;
```

Example

```
dim UnshlvSlctdPri as boolean;
UnshlvSlctdPri = AlarmClient1.ContextMenu.UnshelveSelectedPriorities;
```

Remarks

For more information, see Configure the runtime shortcut menu

ContextMenu.UnshelveSelectedSeverities property

ContextMenu.UnshelveSelectedSeverities is read-write Boolean property that gets or sets the appearance of the **Unshelve Selected Severities** option on the Alarm Control's shortcut menu.

Syntax

```
To Set:
AlarmClient.ContextMenu.UnshelveSelectedSeverities = boolean_value;
```

Example

```
AlarmClient1.ContextMenu.UnshelveSelectedSeverities = true;
```

Syntax

```
To Get:
boolean_name = AlarmClient.ContextMenu.UnshelveSelectedSeverities;
```

Example

```
dim UnshlvSlctdSev as boolean;
UnshlvSlctdSev = AlarmClient1.ContextMenu.UnshelveSelectedSeverities;
```

Remarks

For more information, see Configure the runtime shortcut menu

ContextMenu.UnshelveSelectedTags property

ContextMenu.UnshelveSelectedTags is read-write Boolean property that gets or sets the appearance of the **Unshelve Selected Tags** option on the Alarm Control's shortcut menu.

Syntax

```
To Set:
AlarmClient.ContextMenu.UnshelveSelectedTags = boolean_value;
```

Example

```
AlarmClient1.ContextMenu.UnshelveSelectedTags = true;
```

Syntax

```
To Get:
boolean_name = AlarmClient.ContextMenu.UnshelveSelectedTags;
```

Example

```
dim UnshlvSlctdTag as boolean;
UnshlvSlctdTag = AlarmClient1.ContextMenu.UnshelveSelectedTags;
```

Remarks

For more information, see Configure the runtime shortcut menu

ContextMenu.UnshelveVisible property

ContextMenu.UnshelveVisible is read-write Boolean property that gets or sets the appearance of the **Unshelve Visible** option on the Alarm Control's shortcut menu.

Syntax

```
To Set:
AlarmClient.ContextMenu.UnshelveSelectedVisible = boolean_value;
```

Example

```
AlarmClient1.ContextMenu.UnshelveSelectedVisible = true;
```

Syntax

```
To Get:
boolean_name = AlarmClient.ContextMenu.UnshelveSelectedVisible;
```

Example

```
dim UnshlvSlctdVis as boolean;
UnshlvSlctdVis = AlarmClient1.ContextMenu.UnshelveSelectedVisible;
```

Remarks

For more information, see Configure the runtime shortcut menu

Database.Authentication property

The Database.Authentication property is a read-write string property that gets or sets the authentication mode to connect to the Alarm Database. Possible values are:

- Windows Integrated
- Windows Account
- SQL Server

The default value is "Windows Integrated".

Syntax

```
result = AlarmClient.Database.Authentication;
AlarmClient.Database.Authentication = AuthMode;
```

Example

```
AlarmClient.Database.Authentication = "Windows Integrated";
```

Remarks

For more information, see Show historical alarms and/or events.

Database.Name property

The Database.Name property is a read-write string property that gets or sets the name of the Alarm Database. The default value is "WWALMDB". WWALMDB is the name of the InTouch Database and A2ALMDB is the name of the ArchestrA Database.

If you change the Database.Name property at run time, you need to call the Connect method to connect to the new alarm database.

Syntax

```
result = AlarmClient.Database.Name;
AlarmClient.Database.Name = AlmDBName;
```

Remarks

For more information, see Show historical alarms and/or events.

Database.Password property

The Database.Password property is a read-write string property that gets or sets the password associated with the user name to connect to the Alarm Database.

Syntax

```
result = AlarmClient.Database.Password;
AlarmClient.Database.Password = Psswrd;
```

Remarks

For more information, see Show historical alarms and/or events

Database.ServerName property

The Database.ServerName property is a read-write string property that gets or sets the name of the server that hosts the Alarm Database.

Syntax

```
result = AlarmClient.Database.ServerName;
AlarmClient.Database.ServerName = SrvName;
```

Remarks

For more information, see Show historical alarms and/or events.

Database.UserID property

The Database.UserID property is a read-write string property that gets or sets the name of user authorized to access the Alarm Database.

Syntax

```
result = AlarmClient.Database.UserID;
AlarmClient.Database.UserID = UserName;
```

Remarks

For more information, see Show historical alarms and/or events.

DisableFileBrowsing property

Specify **True** to get/set the option to access the file browser when importing or exporting files.

Syntax

```
result = AlarmClient.DisableFileBrowsing;
AlarmClient.DisableFileBrowsing = boolean_value;
```

Example

```
AlarmClient.DisableFileBrowsing = true;
```

Remarks

Default is **True**.

Domain property

The Domain property is a read-write string property that gets or sets the domain name of the user to connect to the Alarm Database.

Syntax

```
result = AlarmClient.Domain;
AlarmClient.Domain = DomName;
```

Remarks

For more information, see Show historical alarms and/or events.

Enabled property

The Enabled property is a read-write Boolean property that gets or sets the enablement of Alarm Control. When the Alarm Control is disabled, alarm records are still updated, but the operator cannot interact with the control.

The operator can still use scripting to interact with the control.

Syntax

```
result = AlarmClient.Enabled;
AlarmClient.Enabled = EnableFlag;
```

EventColor.BackGround property

The EventColor.BackGround property is a read-write color property that gets or sets the background color of event records.

Syntax

```
Color = AlarmClient.EventColor.BackGround;
AlarmClient.EventColor.BackGround = Color;
```

Parameters

Color

Color of background.

Example

```
AlarmClient1.EventColor.BackGround = Color.Blue;
```

Remarks

Color is a .NET Framework data type. You can use various Color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName().

For a list of the .NET color names and the hexadecimal codes, see .NET colors.

For more information on the color methods, see the online Microsoft documentation for .NET Framework Development.

EventColor.ForeGround property

The EventColor.ForeGround property is a read-write color property that gets or sets the text color of event records.

Syntax

```
Color = AlarmClient.EventColor.ForeGround;
AlarmClient.EventColor.ForeGround = Color;
```

Parameters

Color

Color of text.

Example

```
AlarmClient1.EventColor.ForeGround = Color.Blue;
```

Remarks

Color is a .NET Framework data type. You can use various Color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName().

For a list of the .NET color names and the hexadecimal codes, see .NET colors.

For more information on the color methods, see the online Microsoft documentation for .NET Framework Development.

Favorite property

The Favorite property is a read-write string property that gets or sets the name of the current query filter

favorite.

Syntax

```
QueryFilterName = AlarmClient.Favorite;
AlarmClient.Favorite = QueryFilterName;
```

Parameters

QueryFilterName

The name of a query filter favorite.

Example

The following example sets the current Alarm Control grid to the Query Filter Favorite with the name "All Hi Priority Alarms".

```
AlarmClient1.Favorite = "All Hi Priority Alarms";
```

Remarks

You can also use this property to reset the currently used query filter to its default with the following script:

```
AlarmClient.Favorite = "Default";
```

The Favorite property executes as a query to retain previous release default behavior. If you want to execute the query/filter as a filter only, then use the RunQuery() method.

FlashUnAckAlarms property

The FlashUnAckAlarms property is a read-write Boolean property that gets or sets the flashing of unacknowledged alarm records.

Syntax

```
result = AlarmClient.FlashUnAckAlarms;
AlarmClient.FlashUnAckAlarms = FlashUnAckRecs;
```

Remarks

For more information, see Set unacknowledged alarms to flash.

GridColor property

The GridColor property is a read-write color property that gets or sets the color of the grid lines.

Syntax

```
Color = AlarmClient.GridColor;
AlarmClient.GridColor = Color;
```

Parameters

Color

Color of the grid lines.

Example

```
AlarmClient1.GridColor = Color.Black;
```

Remarks

For more information, see Setting Heading, Grid, and Window Color.

Color is a .NET Framework data type. You can use various Color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName().

For a list of the .NET color names and the hexadecimal codes, see .NET colors.

For more information on the color methods, see the online Microsoft documentation for .NET Framework Development.

HeadingColor.BackGround property

The HeadingColor.BackGround property is a read-write color property that gets or sets the background color of the heading.

Syntax

```
Color = AlarmClient.HeadingColor.BackGround;
AlarmClient.HeadingColor.BackGround = Color;
```

Parameters

Color

Color of background.

Example

```
AlarmClient1.HeadingColor.BackGround = Color.Blue;
```

Remarks

For more information, see Setting Heading, Grid, and Window Color.

Color is a .NET Framework data type. You can use various Color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName().

For a list of the .NET color names and the hexadecimal codes, see .NET colors.

For more information on the color methods, see the online Microsoft documentation for .NET Framework Development.

HeadingColor.ForeGround property

The HeadingColor.ForeGround property is a read-write color property that gets or sets the text color of the heading.

Syntax

```
Color = AlarmClient.HeadingColor.ForeGround;
AlarmClient.HeadingColor.ForeGround = Color;
```

Parameters

Color

Color of text.

Example

```
AlarmClient1.HeadingColor.ForeGround = Color.Blue;
```

Remarks

For more information, see Setting Heading, Grid, and Window ColorSetting Heading, Grid, and Window Color.

Color is a .NET Framework data type. You can use various Color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName().

For a list of the .NET color names and the hexadecimal codes, see .NET colors.

For more information on the color methods, see the online Microsoft documentation for .NET Framework Development.

Height property

The Height property is a read-write integer property that gets or sets the height of the Alarm Control in pixels.

Syntax

```
result = AlarmClient.Height;
```

```
AlarmClient.Height = Hght;
```

HiddenAlarms property

The HiddenAlarms property is a read-only integer property that gets the number of hidden alarms.

Syntax

```
Result = AlarmClient.HiddenAlarms;
```

Example

```
LogMessage("There are " + Text(AlarmClient1.HiddenAlarms,"#")+" hidden alarms.");
```

HideErrors property

The HideErrors property is a read-write Boolean property that gets or sets the Hide Errors option.

- TRUE - Run-time errors, warnings, and status messages are written to the ArchestrA Logger. No pop-ups appear
- FALSE - Run-time errors, warnings, and status messages pop-up and are also written to the ArchestrA Logger

Syntax

```
result = AlarmClient.HideErrors;
AlarmClient.HideErrors = SilentMode;
```

Remarks

For more information, see Hide errors, warnings, and status messages.

IsFrozen property

The IsFrozen property is a read-only Boolean property that can be used to identify whether the alarm client control is in a Frozen mode.

When the value is True, the control is frozen, else the control is not frozen.

Remarks

For more information, see Freeze and unfreeze the Alarm Control grid.

Example

```
Dim FR as boolean;
FR = AlarmClient1.IsFrozen;
```

MaxDatabaseRecords property

The MaxDatabaseRecords property is a read-write integer property that gets or sets the maximum records retrieved from the database that will appear in each page of the Alarm Control Client at one instance. The valid range is 1 to 32766.

Syntax

```
result = AlarmClient.MaxDatabaseRecords;
AlarmClient.MaxDatabaseRecords = MaxRecs;
```

Remarks

This property is for Historical Alarms, Historical Events, or Historical Alarms and Events modes. For more information, see Show historical alarms and/or events.

MaxTotalRetrievalCount property

The MaxTotalRetrievalCount property is a read-write integer property that gets or sets the maximum total number of records that will be retrieved from the database.

Syntax

```
result = AlarmClient.MaxTotalRetrievalCount;
AlarmClient.MaxTotalRetrievalCount = MaxRecs;
```

Remarks

This property is for Historical Alarms, Historical Events, or Historical Alarms and Events modes. If the property is set to a positive integer, in historical mode Alarm Client Control will retrieve up to MaxTotalRetrievalCount of alarm records. For example, there are 1000 alarm records in the database. If Set MaxTotalRetrievalCount = 500, then only top 500 alarm records will be returned. If Set MaxTotalRetrievalCount = 2000, then all 1000 alarm records will be returned.

NewAlarmEventMode property

The NewAlarmEventMode property is an read-write integer property that gets or sets the trigger behavior of the New Alarm event.

Syntax

```
EMode = AlarmClient.NewAlarmEventMode;
AlarmClient.NewAlarmEventMode = EMode;
```

Parameters

EMode

Event mode with following possible values:

Value	Description
0	The NewAlarm event cannot be triggered. (default).
1	The NewAlarm event is triggered only one time the first time a new alarm occurs.
2	The NewAlarm event is triggered every time a new alarm occurs.

NoRecordsMessage.Enabled property

The NoRecordsMessage.Enabled property is a read-write Boolean property that gets or sets the visibility of a custom message when no alarm records are available.

Syntax

```
result = AlarmClient.NoRecordsMessage.Enabled;
AlarmClient.NoRecordsMessage.Enabled = showMessage;
```

Example

```
AlarmClient1.NoRecordsMessage.Enabled = 1;
AlarmClient1.NoRecordsMessage.Message = "There are no alarm records available";
```

Remarks

Use this property in combination with the NoRecordsMessage.Message property.

NoRecordsMessage.Message property

The NoRecordsMessage.Message property is a read-write string property that gets or sets the custom message text when no alarm records are available and the **NoRecordsMessage.Enabled** property value is TRUE.

Syntax

```
result = AlarmClient.NoRecordsMessage.Message;
AlarmClient.NoRecordsMessage.Message = myCustomMessage;
```

Example

```
AlarmClient1.NoRecordsMessage.Enabled = 1;
AlarmClient1.NoRecordsMessage.Message = "There are no alarm records available";
```

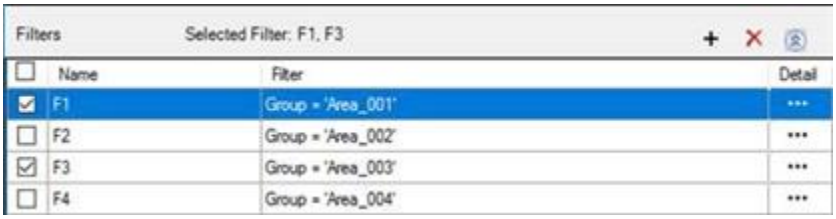
Remarks

Use this property in combination with the NoRecordsMessage.Enabled property.

QueryFilters.SelectedFilters property

The QueryFilters.SelectedFilters property is a read-only String property used to get the filter selecting status.

For example, an alarm client control with the following filters selected, the QueryFilters.SelectedFilters property will return value of “F1, F3”.



In the property, each filter will be separated using a comma ‘,’.

Example

```
Dim SF as string;
```

SF = AlarmClient1.QueryFilters.SelectedFilters;

QueryStartup property

The QueryStartup property is a read-write Boolean property that gets or sets or sets the automatic update of the Alarm Control on startup.

Syntax

```
result = AlarmClient.QueryStartup;
AlarmClient.QueryStartup = AutoQry;
```

Remarks

For more information, see Automatic query for alarms on startup.

RequiresShelveSignature property

The RequiresShelveSignature property is a read-write Boolean property to specify if a user signature is required

to shelve an alarm.

Syntax

```
To Set:
AlarmClient.RequiresShelveSignature = boolean_value;
```

Example

```
AlarmClient1.RequiresShelveSignature = true;
```

Syntax

```
To Get:
boolean_variable_name> = AlarmClient.RequiresShelveSignature;
```

Example

```
dim bIsShlvSigned as boolean;
bIsShlvSigned = AlarmClient1.RequiresShelveSignature;
```

Remarks

For more information, see Configure the Alarm Control to require a SHELVE signature.

RetainHidden property

The RetainHidden property is a read-write Boolean property that gets or sets the retention of hidden alarms or events when the alarm query or query filter to retrieve records changes at run time.

Syntax

```
result = AlarmClient.RetainHidden;
AlarmClient.RetainHidden = RetainHddn;
```

Remarks

For more information, see Retain hiding when changing the alarm query filter.

RowCount property

The RowCount property is a read-only integer property that gets the number of records shown in the Alarm Control grid.

For current alarms (and recent alarms and events), the **RowCount** property value is always the same as the **TotalRowCount** property value.

For historical alarms, if the Alarm Control retrieves more alarm records than specified by the **MaxDatabaseRecords** property value, it splits these into multiple pages.

The **RowCount** property shows how many alarm records are currently shown on the current page. The RowCount property value is the same as the **MaxDatabaseRecords** property value, with exception of the last page.

Syntax

```
Result = AlarmClient.RowCount;
```

Example

```
NRows = AlarmClient1.RowCount;
LogMessage("There are " + Text(NRows, "#") + " alarm records on the retrieved page.");
```

RowSelection property

The RowSelection property is a read-write string property that determines if row selection is allowed at run time. The following values are possible:

Value	Description
No	Operator cannot select rows.
Single	Operator can only select one row at a time.
Multiple	Operator can select one or more rows.

The default value is "Multiple".

Syntax

```
Result = AlarmClient.RowSelection;
AlarmClient.RowSelection = RwSel;
```

Example

```
AlarmClient1.RowSelection = "Multiple";
```

Remarks

For more information, see Restrict user access to rows and columns.

SaveUserQueryFilter property

Gets/Sets the option to save runtime user queries and filters. If false, then user modified queries or filters will not be saved to file. If true, user modification will be saved to file.

Syntax

```
result = AlarmClient.SaveUserQueryFilter;
AlarmClient.SaveUserQueryFilter = boolean_true;
```

Example

```
AlarmClient.SaveUserQueryFilter = true;
```

Remarks

Default is **True**.

SelectedCount property

The SelectedCount property is a read-only integer property that gets the total number of selected alarm records.

Syntax

```
Result = AlarmClient.SelectedCount;
```

Return Value

Returns the number of selected alarm records.

Example

```
NSelRows = AlarmClient1.SelectedCount;
If NSelRows > 5 Then
   LogMessage("There are more than 5 rows selected.");
Endif;
```

ShelveColor.BackGround property

ShelveColor.BackGround is an integer property to set the background color of shelved alarm records in the Alarm Control.

Syntax

```
Color= AlarmClient.ShelveColor.Background;
AlarmClient.ShelveColor.BackGround = Color
```

Parameters

Color

Background color of a shelved alarm record.

Example

```
AlarmClient1.ShelveColor.BackGround = Color.FromARGB(255,0,128);
```

Remarks

For more information, see Setting Heading, Grid, and Window Color.

Color is a .NET Framework data type. You can use various Color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName().

For a list of the .NET color names and the hexadecimal codes, see .NET colors.

For more information on the color methods, see the online Microsoft documentation for .NET Framework Development.

ShelveColor.ForeGround property

ShelveColor.ForeGround is an integer property to set the foreground or text color of shelved alarm records in the Alarm Control.

Syntax

```
Color= AlarmClient.ShelveColor.Foreground;
AlarmClient.ShelveColor.Foreground = Color
```

Parameters

Color

Foreground color of a shelved alarm record.

Example

```
AlarmClient1.ShelveColor.Foreground = Color.FromARGB(255,255,255);
```

Remarks

For more information, see Setting Heading, Grid, and Window Color.

Color is a .NET Framework data type. You can use various Color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName().

For a list of the .NET color names and the hexadecimal codes, see .NET colors.

For more information on the color methods, see the online Microsoft documentation for .NET Framework Development.

ShowContextMenu property

The ShowContextMenu property is a read-write Boolean property that gets or sets the ability to open the

shortcut menu at run time.

Syntax

```
result = AlarmClient.ShowContextMenu;
AlarmClient.ShowContextMenu = ContxtMnuAvail;
```

ShowGrid property

The ShowGrid property is a read-write Boolean property that gets or sets the appearance of grid lines.

Syntax

```
result = AlarmClient.ShowGrid;
AlarmClient.ShowGrid = showGrid;
```

ShowGroupByHeader property

The ShowGroupByHeader property is a read-write Boolean property to show or hide the column grouping label at the top of the run-time Alarm Control in the historical mode. Set the ShowGroupByHeader property to true to show the label "Drag a column header here to group by that column".

Syntax

```
result = AlarmClient.ShowGroupByHeader;
AlarmClient.ShowGroupByHeader = ShowGroupByHeader;
```

ShowHeading property

The ShowHeading property is a read-write Boolean property that gets or sets the visibility of the grid heading at run time.

Syntax

```
result = AlarmClient.ShowHeading;
AlarmClient.ShowHeading = showHeading;
```

ShowStatusBar property

The ShowStatusBar property is a read-write Boolean property that gets or sets the visibility of the status bar at run time.

Syntax

```
result = AlarmClient.ShowStatusBar;
```

```
AlarmClient.ShowStatusBar = showStatusBar;
```

SortColumn.First property

The SortColumn.First property is a read-write string property that gets or sets the first sort column.

The default value is "Time (LCT)".

Syntax

```
result = AlarmClient.SortColumn.First;
AlarmClient.SortColumn.First = sortByFirst;
```

Example

```
AlarmClient1.SortColumn.First = "Class";
```

Remarks

Use this property in connection with the SortOrder.First to determine the sorting direction.

SortColumn.Second property

The SortColumn.Second property is a read-write string property that gets or sets the second sort column.

The default value is blank.

Syntax

```
result = AlarmClient.SortColumn.Second;
AlarmClient.SortColumn.Second = sortBySecond;
```

Example

```
AlarmClient1.SortColumn.Second = "Type";
```

Remarks

Use this property in connection with the SortOrder.Second to determine the sorting direction.

SortColumn.Third property

The SortColumn.Third property is a read-write string property that gets or sets the third sort column.

The default value is blank.

Syntax

```
result = AlarmClient.SortColumn.Third;
AlarmClient.SortColumn.Third = sortByThird;
```

Example

```
AlarmClient1.SortColumn.Third = "State";
```

Remarks

Use this property in connection with the SortOrder.Third to determine the sorting direction.

SortColumn.Fourth property

The SortColumn.Fourth property is a read-write string property that gets or sets the fourth sort column.

The default value is blank.

Syntax

```
result = AlarmClient.SortColumn.Fourth;
AlarmClient.SortColumn.Fourth = sortByFourth;
```

Example

```
AlarmClient1.SortColumn.Fourth = "Priority";
```

Remarks

Use this property in connection with the SortOrder.Fourth to determine the sorting direction.

SortOrder.First property

The SortOrder.First property is a read-write Boolean property that gets or sets the sorting direction of the first sort column. The following values are possible:

Value	Description
FALSE	Ascending sorting direction
TRUE	Descending sorting direction

The default value is FALSE (Ascending).

Syntax

```
result = AlarmClient.SortOrder.First;
AlarmClient.SortOrder.First = sortDirFirst;
```

Remarks

Use this property in connection with the SortColumn.First to determine which column is sorted.

SortOrder.Second property

The SortOrder.Second property is a read-write Boolean property that gets or sets the sorting direction of the second sort column. The following values are possible:

Value	Description
FALSE	Ascending sorting direction
TRUE	Descending sorting direction

The default value is FALSE (Ascending).

Syntax

```
result = AlarmClient.SortOrder.Second;
AlarmClient.SortOrder.Second = sortDirSecond;
```

Remarks

Use this property in connection with the SortColumn.Second to determine which column is sorted.

SortOrder.Third property

The SortOrder.Third property is a read-write Boolean property that gets or sets the sorting direction of the third sort column. The following values are possible:

Value	Description
FALSE	Ascending sorting direction
TRUE	Descending sorting direction

The default value is FALSE (Ascending).

Syntax

```
result = AlarmClient.SortOrder.Third;
AlarmClient.SortOrder.Third = sortDirThird;
```

Remarks

Use this property in connection with the SortColumn.Third to determine which column is sorted.

SortOrder.Fourth property

The SortOrder.Fourth property is a read-write Boolean property that gets or sets the sorting direction of the fourth sort column. The following values are possible:

Value	Description
FALSE	Ascending sorting direction
TRUE	Descending sorting direction

The default value is FALSE (Ascending).

Syntax

```
result = AlarmClient.SortOrder.Fourth;
AlarmClient.SortOrder.Fourth = sortDirFourth;
```

Remarks

Use this property in connection with the SortColumn.Fourth to determine which column is sorted.

Time.Format property

The Time.Format property is a read-write string property that gets or sets the date and time formats of the alarm records in the Alarm Control.

You can either use the .NET time format or the default format. Set the Time.Type property to determine which time format type to use.

Syntax

```
result = AlarmClient.Time.Format;
AlarmClient.Time.Format = TmFormat;
```

Example

This example shows the time format in French format (day/month/year) using the .NET datetime type.

```
AlarmClient1.Time.Type = 1;
AlarmClient1.Time.Format = "dd/MM/yyyy";
```

Remarks

For more information about the .NET time format, see Set a .NET DateTime format.

For more information about the default time format, see Set the time format.

Time.Type property

The Time.Type property is a read-write Boolean property that gets or sets the time format type of the alarm records. The following values are possible:

Value	Description
FALSE	Default time format
TRUE	.NET time format (default)

Syntax

```
result = AlarmClient.Time.Type;
AlarmClient.Time.Type = TmType;
```

Example

This example shows the time format in German format (day.month.year) using the default datetime type.

```
AlarmClient1.Time.Type = 0;
AlarmClient1.Time.Format = "%d.%m.%Y %H:%M:%S";
```

Remarks

For more information about the .NET time format, see Set a .NET DateTime format.

For more information about the default time format, see Set the time format.

TimeSelector property

The TimeSelector property gets the Time Range Picker object used in the Alarm Control. You can use it in scripting to shorten the code using its properties and methods.

For the individual properties and methods, see the following properties, or the methods starting at TimeSelector.SetStartAndEndTimes() method.

Example 1

```
dim TRP as object;
TRP = AlarmClient1.TimeSelector;
Timeselect = TRP;
StartDate = TRP.StartDate;
EndDate = TRP.EndDate;
```

```
duration = TRP.TimeDuration;
```

Example 2

```
dim TRP as object;
TRP = AlarmClient1.TimeSelector;
TRP.SetStartAndEndTimes(StartDate, EndDate, Duration);
```

TimeSelector.DurationMS property

The TimeSelector.DurationMS property is a read-write integer property that gets the time duration measured in milliseconds.

The start time of the Alarm control (TimeSelector.StartDate) is calculated as the end time (TimeSelector.EndDate) minus the new time duration (TimeSelector.DurationMS).

When you set the value of the TimeSelector.DurationMS property, the TimeSelector.TimeDuration property is set to 0.

The default value is 3600000.

Syntax

```
result = AlarmClient.TimeSelector.DurationMS;
AlarmClient.TimeSelector.DurationMS = Value;
```

Example

```
AlarmClient1.TimeSelector.DurationMS = 1800000;
// The Alarm Control now retrieves alarms from the last 30 minutes.
```

TimeSelector.EndDate property

The TimeSelector.EndDate property is a read-only string property that gets the end date and time of the Alarm Control.

The default value is the time the Alarm Control is placed on the canvas. If the **Update to Current Time** option is enabled, the TimeSelector.EndDate property is updated with the current time.

Note: To set the end date and time of the Alarm Control, use the TimeSelector.SetStartAndEndTimes() method.

Syntax

```
result = AlarmClient.TimeSelector.EndDate;
```

Example

```
LogMessage(AlarmClient1.TimeSelector.EndDate);
```

TimeSelector.StartDate property

The TimeSelector.StartDate property is a read-only string property that gets the start date and time of the Alarm Control.

The default value is the time the Alarm Control is placed on the canvas. If the **Update to Current Time** option is enabled, the TimeSelector.StartDate property is updated as current time minus duration.

Note: To set the start date and time of the Alarm Control, use the TimeSelector.SetStartAndEndTimes() method.

Syntax

```
result = AlarmClient.TimeSelector.StartDate;
```

Example

```
LogMessage(AlarmClient1.TimeSelector.StartDate);
```

TimeSelector.TimeDuration property

The TimeSelector.TimeDuration property is a read-write integer property that gets or sets the time duration. The start time of the Alarm control (TimeSelector.StartDate) is calculated as the end time (TimeSelector.EndDate) minus the new time duration.

The TimeSelector.TimeDuration can have one of the following values:

Value	Description
0	Custom
1	The last minute.
2	The last five minutes.
3	The last ten minutes.
4	The last 15 minutes.
5	The last 30 minutes.
6	The last hour.
7	The last two hours.
8	The last four hours.
9	The last eight hours.
10	The last 12 hours.
11	The last 24 hours.

Value	Description
12	The last two days.
13	The last week.
14	The last two weeks.
15	The last month.
16	The last three months.
17	One minute.
18	Five minutes.
19	Ten minutes.
20	15 minutes.
21	30 minutes.
22	One hour.
23	Two hours.
24	Four hours.
25	Eight hours.
26	12 hours.
27	24 hours.
28	Two days.
29	One week.
30	Two weeks.
31	One month.
32	Three months.
33	Yesterday: 0:00:00 of the previous day to 0:00:00 of the current day.
34	Current day: 0:00:00 of the current day to the current time.
35	Previous hour: The start of the previous hour to the start of the current hour.

Value	Description
36	Current hour: The start of the current hour to the current time.

The default value is 6 (Last Hour).

Syntax

```
result = AlarmClient.TimeSelector.TimeDuration;
AlarmClient.TimeSelector.TimeDuration = Value;
```

Example

```
AlarmClient1.TimeSelector.TimeDuration = 5;
// The Alarm Control now retrieves alarms from the last 30 minutes.
```

Remarks

For more information, see Show historical alarms and/or events.

TimeZone.TimeZone property

The TimeZone.TimeZone property is a read-write string property that gets or sets the time zone of the Alarm Control.

The default value depends on the current setting of the operating system.

If you want to show time stamps using the local time of the computer, set the TimeZone.TimeZone property to an empty string.

Syntax

```
result = AlarmClient.TimeZone.TimeZone;
AlarmClient.TimeZone.TimeZone = TimeZone;
```

Example

```
AlarmClient1.TimeZone.TimeZone = "(GMT-09:00) Alaska";
```

Remarks

For more information, see Set time zone and format.

TotalRowCount property

The TotalRowCount property is a read-only integer property that gets the total number of alarm records in the

Alarm Control.

For current alarms (and recent alarms and events), the **RowCount** property value is always the same as the **TotalRowCount** property value.

For historical alarms, if the Alarm Control retrieves more alarm records than specified by the **MaxDatabaseRecords** property value, it splits these into multiple pages.

The **RowCount** property value shows how many alarm records are currently shown on the current page, whereas the **TotalRowCount** property value shows how many alarm records are retrieved from the alarm database.

Syntax

```
Result = AlarmClient.TotalRowCount;
```

Return Value

Returns the end date and time of the Alarm Control in historical mode.

Example

```
NTRows = AlarmClient1.TotalRowCount;
If (NTRows > 1000) then
   LogMessage("More than 1000 records are currently in the Alarm Control");
Endif;
```

UnAckAlarms property

The UnAckAlarms property is a read-only integer property that gets the number of unacknowledged alarm records in the Alarm Control.

Syntax

```
Result = AlarmClient.UnackAlarms;
```

Return Value

Returns the number of unacknowledged alarm records in the Alarm Control.

Example

```
NUnack = AlarmClient1.UnAckAlarms;
If NUnack > 10 Then
   LogMessage("There are more than 10 unacknowledged alarms in the grid!");
Endif;
```

UpdateToCurrentTime property

The UpdateToCurrentTime property is a read-write Boolean property that gets or sets the **Update to Current Time** option.

If you set this property to TRUE, the Alarm Control end time is set to the current time and the start time is calculated as the difference of end time and duration. Whenever you refresh the Alarm Control, the end time is set as current time.

If you set this property to FALSE, the Alarm Control uses the end time, duration, and start time as defined by the Time Range Picker control.

The default value is TRUE.

Syntax

```
result = AlarmClient.UpdateToCurrentTime;
AlarmClient.UpdateToCurrentTime = UpdToCurrTime;
```

Example

```
AlarmClient1.UpdateToCurrentTime = 1;
AlarmClient1.Requery();
```

Remarks

For more information, see Show historical alarms and/or events.

UserQueryFilterFilePath property

Gets/Sets the file path of the runtime user queries and filters. The file path excludes the file name.

Syntax

```
result = AlarmClient.UserQueryFilterFilePath;
AlarmClient.UserQueryFilterFilePath = myUserQueryFilterFilePath;
```

Example

```
AlarmClient1.UserQueryFilterFilePath = "C:\Users\Public\AppData\Wonderware";
```

Remarks

```
Default is %Userprofile%\Wonderware;
```

Visible property

The Visible property is a read-write Boolean property that gets or sets the visibility of the Alarm Control.

Syntax

```
result = AlarmClient.Visible;
AlarmClient.Visible = Boolean;
```

Width property

The Width property is a read-write integer property that gets or sets the width of the Alarm Control in pixels.

Syntax

```
result = AlarmClient.Width;
AlarmClient.Width = Wdth;
```

WindowColor property

The WindowColor property is a read-write color property that gets or sets the color of the Alarm Control background.

Syntax

```
Color = AlarmClient.WindowColor;
AlarmClient.WindowColor = Color;
```

Parameters

Color

Color of background.

Example

```
AlarmClient1.WindowColor = Color.FromARGB(240,200,198);
```

Remarks

For more information, see Setting Heading, Grid, and Window Color.

Color is a .NET Framework data type. You can use various Color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName().

For a list of the .NET color names and the hexadecimal codes, see .NET colors.

For more information on the color methods, see the online Microsoft documentation for .NET Framework Development.

X property

The X property is a read-write integer property that gets or sets the horizontal position of the Alarm Control in relation to the left edge of the InTouch window in which it appears.

Syntax

```
result = AlarmClient.X;
AlarmClient.X = LeftPos;
```

Y property

The Y property is a read-write integer property that gets or sets the vertical position of the Alarm Control in relation to the top edge of the InTouch window in which it appears.

Syntax

```
result = AlarmClient.Y;
AlarmClient.Y = TopPos;
```

Alarm Control methods

This section describes the scripting methods available for the Alarm Control.

AboutBox() method

The AboutBox method shows the **About** dialog box of the Alarm Control.

Syntax

```
AlarmClient.AboutBox();
```

Ack.All() method

The Ack.All method acknowledges all alarms in the Alarm Control, including those not shown.

Syntax

```
AlarmClient.Ack.All(AckComment);
```

Parameters

AckComment

A string indicating the alarm acknowledgement comment.

Example

```
AlarmClient1.Ack.All("Alarm is acknowledged");
```

Ack.Group() method

The Ack.Group method acknowledges all alarms for a given alarm source and group.

The alarm source and group names are case-insensitive.

Syntax

```
AlarmClient.Ack.Group(AlarmSource, Group, AckComment);
```

Parameters

AlarmSource

The name of the provider and optionally node providing alarms including backslash. For example:

```
\\node1\galaxy
\intouch
```

Group

The name of the alarm group. For example, $system.

AckComment

A string indicating the alarm acknowledgement comment.

Example

```
AlarmClient1.Ack.Group("\\machine1\galaxy", "Area_001", "All alarms in Area_001
acknowledged");
```

Ack.Priority() method

The Ack.Priority method acknowledges all alarms for a given alarm source, group, and priority range.

The alarm source and group names are case-insensitive.

Syntax

```
AlarmClient.Ack.Priority(AlarmSource, Group, FromPriority, ToPriority, AckComment);
```

Parameters

AlarmSource

The name of the provider and optionally node providing alarms including backslash. For example:

```
\\node1\galaxy
\intouch
```

Group

The name of the alarm group. For example, $system.

FromPriority

Starting priority of alarms. For example, 100.

ToPriority

End priority of alarms. For example, 900.

AckComment

A string indicating the alarm acknowledgement comment.

Example

```
GrpName = "ValveGroup";
AlarmClient1.Ack.Priority("\intouch", GrpName, 250, 500, "All local InTouch alarms in the
ValveGroup alarm group with priorities from 250 to 500 are now acknowledged.");
```

Ack.Selected() method

The Ack.Selected method acknowledges all selected alarms.

Syntax

```
AlarmClient.Ack.Selected(AckComment);
```

Parameters

AckComment

A string indicating the alarm acknowledgement comment.

Example

```
AlarmClient1.Ack.Selected("This selected alarm is acknowledged");
```

Ack.SelectedGroup() method

The Ack.SelectedGroup method acknowledges all alarms that have the same alarm sources and groups as one or more selected alarms.

Syntax

```
AlarmClient.Ack.SelectedGroup(AckComment);
```

Parameters

AckComment

A string indicating the alarm acknowledgement comment.

Example

```
AlarmClient1.Ack.SelectedGroup("Alarm acknowledged");
```

Ack.SelectedPriority () method

The Ack.SelectedPriority method acknowledges all alarms that have the same alarm sources, groups, and within the priority ranges as one or more selected alarms.

Syntax

```
AlarmClient.Ack.SelectedPriority(AckComment);
```

Parameters

AckComment

A string indicating the alarm acknowledgement comment.

Example

```
AlarmClient1.Ack.SelectedPriority("Alarm acknowledged");
```

Ack.SelectedTag() method

The Ack.SelectedTag method acknowledges all alarms that have the same alarm sources, groups, tags, and within the priority ranges as one or more selected alarms.

Syntax

```
AlarmClient.Ack.SelectedTag(AckComment);
```

Parameters

AckComment

A string indicating the alarm acknowledgement comment.

Example

```
AlarmClient1.Ack.SelectedTag("Alarm acknowledged");
```

Ack.Tag() method

The Ack.Tag method acknowledges all alarms for a given alarm source, group, tag name, and priority range.

The alarm source, group names, and tag names are case-insensitive.

Syntax

```
AlarmClient.Ack.Tag(AlarmSource, Group, Tag, FromPriority, ToPriority, AckComment);
```

Parameters

AlarmSource

The name of the provider and optionally node providing alarms including backslash. For example:

```
   \\node1\galaxy
   \intouch
```

Group

The name of the alarm group. For example, $system.

Tag

The name of the alarm tag. For example, ValveTag1.

FromPriority

Starting priority of alarms. For example, 100.

ToPriority

End priority of alarms. For example, 900.

AckComment

A string indicating the alarm acknowledgement comment.

Example

```
AckComment = "All alarm client records of the attribute Valve17 in the group (area)
Vessel_25B of the galaxy on machine25 with priorities from 1 to 99 are now acknowledged.";
AlarmClient1.Ack.Tag("\\machine25\galaxy", "Vessel_25B", "Valve17", 1, 99, AckComment);
```

Ack.Visible() method

The Ack.Visible method acknowledges all alarms currently visible in the Alarm Control.

Syntax

```
AlarmClient.Ack.Visible(AckComment);
```

Parameters

AckComment

A string indicating the alarm acknowledgement comment.

Example

```
AlarmClient1.Ack.Visible("Alarm acknowledged");
```

Connect() method

The Connect method connects the Alarm Control to the Alarm Database.

Syntax

```
AlarmClient.Connect();
```

Disconnect() method

The Disconnect method disconnects the Alarm Control from the Alarm Database.

Syntax

```
AlarmClient.Disconnect();
```

Dismiss.All() method

The Dismiss.All method dismisses all alarms in LATCHED alarm state.

Syntax

```
AlarmClient.Dismiss.All(Comments);
```

Parameters

Comments
A string indicating the alarm dismiss comment.

Example:

```
AlarmClient1.Dismiss.All(“all latched alarms are now dismissed”);
```

Dismiss.Group() method

The Dismiss.Group method dismisses all currently LATCHED alarms for a given alarm source and group.

Syntax

```
AlarmClient.Dismiss.Group(ProviderName,GroupName,Comments);
```

Parameters

ProviderName

Node and provider name combination that specifies the origin of alarm monitoring. For example:

```
   \\TankServer1\Galaxy
   \InTouch
```

GroupName

Alarm group or area name whose alarms are monitored by the Alarm Control. For example:

```
   $System
```

Comments

A string indicating the alarm dismiss comment.

Examples

```
ProviderName = \\TankServer1\Galaxy
GroupName = Area_001
Comments = All LATCHED alarms in Area_001 are dismissed
AlarmClient1.Dismiss.Group("\\TankServer1\Galaxy","Area_001", “All LATCHED alarms in
Area_001 are dismissed”);
```

Dismiss.Priority() method

The Dismiss.Priority method dismisses currently latched alarms within a specified alarm priority range that belong to the same specified provider and alarm group.

Syntax

```
AlarmClient1.Dismiss.Priority(ProviderName,GroupName, FromPriority, ToPriority, Comments);
```

Parameters

ProviderName

Node and provider name combination that specifies the origin of alarm monitoring. For example:

```
   \\TankServer1\Galaxy
   \intouch
```

GroupName

Alarm group or area name whose alarms are monitored by the Alarm Control. For example:

```
   $system
```

FromPriority

Starting priority of alarms. For example, 100

ToPriority

End priority of alarms. For example, 900.

Comments

A string indicating the alarm dismiss comment.

Example

```
ProviderName = \\TankServer1\Galaxy
GroupName = ValveGroup
FromPriority = 250
ToPriority = 500
Comments = alarms are now dismissed
AlarmClient1.Dismiss.Priority("\Galaxy"," ValveGroup",250,500," All local InTouch LATCHED
alarms in the ValveGroup alarm group with priorities from 250 to 500 are now dismissed");
```

Dismiss.Selected() method

The Dismiss.Selected method dismisses currently latched alarms selected by the user from the Alarm Control.

Syntax

```
AlarmClient1.Dismiss.Selected(Comments);
```

Parameters

Comments

A string indicating the alarm dismiss comment.

Example:

```
Comment = latched alarms are now dismissed
AlarmClient1.Dismiss.Selected(“latched alarms are now dismissed”);
```

Dismiss.SelectedGroup() method

The Dismiss.SelectedGroup method dismisses all latched alarms that belong to the same provider and alarm groups as the alarms selected by the user from the Alarm Control.

Syntax

```
AlarmClient1.Dismiss.SelectedGroup(Comments);
```

Parameters

Comments

A string indicating the alarm dismiss comment.

Example:

```
Comment = latched alarms are now dismissed
AlarmClient1.Dismiss.SelectedGroup("latched alarms are now dismissed");
```

Dismiss.SelectedPriority() method

The Dismiss.SelectedPriority method dismisses all latched alarms that have the same alarm sources, groups, and within the priority ranges as one or more selected alarms.

Syntax

```
AlarmClient1.Dismiss.SelectedPriority(Comments);
```

Parameters

Comments

A string indicating the alarm dismiss comment.

Example:

```
Comment = latched alarms are now dismissed
AlarmClient1.Dismiss.SelectedPriority("latched alarms are now dismissed");
```

Dismiss.SelectedTag() method

The Dismiss.SelectedTag method dismisses all latched alarms that have the same alarm sources, groups, tags, and within the priority ranges as one or more selected alarms.

Syntax

AlarmClient1.Dismiss.SelectedTag(Comments);

Parameters

Comments

A string indicating the alarm dismiss comment.

Example:

```
Comment = latched alarms are now dismissed
AlarmClient1.Dismiss.SelectedTag("latched alarms are now dismissed");
```

Dismiss.Tag() method

The Dismiss.Tag method dismisses all latched alarms for a given alarm source, group, tag name, and priority range.

The alarm source, group names, and tag names are case-insensitive.

Syntax

AlarmClient1.Dismiss.Tag(ProviderName,GroupName,TagName,FromPriority,ToPriority,Comments);

Parameters

ProviderName

The name of the provider and optionally node providing alarms including backslash. For example: \\node1\galaxy

GroupName

The name of the alarm group. For example, $system.

TagName

The name of the alarm tag. For example, ValveTag1.

FromPriority

Starting priority of alarms. For example, 100

ToPriority

End priority of alarms. For example, 900.

Comments

A string indicating the alarm dismiss comment.

Example:

```
ProviderName = \\machine25\galaxy
GroupName = Vessel_25B
TagName = Valve17
FromPriority = 1
ToPriority = 99
Comments = All alarm client records of the attribute Valve17 in the group (area) Vessel_25B
of the galaxy on machine25 with priorities from 1 to 99 are now acknowledged.
AlarmClient1.Dismiss.Tag("\\machine25\galaxy", "Vessel_25B", "Valve17", 1, 99, “All alarm
client records of the attribute Valve17 in the group (area) Vessel_25B of the galaxy on
machine25 with priorities from 1 to 99 are now acknowledged");
```

Dismiss.Visible() method

The Dismiss.Visible method dismisses only those latched alarms that are currently visible from the Alarm Control.

Syntax

```
AlarmClient1.Dismiss.Visible(Comments);
```

Parameters

Comments

A string indicating the alarm dismiss comment.

Example:

```
Comment = latched alarms are now dismissed
AlarmClient1.Dismiss.Visible(“latched alarms are now dismissed”);
```

Favorites.Export() method

The Favorites.Export method exports the list of query and filter favorites list to an XML file.

Syntax

```
AlarmClient.Favorites.Export(FilePath, FileName);
```

Parameters

FilePath

Name of the path to export the query and filter favorites file.

FileName

Name of the query and filter favorites file to export.

Example

```
AlarmClient1.Favorites.Export("c:\", "MyFavorites.xml");
```

Favorites.Import() method

The Favorites.Import method imports the list of query and filter favorites list from an XML file. You can either overwrite the existing query and filter favorites with the new favorites, or append them.

Syntax

```
AlarmClient.Favorites.Import(FilePath, FileName, OverwriteAppend);
```

Parameters

FilePath

Name of the path to the query and filter favorites file to import.

FileName

Name of the query and filter favorites file to import.

OverwriteAppend

String determining if the import of the query filter favorites overwrites existing favorites, or appends to existing favorites. Set to one of the following:

- Overwrite to overwrite existing query filter favorites
- Append to append to existing query filter favorites. If a query filter with the same name already exists, it is not overwritten by the query filter in the file

Example

```
AlarmClient1.Favorites.Import("c:\MyFavs\", "Favs.xml", "Overwrite");
```

FreezeDisplay() method

The FreezeDisplay method freezes or unfreezes the Alarm Control. The following values are possible:

Value	Description
TRUE	Freezes the Alarm Control.
FALSE	Unfreezes the Alarm Control.

Syntax

```
AlarmClient.FreezeDisplay(FreezeFlag);
```

Parameters

FreezeFlag

Boolean value or expression (TRUE = freeze control, FALSE = unfreeze control)

Example

```
AlarmClient1.FreezeDisplay($hour > 17 OR $hour<9 );
LogMessage("The Alarm Control is frozen between 6 PM and 8 AM.");
```

GetItem() method

The GetItem method returns the data at the given row and column. The row is given as a zero-based index. You need to specify 0 to retrieve data from the 1st row. The column name can either be the original column name, or the displayed column name.

Syntax

```
Result = AlarmClient.GetItem(RowNumber, ColumnName);
```

Parameters

RowNumber

An integer row number for the alarm record containing the value you want to fetch.

ColumnName

Name of the column.

Return Value

Returns the data at the given row and column as a string value.

Example

```
Data1 = AlarmClient1.GetItem(5, "Current Value");
LogMessage("The current value of the 6th alarm record is " + Data1);
```

Remarks

To get alarm record data from the currently selected row in a given column name, use the **GetSelectedItem** method.

GetLastError() method

The GetLastError method returns the last error message. This is useful if the Hide Errors option is selected.

Syntax

```
ErrMsg = AlarmClient.GetLastError();
```

Return Value

Returns the last error message.

Example

```
ErrMsg = AlarmClient1.GetLastError();
ComboBox1.AddItem(ErrMsg);
```

GetSelectedItem() method

The GetSelectedItem method returns the data at the currently selected row and specified column. The column name can either be the original column name, or the displayed column name.

Syntax

```
Result = AlarmClient.GetSelectedItem(ColumnName);
```

Parameters

ColumnName

Name of the column.

Return Value

Returns the data in the currently selected row and specified column as a string value.

Example

```
Data2 = AlarmClient1.GetSelectedItem ("State");
LogMessage("The current state of the selected alarm record is " + Data2);
```

Remarks

To get alarm record data from a given column name and row index, use the **GetItem** method.

Hide.All() method

The Hide.All method hides all current alarms in the Alarm Control, including future alarms.

Syntax

```
AlarmClient.Hide.All();
```

Hide.Group() method

The Hide.Group method hides all alarms for a given alarm source and group.

The alarm source and group names are case-insensitive.

Syntax

```
AlarmClient.Hide.Group(AlarmSource, Group);
```

Parameters

AlarmSource

The name of the provider and optionally node providing alarms including backslash. For example:

```
   \\node1\galaxy
   \intouch
```

Group

The name of the alarm group. For example, $system.

Example

```
AlarmClient1.Hide.Group("\\machine1\galaxy", "Area_001");
LogMessage("All alarms in Area_001 hidden.");
```

Hide.Priority() method

The Hide.Priority method hides all alarms for a given alarm source, group, and priority range.

The alarm source and group names are case-insensitive.

Syntax

```
AlarmClient.Hide.Priority(AlarmSource, Group, FromPriority, ToPriority);
```

Parameters

AlarmSource

The name of the provider and optionally node providing alarms including backslash. For example:

```
   \\node1\galaxy
   \intouch
```

Group

The name of the alarm group. For example, $system.

FromPriority

Starting priority of alarms. For example, 100.

ToPriority

End priority of alarms. For example, 900.

Example

```
GrpName = "ValveGroup";
AlarmClient1.Hide.Priority("\intouch", GrpName, 250, 500);
```

```
LogMessage("All local InTouch alarms in the ValveGroup alarm group with priorities from 250
to 500 are now hidden.");
```

Hide.Selected() method

The Hide.Selected method hides all selected alarms.

Syntax

```
AlarmClient.Hide.Selected();
```

Hide.SelectedGroup() method

The Hide.SelectedGroup method hides all alarms that have the same alarm sources and groups as one or more selected alarms.

Syntax

```
AlarmClient.Hide.SelectedGroup();
```

Hide.SelectedPriority() method

The Hide.SelectedPriority method hides all alarms that have the same alarm sources, groups, and within the priority ranges as one or more selected alarms.

Syntax

```
AlarmClient.Hide.SelectedPriority();
```

Hide.SelectedTag() method

The Hide.SelectedTag method hides all alarms that have the same alarm sources, groups, tag names, and within the priority ranges as one or more selected alarms.

Syntax

```
AlarmClient.Hide.SelectedTag();
```

Remarks

None

Hide.Tag() method

The Hide.Tag method hides all alarms for a given alarm source, group, tag name, and priority range.

The alarm source, group name, and tag names are case-insensitive.

Syntax

```
AlarmClient.Hide.Tag(AlarmSource, Group, Tag, FromPriority, ToPriority);
```

Parameters

AlarmSource

The name of the provider and optionally node providing alarms including backslash. For example:

```
   \\node1\galaxy
   \intouch
```

Group

The name of the alarm group. For example, $system.

Tag

The name of the alarm tag. For example, ValveTag1.

FromPriority

Starting priority of alarms. For example, 100.

ToPriority

End priority of alarms. For example, 900.

Example

```
AlarmClient1.Hide.Tag("\\machine25\galaxy", "Vessel_25B", "Valve17", 1, 99);
LogMessage("All alarm client records of the attribute Valve17 in the group (area)
Vessel_25B of the galaxy on machine25 with priorities from 1 to 99 are now hidden.");
```

Hide.Visible() method

The Hide.Visible method hides all alarms currently shown in the Alarm Control.

Syntax

```
AlarmClient.Hide.Visible();
```

LoadQueryFilterFile

Load from an xml file with query and filter favorites entries with different override options. Return false for any error.

Syntax

```
AlarmClient.LoadQueryFilterFile(FilePath, OverrideOption);
```

Parameters

FilePath

Name of the path to the query filter favorites file to import.

OverrideOption

- **Append:** Add to the query and filter favorites. If a duplicate entry is already present, it will not append.
- **Overwrite:** Delete all current query and filter favorite entries and add the new entries from the file.
- **Replace:** Append and replace any duplicated entries with the contents from the file.

Example

```
AlarmClient. LoadQueryFilterFile("c:\users\public\file1.xml","Append")
```

MoveWindow() method

The MoveWindow method scrolls the alarm records in the control in a given direction.

Syntax

```
AlarmClient.MoveWindow(ScrollDir, Repeat);
```

Parameters

ScrollDir

String indicating the direction to scroll. This parameter is case-insensitive. See the following table.

ScrollDir	Description
LineDn	Line down. The Repeat parameter controls the number of lines to be scrolled.
LineUp	Line up. The Repeat parameter controls the number of lines to be scrolled.
PageDn	Page down. The Repeat parameter controls the number of pages to be scrolled.
PageUp	Page up. The Repeat parameter controls the number of pages to be scrolled.
Top	To the top of the control
Bottom	To the bottom of the control.
PageRt	Page to the right. The Repeat parameter controls the number of pages to be scrolled.

ScrollDir	Description
PageLf	Page to the left. The Repeat parameter controls the number of pages to be scrolled.
Right	Scrolls right. The Repeat parameter controls the number of columns to be scrolled.
Left	Scrolls left. The Repeat parameter controls the number of columns to be scrolled.
Home	Scrolls to the top row and left most column of the control.

Repeat

Number of times to repeat the scroll action.

Example

```
AlarmClient1.MoveWindow("Bottom", 0);
```

Requery() method

The Requery method refreshes the alarm records in the Alarm Control.

For current alarms and recent alarms and events, the control requeries the Alarm Manager. For historical alarms or events, the control retrieves alarm records from the Alarm Database.

Syntax

```
AlarmClient.Requery();
```

Reset() method

The Reset method resets column widths and the column order to their last known design-time settings. The Reset method also resets the current query filter to the default query.

Syntax

```
AlarmClient.Reset();
```

ResetSortCriteria() method

Reset the sorting criteria back to the design time settings and apply the sorting criteria.

Syntax

```
AlarmClient.ResetSortCriteria();
```

RunQuery() method

Execute the query or filter available on the Alarm Client Control by name. If the ApplyAsFilter option is true, then execute the query as a filter only. Note the following behaviors:

- If ApplyAsFilter is true, the parameter QueryName needs to be a valid Filter.
- If ApplyAsFilter is false, the parameter QueryName needs to be a valid Query.
- In all other scenarios, the RunQuery() will not execute.

Syntax

```
AlarmClient.RunQuery(QueryName, ApplyAsFilter);
```

Parameters

QueryName

The name of the query or filter available on the Alarm Client Control.

ApplyAsFilter

Specify **True** to execute a query as a filter using the filter criteria and retain the existing alarm subscription.

Example

```
AlarmClient.RunQuery("High Alarms", true);
```

RunQueryFromFile() method

Execute the query or filter by name defined in a file. This does not impact the current list of queries and filters. If ApplyAsFilter option is true, then execute the query as filter only. Note the following behaviors:

- If ApplyAsFilter is true, the parameter QueryName needs to be a valid Filter.
- If ApplyAsFilter is false, the parameter QueryName needs to be a valid Query.
- In all other scenarios, the RunQuery() will not execute.

Syntax

```
AlarmClient.RunQueryFromFile(FilePath, QueryName, ApplyAsFilter);
```

Parameters

FilePath

The file path to the query filter favorites file to execute.

QueryName

The name of the query or filter defined in the user provided file.

ApplyAsFilter

Specify **True** to execute a query as a filter using the filter criteria and retain the existing alarm subscription.

Example

```
AlarmClient.RunQueryFromFile("C:\UserQueryFilter.xml", "User High Alarms", true);
```

SelectFilters() method

The SelectFilters() script function will apply filters to the alarm records basing on the list of filter names in the parameter FilterNames, where each filter name is separate by comma. If FilterNames is empty, no filter will be applied to the alarm records. Invalid filter name will be ignored.

Syntax

```
AlarmClient.SelectFilters(FilterNames);
```

Parameters

FilterNames

The name of the filters which are stored in the filter list. It can be a string constant or a string type reference.

Example

```
AlarmClient1.SelectFilters("F1, F3, F5");
```

AlarmClient1.SelectFilters(" ");

Select.All() method

The Select.All method selects all alarms in the Alarm Control.

Syntax

```
AlarmClient.Select.All();
```

Select.Group() method

The Select.Group method selects all alarms for a given provider and group.

Syntax

```
AlarmClient.Select.Group(AlarmSource, Group);
```

Parameters

AlarmSource

The name of the provider and optionally node providing alarms including backslash. For example:

```
   \\node1\galaxy
   \intouch
```

Group

The name of the alarm group. For example, $system.

Example

```
AlarmClient1.Select.Group("\\machine1\galaxy", "Area_001");
LogMessage("All galaxy alarms of group Area_001 from machine1 are now selected.");
```

Select.Item() method

The Select.Item method selects an alarm record at a given zero-based row number.

Syntax

```
AlarmClient.Select.Item(RowNumber);
```

Parameters

RowNumber

An integer row number for the alarm record to select. The first row in the control is 0.

Example

```
AlarmClient1.Select.Item(5);
LogMessage("The alarm record in the 6th row (index 5) is now selected.");
```

Select.Priority() method

The Select.Priority method selects all alarms for a given alarm source, group, and priority range.

Syntax

```
AlarmClient.Select.Priority(AlarmSource, Group, FromPriority, ToPriority);
```

Parameters

AlarmSource

The name of the provider and optionally node providing alarms including backslash. For example:

```
\\node1\galaxy
\intouch
```

Group

The name of the alarm group. For example, $system.

FromPriority

Starting priority of alarms. For example, 100.

ToPriority

End priority of alarms. For example, 900.

Example

```
GrpName = "ValveGroup";
AlarmClient1.Select.Priority("\intouch", GrpName, 250, 500);
LogMessage("All local InTouch alarms in the ValveGroup alarm group with priorities from 250
to 500 are now selected.");
```

Select.Tag() method

The Select.Tag method selects all alarms for a given alarm source, group, tag name, and priority range.

Syntax

```
AlarmClient.Select.Tag(AlarmSource, Group, Tag, FromPriority, ToPriority);
```

Parameters

AlarmSource

The name of the provider and optionally node providing alarms including backslash. For example:

```
   \\node1\galaxy
   \intouch
```

Group

The name of the alarm group. For example, $system.

Tag

The name of the alarm tag. For example, ValveTag1.

FromPriority

Starting priority of alarms. For example, 100.

ToPriority

End priority of alarms. For example, 900.

Example

```
AlarmClient1.Select.Tag("\\machine25\galaxy", "Vessel_25B", "Valve17", 1, 99);
LogMessage("All alarm client records of the attribute Valve17 in the group (area)
Vessel_25B of the galaxy on machine25 with priorities from 1 to 99 are now selected.");
```

SetSort() method

The SetSort method sets the level of sorting according to the defined sort columns and sort orders.

Syntax

```
AlarmClient.SetSort(Level);
```

Parameters

Level

The level of sorting:

Value	Description
1	Only use the first sort column.
2	Use first and second sort columns.
3	Use first, second, and third sort columns.
4	Use first, second, third, and fourth sort columns.

Example

```
AlarmClient1.SetSort(2);
```

Remarks

Use the **Show.Sort** method to open the **Sort** dialog box instead.

SetSortCriteria() method

Set the sorting criteria by supplying a string of which columns to sort and their order. Then apply the sort criteria. Return false if any error.

Syntax

```
AlarmClient.SetSortCriteria(SortCriteriaString);
```

Parameters

SortCriteriaString

The string containing the sorting criteria to be executed.

```
The syntax of the sort criteria string is: [CriteriaName]:[1|-1];
```

where:

CriteriaName is the name of the sorting criteria

Colon is separator for sort order integer

Use 1 for Ascending order

Use -1 for Descending order

Semicolon is separator for the next sorting criteria

If ordering integer is not supplied, then default to ascending order

Example

```
Example1: Sort by time ascending order and then by Priority ascending order
AlarmClient.SetSortCriteria("TimeLCT:1; Priority:1");
Example2: Sort by time ascending order using default value and then by Priority descending
```

```
order
AlarmClient.SetSortCriteria("TimeLCT; Priority:-1");
```

```
Example3: Sort by ascending order in sequence of InAlarm, AckState, Priority, and Time.
AlarmClient.SetSortCriteria("InAlarm;AckState;Priority;TimeLCT");
```

Shelve.All() method

The Shelve.All() method shelves all active alarms shown in the Alarm Control.

Syntax

```
AlarmClient.Shelve.All(Duration=Duration;Reason="Reason");
```

Parameters

Duration

Length of time in hours that selected alarms are shelved. Duration can be specified as an integer or a floating point number greater than 0.

Example:

```
   Duration=0.5
```

Reason

Explanation up to 200 characters for shelving all active alarms shown in the Alarm Control.
An explanation can be delimited by double or single quotation marks. Quotation marks and back slashes (\) can appear within the text of an explanation. If an explanation is delimited by double quotation marks, a single quotation mark is simply a character in the string. If an explanation is delimited by single quotation marks, a double quotation mark is simply a character in the string.

Example

```
AlarmClient1.Shelve.All("Duration=0.1;Reason=’Shelve all alarms’;");
```

Remarks

For more information about shelving alarms, see Shelve alarms.

Shelve.Group() method

The Shelve.Group() method shelves all alarms belonging to a specified provider and alarm group.

Syntax

```
AlarmClient.Shelve.Group("ProviderName", "GroupName",Duration=Duration;Reason="Reason");
```

Parameters

ProviderName

Node and provider name combination that specifies the origin of alarm monitoring.

GroupName

Alarm group or area name whose alarms are monitored by the Alarm Control.

Duration

Length of time in hours that selected alarms are shelved. Duration can be specified as an integer or a floating point number greater than 0.

Example:

```
   Duration=0.5
```

Reason

Explanation up to 200 characters for shelving alarms by groups shown in the Alarm Control.
An explanation can be delimited by double or single quotation marks. Quotation marks and back slashes (\) can appear within the text of an explanation. If an explanation is delimited by double quotation marks, a single quotation mark is simply a character in the string. If an explanation is delimited by single quotation marks, a double quotation mark is simply a character in the string.

Examples

```
AlarmClient1.Shelve.Group("\\TankServer1\Galaxy","Area_001", Duration=0.1;Reason=" + """ +
"Shelved"+ """ + ";");
AlarmClient1.Shelve.Group("\Galaxy","Area_001",Duration=0.1; Reason=" + """ + "Shelved"+
""" + ";");
```

Remarks

For more information about specifying provider and group names, see Alarm queries.

Shelve.Priority() method

The Shelve.Priority() method shelves all active alarms within a specified alarm priority range that belong to the same specified provider and alarm group.

Syntax

```
AlarmClient.Shelve.Priority("ProviderName","GroupName",
FromPriority,ToPriority,Duration=Duration;Reason="Reason");
```

Parameters

ProviderName

Node and provider name combination that specifies the origin of alarm monitoring.

GroupName

Alarm group or area name whose alarms are monitored by the Alarm Control.

FromPriority

Starting point of the alarm priority range (1-999). The `FromPriority` value must be less than the `ToPriority` value.

ToPriority

Three-digit end point of the alarm priority range (2-999). The `ToPriority` value must be greater than the `FromPriority` value.

Duration

Length of time in hours that selected alarms are shelved. Duration can be specified as an integer or a floating point number greater than 0.

Example:

```
   Duration=0.5
```

Reason

Explanation up to 200 characters for shelving alarms by priority range that are shown in the Alarm Control. An explanation can be delimited by double or single quotation marks. Quotation marks and back slashes (\) can appear within the text of an explanation. If an explanation is delimited by double quotation marks, a single quotation mark is simply a character in the string. If an explanation is delimited by single quotation marks, a double quotation mark is simply a character in the string.

Example

```
AlarmClient1.Shelve.Priority("\Galaxy","Area_001",100,600, Duration=0.1;Reason=" + """ +
"Shelved"+ """ + ";");
```

Remarks

For more information about setting an alarm priority range, see Set priority ranges for alarm records.

Shelve.Selected() method

The Shelve.Selected() method shelves one or more active alarms selected by the user from the Alarm Control.

Syntax

```
AlarmClient.Shelve.Selected("Duration=Duration;Reason='Reason'");
```

Parameters

Duration

Length of time in hours that selected alarms are shelved. Duration can be specified as an integer or a floating point number greater than 0.

Example:

```
   Duration=0.5
```

Reason

Explanation up to 200 characters for shelving active alarms selected by the user from the Alarm Control.
An explanation can be delimited by double or single quotation marks. Quotation marks and back slashes (\) can appear within the text of an explanation. If an explanation is delimited by double quotation marks, a single quotation mark is simply a character in the string. If an explanation is delimited by single quotation marks, a double quotation mark is simply a character in the string.

Example

```
AlarmClient.Shelve.Selected("Duration=1.0;Reason='Nuisance alarms'");
```

Remarks

For more information about selecting alarms to shelve, see Shelve alarms.

Shelve.SelectedGroup() method

The Shelve.Selected.Group method shelves all active alarms that belong to the same alarm groups as the alarms selected by the user from the Alarm Control.

Syntax

```
AlarmClient.Shelve.SelectedGroup("Duration=Duration;Reason='Reason'");
```

Parameters

Duration

Length of time in hours that selected alarms are shelved. Duration can be specified as an integer or a floating point number greater than 0.

Example:

```
   Duration=0.5
```

Reason

Explanation up to 200 characters for shelving alarms in the same group as the alarms selected by the user from the Alarm Control.
An explanation can be delimited by double or single quotation marks. Quotation marks and back slashes (\) can appear within the text of an explanation. If an explanation is delimited by double quotation marks, a single quotation mark is simply a character in the string. If an explanation is delimited by single quotation marks, a double quotation mark is simply a character in the string.

Example

```
AlarmClient.Shelve.SelectedGroup("Duration=1.0;Reason='Low severity nuisance alarms'");
```

Remarks

For more information about selecting alarms to shelve, see Shelve alarms.

Shelve.SelectedPriority() method

The Shelve.SelectedPriority method shelves all active alarms that belong to the same alarm priority as the alarms selected by the user from the Alarm Control.

Syntax

```
AlarmClient.Shelve.SelectedPriority("Duration=Duration;Reason='Reason'");
```

Parameters

Duration

Length of time in hours that selected alarms are shelved. Duration can be specified as an integer or a floating point number greater than 0.

Example:

```
   Duration=0.5
```

Reason

Explanation up to 200 characters for shelving alarms in the same priority range as the alarms selected by the user from the Alarm Control.
An explanation can be delimited by double or single quotation marks. Quotation marks and back slashes (\) can appear within the text of an explanation. If an explanation is delimited by double quotation marks, a single quotation mark is simply a character in the string. If an explanation is delimited by single quotation marks, a double quotation mark is simply a character in the string.

Example

```
AlarmClient.Shelve.SelectedPriority("Duration=1.0;Reason='Low priority nuisance alarms'");
```

Remarks

For more information about selecting alarms to shelve, see Shelve alarms.

Shelve.SelectedSeverity() method

The Shelve.Selected.Severity method shelves all active alarms for a given alarm group, tag name, and severity as the alarms selected by the user from the Alarm Control.

Syntax

```
AlarmClient.Shelve.SelectedSeverity("Duration=Duration;Reason='Reason'");
```

Parameters

Duration

Length of time in hours that selected alarms are shelved. Duration can be specified as an integer or a floating point number greater than 0.

Example:

Duration=0.5

Reason

Explanation up to 200 characters for shelving alarms at the same severity as the alarms selected by the user from the Alarm Control.
An explanation can be delimited by double or single quotation marks. Quotation marks and back slashes (\) can appear within the text of an explanation. If an explanation is delimited by double quotation marks, a single quotation mark is simply a character in the string. If an explanation is delimited by single quotation marks, a double quotation mark is simply a character in the string.

Example

```
AlarmClient.Shelve.SelectedSeverity("Duration=1.0;Reason='Low severity nuisance alarms'");
```

Remarks

For more information about selecting alarms to shelve, see Shelve alarms.

Shelve.SelectedTag() method

The Shelve.SelectedTag method shelves all active alarms all active alarms for a given provider, alarm group, and tag name as the alarms selected by the user from the Alarm Control.

Syntax

```
AlarmClient.Shelve.SelectedTag("Duration=Duration;Reason='Reason'");
```

Parameters

Duration

Length of time in hours that selected alarms are shelved. Duration can be specified as an integer or a floating point number greater than 0.

Example:

Duration=0.5

Reason

Explanation up to 200 characters for shelving alarms from the same tags as the alarms selected by the user from the Alarm Control.
An explanation can be delimited by double or single quotation marks. Quotation marks and back slashes (\) can appear within the text of an explanation. If an explanation is delimited by double quotation marks, a single

quotation mark is simply a character in the string. If an explanation is delimited by single quotation marks, a double quotation mark is simply a character in the string.

Example

```
AlarmClient.Shelve.SelectedTag("Duration=1.0;Reason='Nuisance alarms'");
```

Remarks

For more information about selecting alarms to shelve, see Shelve alarms.

Shelve.Severity() method

The Shelve.Severity() method shelves all active alarms of a specified alarm severity that belong to the same specified provider and alarm group.

Syntax

```
AlarmClient.Shelve.Severity("ProviderName","GroupName",
Severity,Duration=Duration;Reason="Reason");
```

Parameters

ProviderName

Node and provider name combination that specifies the origin of alarm monitoring.

GroupName

Alarm group or area name whose alarms are monitored by the Alarm Control.

Severity

Single-digit (1-4) alarm severity.

Duration

Length of time in hours that selected alarms are shelved. Duration can be specified as an integer or a floating point number greater than 0.

Example:

Duration=0.5

Reason

Explanation up to 200 characters for shelving alarms at a specified severity from the Alarm Control.
An explanation can be delimited by double or single quotation marks. Quotation marks and back slashes (\) can appear within the text of an explanation. If an explanation is delimited by double quotation marks, a single quotation mark is simply a character in the string. If an explanation is delimited by single quotation marks, a double quotation mark is simply a character in the string.

Example

```
AlarmClient1.Shelve.Severity("\Galaxy","Area_001",3,
```

```
Duration=0.1;Reason=" + """ + "Shelved"+ """ + ";");
```

Remarks

For more information about selecting alarms to shelve, see Shelve alarms.

Shelve.Tag() method

The Shelve.tag() method shelves all active alarms that originate from the same tag and belong to the same provider and alarm group within the same alarm priority range.

Syntax

```
AlarmClient.Shelve.Tag("ProviderName","GroupName","Tag"
FromPriority,ToPriority,Duration=Duration;Reason="Reason");
```

Parameters

ProviderName

Node and provider name combination that specifies the origin of alarm monitoring.

GroupName

Alarm group or area name whose alarms are monitored by the Alarm Control.

Tag

Name of the tag whose active alarms have been selected to be shelved.

FromPriority

Starting point of the alarm priority range (1-998). The `FromPriority` value must be less than the `ToPriority` value.

ToPriority

End point of the alarm priority range (2-999). The `ToPriority` value must be greater than the `FromPriority` value.

Duration

Length of time in hours that selected alarms are shelved. Duration can be specified as an integer or a floating point number greater than 0.

Example:

Duration=0.5

Reason

Explanation up to 200 characters for shelving alarms from the same tags.
An explanation can be delimited by double or single quotation marks. Quotation marks and back slashes (\) can appear within the text of an explanation. If an explanation is delimited by double quotation marks, a single quotation mark is simply a character in the string. If an explanation is delimited by single quotation marks, a double quotation mark is simply a character in the string.

Example

```
AlarmClient1.Shelve.Tag("\Galaxy","Area_002",
"UserDefined_002.Analog_003",1,999,"Duration=1.5;Reason=" + """ + "Shelved"+ """ + ";");
```

Remarks

For more information about selecting alarms to shelve, see Shelve alarms.

Shelve.Visible() method

The Shelve.Visible() method shelves only those alarms that are currently visible from the Alarm Control.

Syntax

```
AlarmClient.Shelve.Visible("Duration=<Duration
Hrs>;Reason="+"<Reason>"+"Shelved"+"<Description>"+";");
```

Parameters

Duration

Length of time in hours that selected alarms are shelved. Duration can be specified as an integer or a floating point number greater than 0.

Example:

Duration=0.5

Reason

Explanation up to 200 characters for shelving all alarms that are visible from the Alarm Control.
An explanation can be delimited by double or single quotation marks. Quotation marks and back slashes (\) can appear within the text of an explanation. If an explanation is delimited by double quotation marks, a single quotation mark is simply a character in the string. If an explanation is delimited by single quotation marks, a double quotation mark is simply a character in the string.

Examples

```
AlarmClient1.Shelve.Visible("Duration=1.0;
Reason="+"""+"Shelved"+"""+";");
```

Remarks

For more information about selecting alarms to shelve, see Shelve alarms.

Show.Context() method

The Show.Context method opens the shortcut menu at run time. This method ignores the ShowContextMenu property setting and always shows the context menu.

Syntax

```
AlarmClient.Show.Context();
```

Show.Favorite() method

The Show.Favorite method opens the **Query Filters** dialog box.

Syntax

```
AlarmClient.Show.Favorite();
```

Show.Hidden() method

The Show.Hidden method opens the **Hidden Alarms** dialog box.

Syntax

```
AlarmClient.Show.Hidden();
```

Show.Sort() method

The Show.Sort method opens the **Sort** dialog box.

Syntax

```
AlarmClient.Show.Sort();
```

Show.Statistics() method

The Show.Statistics method opens the **Alarm Statistics** dialog box.

Syntax

```
AlarmClient.Show.Statistics();
```

TimeSelector.GetStartAndEndTimes() method

The TimeSelector.GetStartAndEndTimes method gets the start and end times for the query.

Syntax

```
AlarmClient.GetStartAndEndTimes(StartTime, EndTime);
```

Parameters

StartTime

String attribute, custom property, or element property to retrieve the start time.

EndTime

String attribute, custom property, or element property to retrieve the end time.

Example

```
dim SDate as string;
dim EDate as string;
AlarmClient1.TimeSelector.GetStartAndEndTimes(SDate, EDate);
StartDate = SDate;
EndDate = EDate;
```

TimeSelector.RefreshTimes() method

The TimeSelector.RefreshTimes method sets the time period for the query by updating the end time to current time and recalculates the start time based on the new end time and duration.

If you set the Boolean parameter to TRUE, the OnChange event is triggered if the time is updated.

Only use this method, if the **Update to Current Time** option is cleared or the **UpdateToCurrentTime** property is FALSE.

Note: This method does not work if the **UpdatetoCurrentTime** property value is TRUE.

Syntax

```
AlarmClient.TimeSelector.RefreshTimes(TriggerEvent);
```

Example

```
dtag = 1;
AlarmClient.TimeSelector.RefreshTimes(dtag);
```

TimeSelector.SetStartAndEndTimes() method

The TimeSelector.SetStartAndEndTimes method sets the start and end times for a query.

To customize start time, duration, and end time you must set the **UpdateToCurrentTime** property to FALSE and specify one of the following parameter combinations:

- Start time and end time. Set the Duration parameter to 0.
- Start time and duration. Set the EndTime parameter to "".
- End time and duration. Set the StartTime parameter to "".
- Start time, duration, and end time. The Alarm Control shows an error message if start time plus duration is not equal to end time.

Syntax

```
AlarmClient.SetStartAndEndTimes(StartTime, EndTime, Duration);
```

Parameters

StartTime

String value or expression indicating the start time.

EndTime

String value or expression indicating the end time.

Duration

Duration enum. For more information on possible values, see TimeSelector.TimeDuration property.

Example

```
AlarmClient1.TimeSelector.SetStartAndEndTimes("08/31/2008 15:33:43","09/01/2009
15:33:43",0);
```

Toggle.All() method

The Toggle.All method reverses the selection of all alarm records. Selected alarms are cleared, and unselected alarms are selected.

Syntax

```
AlarmClient.Toggle.All();
```

Toggle.Item() method

The Toggle.Item method reverses the selection of a given alarm record. If the given alarm record is selected, the selection is cleared; otherwise, it is selected.

Syntax

```
AlarmClient.Toggle.Item(RowNumber);
```

Parameters

RowNumber

An integer row number for the alarm record to reverse the selection. The first row in the control is 0.

Example

```
AlarmClient1.Toggle.Item(5);
```

```
LogMessage("The selection of the alarm record in the 6th row (index 5) is now reversed.");
```

UnhideAll() method

The UnhideAll method unhides all hidden alarms.

Syntax

```
AlarmClient.UnhideAll();
```

UnSelectAll() method

The UnSelectAll method unselects all alarm records.

Syntax

```
AlarmClient.UnSelectAll();
```

Unshelve.All() method

The Unshelve.All() method unshelves all alarms that are currently shelved.

Syntax

```
AlarmClient.Unshelve.All(Duration=0;Reason="Reason");
```

Parameters

Duration

Duration must be set to 0 to unshelve alarms.

Reason

Explanation up to 200 characters for unshelving all alarms.
An explanation is optional to unshelve alarms and the Reason parameter can be specified as `Reason=""` to indicate a null explanation.

Example

```
AlarmClient1.Unshelve.All(Duration=0;Reason="");
```

Remarks

For more information about unshelving alarms, see Unshelve Alarms.

Unshelve.Group() method

The Unshelve.Group() method unshelves all currently shelved alarms belonging to a specified provider and alarm group.

Syntax

```
AlarmClient.Unshelve.Group("ProviderName","GroupName",
Duration=0;Reason="Reason");
```

Parameters

ProviderName

Node and or provider name combination that specifies the origin of alarm monitoring.

GroupName

Alarm group or area name whose alarms are monitored by the Alarm Control.

Duration

Duration must be set to 0 to unshelve alarms.

Reason

Explanation up to 200 characters for unshelving all alarms belonging to a specified group and provider.
An explanation is optional to unshelve alarms and the Reason parameter can be specified as `Reason=""` to indicate a null explanation.

Examples

```
AlarmClient1.Unshelve.Group("\\TankServer1\Galaxy","Area_001",
Reason=" + """ + "Shelved"+ """ + ";");
AlarmClient1.Unshelve.Group("\Galaxy","Area_001",Reason=" + """ + "Shelved"+ """ + ";");
```

Remarks

For more information about specifying provider and group names, see Alarm queries.

Unshelve.Priority() method

The Unshelve.Priority() method unshelves currently shelved alarms within a specified alarm priority range that belong to the same specified provider and alarm group.

Syntax

```
AlarmClient.Unshelve.Priority("ProviderName","GroupName",FromPriority,ToPriority,"Duration=
<Duration>;Reason="+"<Reason>"+"UnShelved"+"<Description>"+";");
```

Parameters

ProviderName

Node and provider name combination that specifies the origin of alarm monitoring.

GroupName

Alarm group or area name whose alarms are monitored by the Alarm Control.

FromPriority

Three-digit starting point of the alarm priority range. The `FromPriority` value must be less than the `ToPriority` value.

ToPriority

Three-digit end point of the alarm priority range. The `ToPriority` value must be greater than the `FromPriority` value.

Duration

Duration must be set to 0 to unshelve alarms.

Reason

Explanation up to 200 characters for unshelving all alarms belonging to a specified alarm priority range.
An explanation is optional to unshelve alarms and the Reason parameter can be specified as `Reason=""` to indicate a null explanation.

Example

```
AlarmClient1.Unshelve.Priority("\Galaxy","Area_001",100,600,"Duration=0;Reason="+"""+"UnSh
elved"+"""+";");
```

Remarks

For more information about unshelving alarms, see Unshelve Alarms.

Unshelve.Selected() method

The Unshelve.Selected() method unshelves currently shelved alarms selected by the user from the Alarm Control.

Syntax

```
AlarmClient.Unshelve.Selected(Duration=0;Reason="Reason");
```

Parameters

Duration

Duration must be set to 0 to unshelve alarms.

Reason

Explanation up to 200 characters for unshelving alarms selected by the user.

An explanation is optional to unshelve alarms and the Reason parameter can be specified as `Reason=""` to indicate a null explanation.

Example

```
AlarmClient.Unshelve.Selected(Duration=0;Reason="Maintenance finished");
```

Remarks

For more information about unshelving alarms, see Unshelve Alarms.

Unshelve.SelectedGroup() method

The Unshelve.SelectedGroup() method unshelves all alarms that belong to the same provider and alarm groups as the alarms selected by the user from the Alarm Control.

Syntax

```
AlarmClient.Unshelve.SelectedGroup(Duration=0;Reason="Reason");
```

Parameters

Duration

Duration must be set to 0 to unshelve alarms.

Reason

Explanation up to 200 characters for unshelving alarms that belong to the same alarm groups as the alarms selected by the user.
An explanation is optional to unshelve alarms and the Reason parameter can be specified as `Reason=""` to indicate a null explanation.

Example

```
AlarmClient.Unshelve.SelectedGroup(Duration=0;Reason="");
```

Remarks

For more information about unshelving alarms, see Unshelve Alarms.

Unshelve.SelectedPriority() method

The Unshelve.SelectedPriority method unshelves all alarms within the same alarm priority range as the alarms selected by the user from the Alarm Control.

Syntax

```
AlarmClient.Unshelve.SelectedPriority(Duration=0;Reason="Reason");
```

Parameters

Duration

Duration must be set to 0 to unshelve alarms.

Reason

Explanation up to 200 characters for unshelving the alarms within the same alarm priority range as the alarms selected by the user.
An explanation is optional to unshelve alarms and the Reason parameter can be specified as `Reason=""` to indicate a null explanation.

Example

```
AlarmClient.Unshelve.SelectedPriority(Duration=0;
Reason="Maintenance finished");
```

Remarks

For more information about unshelving alarms, see Unshelve Alarms.

Unshelve.SelectedSeverity() method

The Unshelve.SelectedSeverity method unshelves all shelved alarms within the same alarm severities as the alarms selected by the user from the Alarm Control.

Syntax

```
AlarmClient.Unshelve.SelectedSeverity(Duration=0;Reason="Reason");
```

Parameters

Duration

Duration must be set to 0 to unshelve alarms.

Reason

Explanation up to 200 characters for unshelving the shelved alarms that belong to the same alarm severities as the alarms selected by the user.
An explanation is optional to unshelve alarms and the Reason parameter can be specified as `Reason=""` to indicate a null explanation.

Example

```
AlarmClient.Unshelve.SelectedSeverity(Duration=0;Reason="Low severity alarms");
```

Remarks

For more information about unshelving alarms, see Unshelve Alarms.

Unshelve.SelectedTag() method

The Unshelve.SelectedTag method unshelves all shelved alarms that originate from the same tags as the alarms selected by the user from the Alarm Control.

Syntax

```
AlarmClient.Unshelve.SelectedTag(Duration=0;Reason="Reason");
```

Parameters

Duration

Duration must be set to 0 to unshelve alarms.

Reason

Explanation up to 200 characters for unshelving the active alarms from the same tags as the alarms selected by the user.
An explanation is optional to unshelve alarms and the Reason parameter can be specified as `Reason=""` to indicate a null explanation.

Example

```
AlarmClient.Unshelve.SelectedTag(Duration=0;Reason="");
```

Remarks

For more information about unshelving alarms, see Unshelve Alarms.

Unshelve.Severity() method

The Unshelve.Severity() method unshelves all shelved alarms of a specified alarm severity that belong to the same specified provider and alarm group.

Syntax

```
AlarmClient.Unshelve.Severity("ProviderName","GroupName",Severity,"Duration=<Duration>;Reason="+"<Reason>"+"UnShelved"+"<Description>"+";");
```

Parameters

ProviderName

Node and provider name combination that specifies the origin of alarm monitoring.

GroupName

Alarm group or area name whose alarms are monitored by the Alarm Control.

Severity

Single-digit (1-4) alarm severity.

Duration

Duration must be set to 0 to unshelve alarms.

Reason

Explanation up to 200 characters for unshelving the active alarms of the specified severity.
An explanation is optional to unshelve alarms and the Reason parameter can be specified as `Reason=""` to indicate a null explanation.

Example

```
AlarmClient1.Unshelve.Severity("\Galaxy","Area_001",3,"Duration=0;Reason="+"""+"UnShelved"
+"""+";");
```

Remarks

For more information about unshelving alarms, see Unshelve Alarms.

Unshelve.Tag() method

The Unshelve.Tag() method unshelves all shelved alarms that have the same tag names from the same provider and alarm groups and are within the same priority range.

Syntax

```
AlarmClient.UnShelve.Tag("ProviderName","GroupName","Tag",FromPriority,ToPriority,"Duration
=<Duration>;Reason="+"<Reason>"+"UnShelved"+"<Description>"+";");
```

Parameters

ProviderName

Node and provider name combination that specifies the origin of alarm monitoring.

GroupName

Alarm group or area name whose alarms are monitored by the Alarm Control.

Tag

Name of the tag whose active alarms have been selected to be unshelved.

FromPriority

Three-digit starting point of the alarm priority range. The `FromPriority` value must be less than the `ToPriority` value.

ToPriority

Three-digit end point of the alarm priority range. The `ToPriority` value must be greater than the `FromPriority` value.

Duration

Duration must be set to 0 to unshelve alarms.

Reason

Explanation up to 200 characters for shelving the active alarms that belong to the same alarm severities as the alarms selected by the user.
An explanation is optional to unshelve alarms and the Reason parameter can be specified as `Reason=""` to indicate a null explanation.

Example

```
AlarmClient1.UnShelve.Tag("\Galaxy","Area_001","User_Defined_001.Attribute004.LoLo",1,999,"
Duration=0;Reason="+"""+"UnShelved"+"""+";");
```

Remarks

For more information about unshelving alarms, see Unshelve Alarms.

Unshelve.Visible() method

Th Unshelve.Visible() method unshelves only those alarms that are currently visible from the Alarm Control.

Syntax

```
AlarmClient.Unshelve.Visible("Duration=<Duration>;Reason="+"<Reason>"+"UnShelved"+"<Descrip
tion>"+";");
```

Parameters

Duration

Duration must be set to 0 to unshelve alarms.

Reason

Explanation up to 200 characters for unshelving the alarms currently shown from the Alarm Control.
An explanation is optional to unshelve alarms and the Reason parameter can be specified as Reason="" to indicate a null explanation.

Example

```
AlarmClient1.Unshelve.Visible("Duration=0;Reason="+"""+"UnShelved"+"""+";");
```

Remarks

For more information about unshelving alarms, see Unshelve Alarms.

Configure events

You can execute an action script when the Alarm Control triggers an event. Examples of basic events are:

- Click: The user clicks the Alarm Control
- DoubleClick: The user double-clicks the Alarm Control
- Startup: The Alarm Control opens at run time
- Shutdown: The Alarm Control closes at run time

The Click, Double Click, Start up, and Shutdown events are standard for all .NET client controls. For more information, see the Creating and Managing Industrial Graphics Users Guide.

The Alarm Control has one event of its own that is triggered when a new alarm occurs, the NewAlarm event.

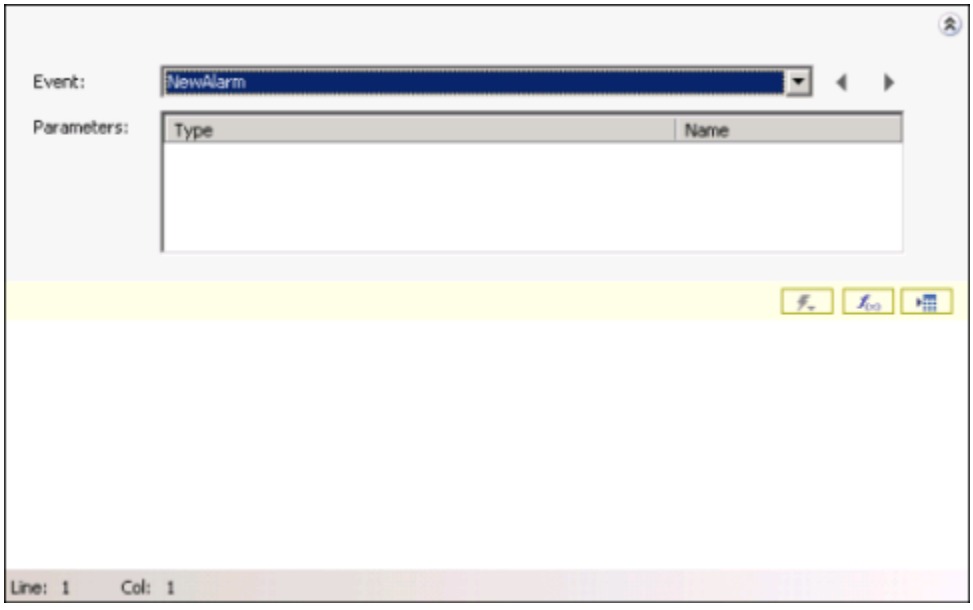
Configure the NewAlarm event

You can configure the NewAlarm event to execute an Industrial graphic script whenever a new alarm occurs.

You can control the trigger behavior with the NewAlarmEventMode property. For more information, see NewAlarmEventMode property.

To configure the NewAlarm event

1. Double-click the Alarm Control. The **Edit Animations** dialog box appears.
2. Click **Event**. The **Event** page appears.
3. In the **Event** list, click **NewAlarm**.



4. In the script area, type the script you want to execute when a new alarm occurs, for example:

```
AlertIcon.Visible = true;
```

5. You must also set the NewAlarmEventMode property to 1 or 2 to enable the NewAlarm event trigger. Do the following:
   a. On the **Special** menu, click **Scripts**. The **Edit Scripts** dialog box appears.
   b. Make sure **Trigger type** is set to **On Show**.
   c. In the script area, type the following:

```
AlarmClient1.NewAlarmEventMode = 1;
```

   d. If you want the script to be executed every time a new alarm occurs, set the NewAlarmEventMode property to 2 instead.
   e. Click **OK**.

.NET colors

The following table is an overview of the .NET color names with hexadecimal code.

Color with Hex Code	**Color with Hex Code**	**Color with Hex Code**
AliceBlue #F0F8FF	AntiqueWhite #FAEBD7	Aqua #00FFFF
Aquamarine #7FFFD4	Azure #F0FFFF	Beige #F5F5DC
Bisque #FFE4C4	Black #000000	BlanchedAlmond #FFEBCD
Blue #0000FF	BlueViolet #8A2BE2	Brown #A52A2A
BurlyWood #DEB887	CadetBlue #5F9EA0	Chartreuse #7FFF00
Chocolate #D2691E	Coral #FF7F50	CornflowerBlue #6495ED
Cornsilk #FFF8DC	Crimson #DC143C	Cyan #00FFFF
DarkBlue #00008B	DarkCyan #008B8B	DarkGoldenrod #B8860B
DarkGray #A9A9A9	DarkGreen #006400	DarkKhaki #BDB76B
DarkMagenta #8B008B	DarkOliveGreen #556B2F	DarkOrange #FF8C00
DarkOrchid #9932CC	DarkRed #8B0000	DarkSalmon #E9967A
DarkSeaGreen #8FBC8B	DarkSlateBlue #483D8B	DarkSlateGray #2F4F4F
DarkTurquoise #00CED1	DarkViolet #9400D3	DeepPink #FF1493
DeepSkyBlue #00BFFF	DimGray #696969	DodgerBlue #1E90FF
Firebrick #B22222	FloralWhite #FFFAF0	ForestGreen #228B22

Color with Hex Code	Color with Hex Code	Color with Hex Code
Fuchsia #FF00FF	Gainsboro #DCDCDC	GhostWhite #F8F8FF
Gold #FFD700	Goldenrod #DAA520	Gray #808080
Green #008000	GreenYellow #ADFF2F	Honeydew #F0FFF0
HotPink #FF69B4	IndianRed #CD5C5C	Indigo #4B0082
Ivory #FFFFF0	Khaki #F0E68C	Lavender #E6E6FA
LavenderBlush #FFF0F5	LawnGreen #7CFC00	LemonChiffon #FFFACD
LightBlue #ADD8E6	LightCoral #F08080	LightCyan #E0FFFF
LightGoldenrodYellow #FAFAD2	LightGray #D3D3D3	LightGreen #90EE90
LightPink #FFB6C1	LightSalmon #FFA07A	LightSeaGreen #20B2AA
LightSkyBlue #87CEFA	LightSlateGray #778899	LightSteelBlue #B0C4DE
LightYellow #FFFFE0	Lime #00FF00	LimeGreen #32CD32
Linen #FAF0E6	Magenta #FF00FF	Maroon #800000
MediumAquamarine #66CDAA	MediumBlue #0000CD	MediumOrchid #BA55D3
MediumPurple #9370DB	MediumSeaGreen #3CB371	MediumSlateBlue #7B68EE
MediumSpringGreen #00FA9A	MediumTurquoise #48D1CC	MediumVioletRed #C71585
MidnightBlue #191970	MintCream #F5FFFA	MistyRose #FFE4E1
Moccasin #FFE4B5	NavajoWhite #FFDEAD	Navy #000080
OldLace #FDF5E6	Olive #808000	OliveDrab #6B8E23
Orange #FFA500	OrangeRed #FF4500	Orchid #DA70D6
PaleGoldenrod #EEE8AA	PaleGreen #98FB98	PaleTurquoise #AFEEEE
PaleVioletRed #DB7093	PapayaWhip #FFEFD5	PeachPuff #FFDAB9
Peru #CD853F	Pink #FFC0CB	Plum #DDA0DD
PowderBlue #B0E0E6	Purple #800080	Red #FF0000
RosyBrown #BC8F8F	RoyalBlue #4169E1	SaddleBrown #8B4513

Color with Hex Code	Color with Hex Code	Color with Hex Code
Salmon #FA8072	SandyBrown #F4A460	SeaGreen #2E8B57
SeaShell #FFF5EE	Sienna #A0522D	Silver #C0C0C0
SkyBlue #87CEEB	SlateBlue #6A5ACD	SlateGray #708090
Snow #FFFAFA	SpringGreen #00FF7F	SteelBlue #4682B4
Tan #D2B48C	Teal #008080	Thistle #D8BFD8
Tomato #FF6347	Transparent #FFFFFF	Turquoise #40E0D0
Violet #EE82EE	Wheat #F5DEB3	White #FFFFFF
WhiteSmoke #F5F5F5	Yellow #FFFF00	YellowGreen #9ACD32

Transfer alarm configuration from InTouch

You can transfer the configuration of the InTouch Alarm Viewer control and the InTouch Alarm DB View control to the configuration of the Alarm Client Control.

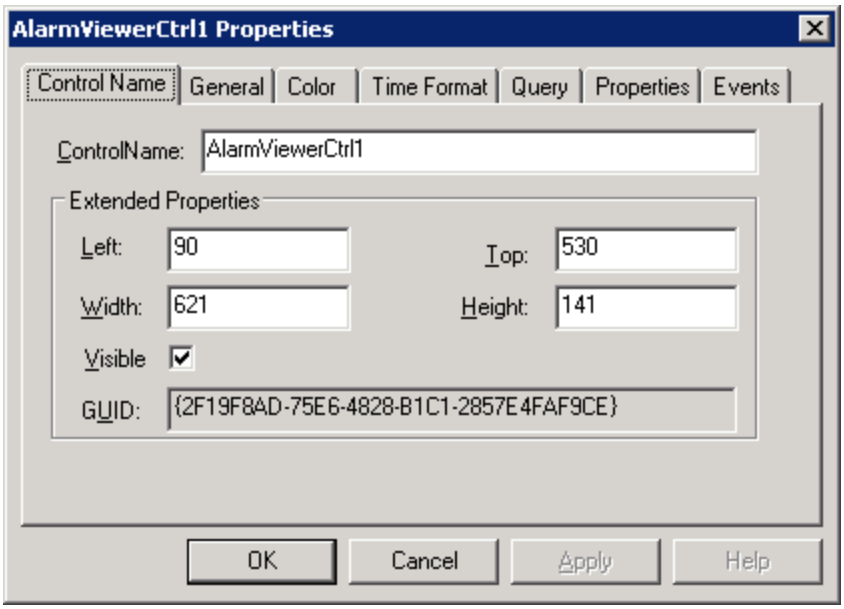
You can also map the InTouch alarm control properties and methods to the properties and methods of the Alarm Client Control.

Transfer the InTouch Alarm Viewer control configuration

You can transfer the configuration of the InTouch Alarm Viewer control tabs options to the Alarm Client Control.

Transfer configuration of the Control Name tab

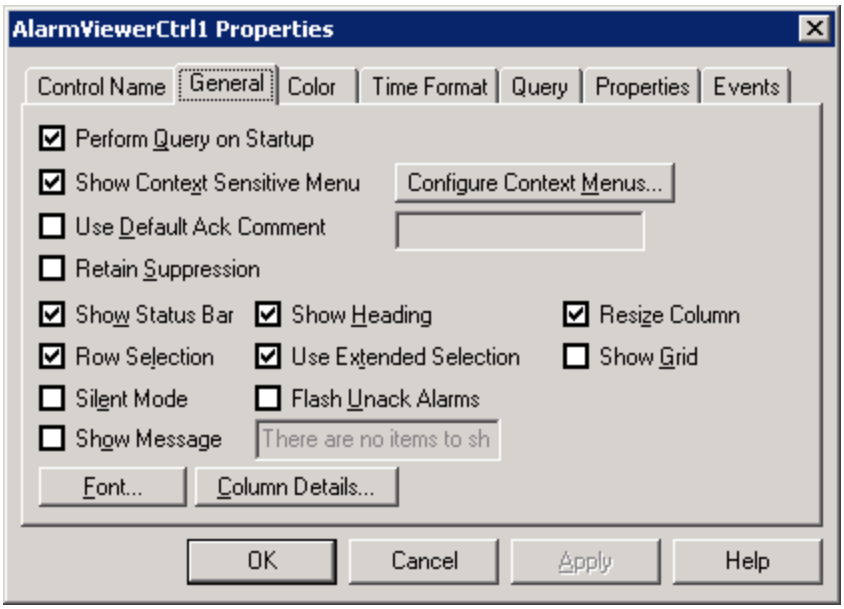
You can transfer the configuration of the **Control Name** tab options of the InTouch Alarm Viewer control to the Alarm Client Control.



InTouch option	Alarm Control option
ControlName	You can rename the Alarm Client Control the same way as any other elements on the canvas. For more information, see the Creating and Managing Industrial Graphics User's Guide.
Left, Top, Width, and Height	You can directly edit the positioning options in the same way as any other element on the canvas. Edit the following properties in the Properties Editor: **X**, **Y**, **Width**, and **Height**.
Visible	You can directly edit the visibility option in the same way as any other element on the canvas. In the Properties Editor, edit the **Visible** property.
GUID	This option has no meaning in the Alarm Client Control.

Transfer configuration of the General tab

You can transfer the configuration of the **General** tab options of the InTouch Alarm Viewer control to the Alarm Client Control.

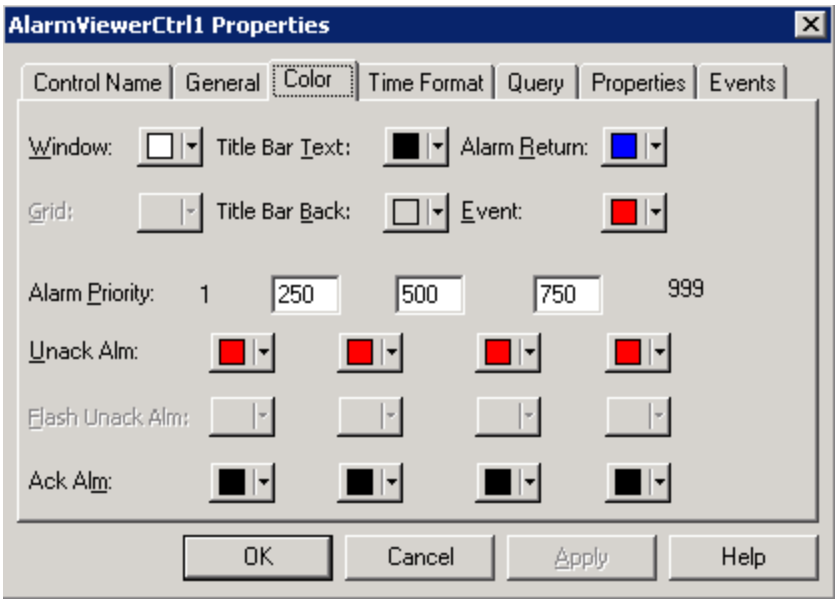


InTouch option	Alarm Control option
Perform Query on Startup	In the Alarm Client Control, this option is called **Query on Startup.** You can configure this option on the **Run-Time Behavior** page.
Show Context Sensitive Menu	In the Alarm Client Control, this option is called **Show Context Menu**. You can configure this option on the **Run-Time Behavior** page.
Configure Context Menus	In the Alarm Client Control, you can configure the availability of individual shortcut menu options at run-time directly on the **Run-Time Behavior** page.
Use Default Ack Comment	In the Alarm Client Control, you can configure the availability of the **Query Filters** shortcut menu option on the **Run-Time Behavior** page.
Retain Suppression	In the Alarm Client Control, this option is called **Retain Hidden**. You can configure it on the **Run-Time Behavior** page.
Show Status Bar	In the Alarm Client Control, you can configure the **Show Grid** option on the **Run-Time Behavior** page.
Show Heading	In the Alarm Client Control, you can configure the **Show Status Bar** option on the **Run-Time Behavior** page.
Row Selection	In the Alarm Client Control, this option is called **Row Selection**. You can configure it on the **Run-Time Behavior** page.

InTouch option	Alarm Control option
Use Extended Selection	In the Alarm Client Control, this option is called **Row Selection**. You can configure it on the **Run-Time Behavior** page.
Show Grid	In the Alarm Client Control, you can configure the **Show Grid** option on the **Run-Time Behavior** page.
Retrieve Buttons	In the Alarm Client Control, the retrieve buttons are not available. The underlying grid technology handles the alarm retrieval from the alarm database.
Silent Mode	In the Alarm Client Control, this option is called **Hide Errors and Warnings**. You can configure it on the **Run-Time Behavior** page.
Flash Unack Alarms	In the Alarm Client Control, you can configure the **Flash Unack Alarms** option on the **Colors** page.
Show Message	In the Alarm Client Control, this option is called **Show Custom 'No Records' Message**. You can configure it on the **Run-Time Behavior** page.
Font	You can configure this option from the Industrial Graphic Editor page. Select the Alarm Client Control on the canvas and select an appropriate font type, size, and style on the menu bars.
Column Details	In Alarm Client Control, you can configure the column details directly on the **Column Details** page.

Transfer configuration of the Color tab

You can transfer the configuration of the **Color** tab options of the InTouch Alarm Viewer control to the Alarm Client Control.



All the options of the **Color** tab in the InTouch Alarm Viewer control can be set on the **Colors** page of the Alarm Client Control.

The following table shows you some minor differences in wording:

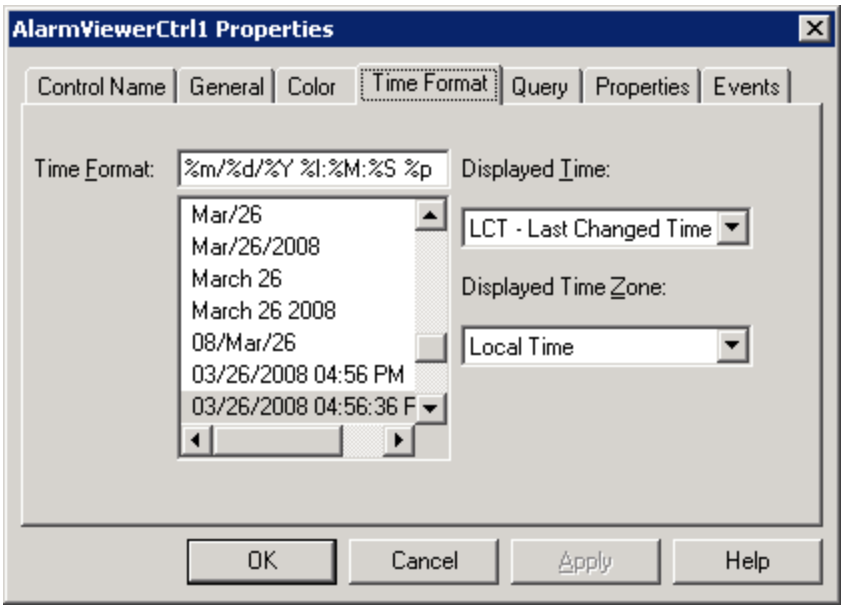
InTouch Alarm View control	**Alarm Client Control**
Title Bar Text	Heading Text
Title Bar Back	Heading Background
Alarm Return	Alarm RTN

you can also set he background color in addition to the text color for most of the alarm records.

You can set the alarm priority range breakpoints directly in the table in the **From Pri** column.

Transfer configuration of the Time Format tab

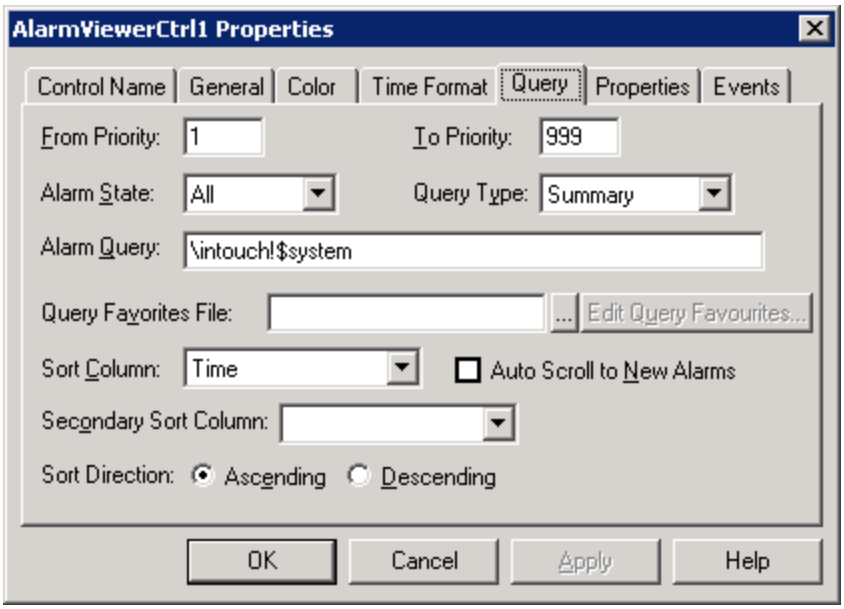
You can transfer the configuration of the **Time Format** tab options of the InTouch Alarm Viewer control to the Alarm Client Control.



InTouch option	Alarm Control option
Time Format	In the Alarm Client Control, you can configure the **Time Format** option on the **Time Settings** page.
Displayed Time	This option has no meaning in the Alarm Client Control. All alarm records are shown with the following time stamps in the Alarm Control grid: • **Time (OAT)**: Original Alarm Time • **Time (LCT)**: Last Changed Time • **Time (LCT, OAT)**: Last Changed Time, but Original Alarm Time if the alarm record is unacknowledged
Displayed Time Zone	In the Alarm Client Control, this option is called **Time Zone**. You can configure it on the **Time Settings** page. You need to explicitly configure the time zone for the correct time stamp.

Transfer configuration of the Query tab

You can transfer the configuration of the **Query** tab options of the InTouch Alarm Viewer control to the Alarm Client Control.

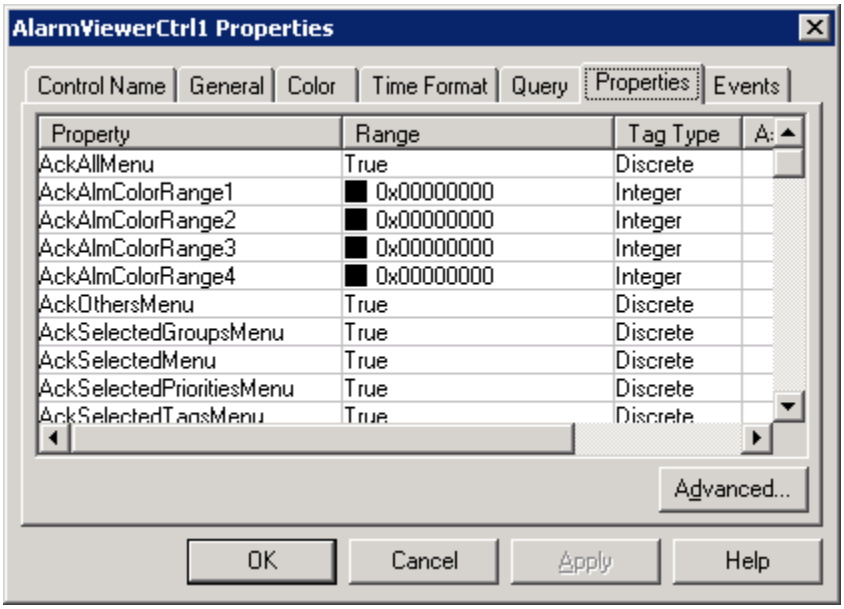


InTouch option	**Alarm Client Control option**
From Priority, To Priority	In the Alarm Client Control, you can only set the priority limits as part of a query filter on the **Query Filters** page. For more information, see Filter alarms.
Alarm State	In the Alarm Client Control, you can only set the alarm state limitation as part of a query filter on the **Query Filters** page. for more information, see Filter alarms.
Query Type	In the Alarm Client Control, you can set the **Client Mode** option on the **Alarm Mode** page as follow: • For query type "Summery", set the client mode to **Current Alarms** • For query type "Historical", set the client mode to **Recent Alarms and Events**
Alarm Query	In the Alarm Client Control, you can configure the **Alarm Query** option on the **Alarm Mode** page.
Query Favorites File, Edit Query Favorites	In the Alarm Client Control, all query favorites and filter favorites are managed on one page and are interchangeable between different client modes. To access the Query Filter Favorites, open the **Query Filters** page.
Sort Column	In the Alarm Client Control, you can configure the sorting of alarm records on the **Column Details** page.

InTouch option	Alarm Client Control option
Auto Scroll to New Alarms	In the Alarm Client Control, you can configure the **Auto Scroll to New Alarms** on the **Run-Time Behavior** page.
Secondary Sort Column, Sort Direction	In the Alarm Client Control, you can configure the sorting of alarm records on the **Column Details** page.

Transfer configuration of the Properties tab

You can set the properties of the Alarm Client Control in the **Properties Editor** when the Alarm Control is selected on the canvas.



For more information on the exact mapping between the InTouch Alarm Viewer control properties and Alarm Client Control properties, see Map properties and methods.

The advanced property filtering feature does not exist in the Alarm Client Control. However, when you browse for properties of the Alarm Client Control from other elements with the **Galaxy Browser**, you can filter the properties. Also, the properties of the Alarm Client Control are logically grouped in the Properties Editor.

Transfer script configuration on the Events tab

You can configure scripts for events of the Alarm Client Control on the **Event** animation page. The events are the same as the events for the InTouch Alarm Viewer Control:

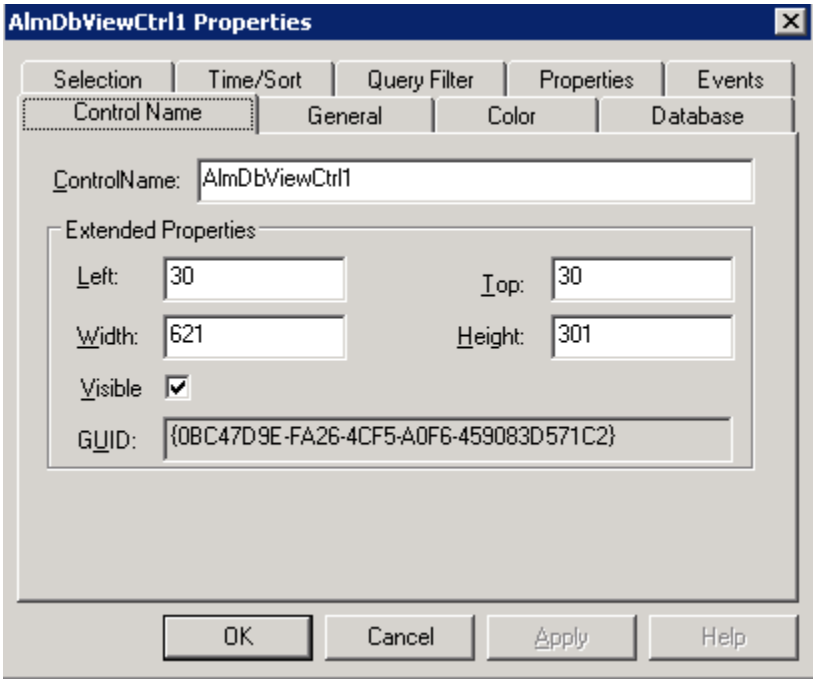
• Click	• Shutdown
• DoubleClick	• StartUp
• New Alarm	

Transfer the InTouch Alarm DB View control configuration

You can transfer the configuration of the InTouch Alarm DB View control tabs options to the Alarm Client Control.

Transfer configuration of the Control Name tab

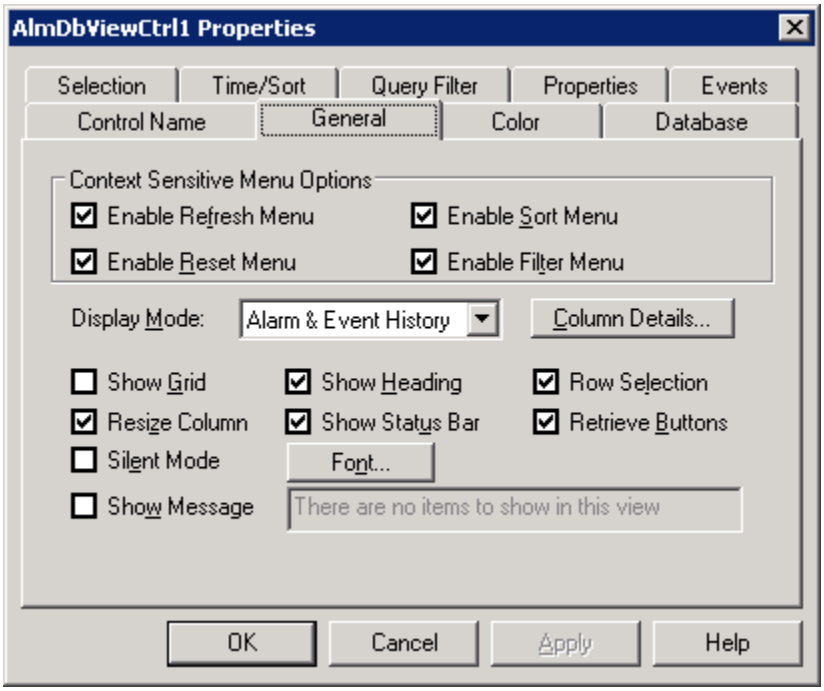
You can transfer the configuration of the **Control Name** tab options of the InTouch Alarm DB View control to the Alarm Client Control.



InTouch option	Alarm Control option
ControlName	You can rename the Alarm Client Control the same way as any other elements on the canvas. For more information, see the Creating and Managing Industrial Graphics User's Guide.
Left, Top, Width, and Height	You can directly edit the positioning options in the same way as any other element on the canvas. Edit the following properties in the Properties Editor: **X**, **Y**, **Width**, and **Height**.
Visible	You can directly edit the visibility option in the same way as any other element on the canvas. In the Properties Editor, edit the **Visible** property.
GUID	This option has no meaning in the Alarm Client Control.

Transfer configuration of the General tab

You can transfer the configuration of the **General** tab options of the InTouch Alarm DB View control to the Alarm Client Control.

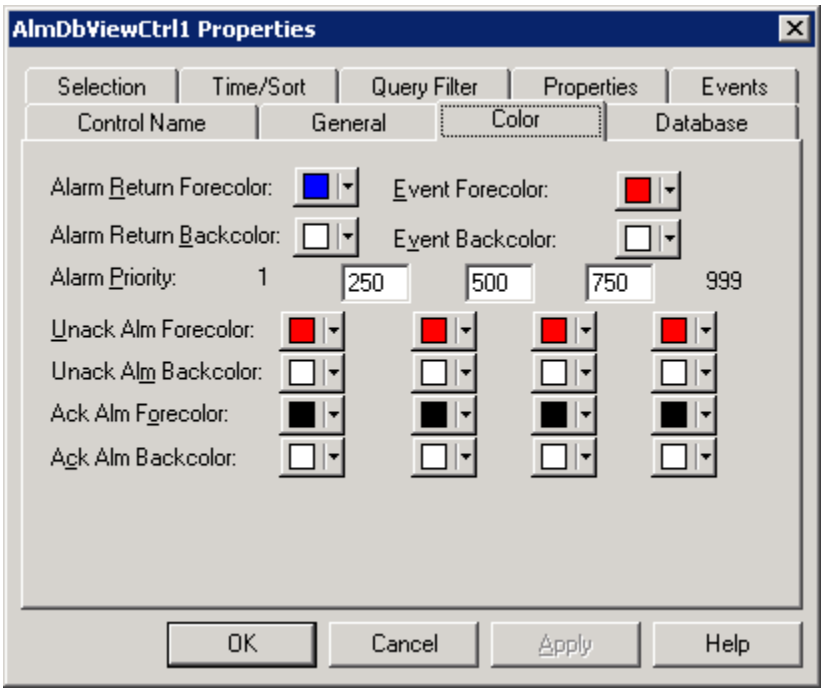


InTouch option	Alarm Control option
Enable Refresh Menu	In the Alarm Client Control, you can configure the availability of the **Requery** shortcut menu option on the **Run-Time Behavior** page.
Enable Sort Menu	In the Alarm Client Control, you can configure the availability of the **Sort** shortcut menu option on the **Run-Time Behavior** page.
Enabled Reset Menu	In the Alarm Client Control, you can configure the availability of the **Reset** shortcut menu option on the **Run-Time Behavior** page.
Enabled Filter Menu	In the Alarm Client Control, you can configure the availability of the **Query Filters** shortcut menu option on the **Run-Time Behavior** page.
Display Mode	In the Alarm Client Control, set the **Client Mode** on the **Alarm Mode** page to the same setting as the **Display Mode** setting in the InTouch Alarm DB View control.
Column Details	In the Alarm Client Control, you can configure the column details directly on the **Column Details** page.
Show Grid	In the Alarm Client Control, you can configure the **Show Grid** option on the **Run-Time Behavior** page.
Show Heading	In the Alarm Client Control, you can configure the **Show Heading** option on the **Run-Time Behavior** page.
Row Selection	In the Alarm Client Control, this option is called **Row Selection**. You can configure it on the **Run-Time Behavior** page.
Resize Column	In the Alarm Client Control, this option is called **Allow Column Resizing**. You can configure it on the **Run-Time Behavior** page.
Show Status Bar	In the Alarm Client Control, you can configure the **Show Status Bar** option on the **Run-Time Behavior** page.
Retrieve Buttons	In the Alarm Client Control, the retrieve buttons are not available. The underlying grid technology handles the alarm retrieval from the alarm database.
Silent Mode	In the Alarm Client Control, this option is called **Hide Errors and Warnings**. You can configure it on the **Run-**

InTouch option	Alarm Control option
	Time Behavior page.
Font	You can configure this option from the Industrial Graphic Editor page. Select the Alarm Client Control on the canvas and select an appropriate font type, size, and style on the menu bars.
Show Message	In the Alarm Client Control, this option is called **Show Custom 'No Records' Message**. You can configure it on the **Run-Time Behavior** page.

Transfer configuration of the Color tab

You can transfer the configuration of the **Color** tab options of the InTouch Alarm DB View control to the Alarm Client Control.



All the options of the **Color** tab in the InTouch Alarm DB View control can be set on the **Colors** page of the Alarm Client Control.

The following table shows you some minor differences in wording:

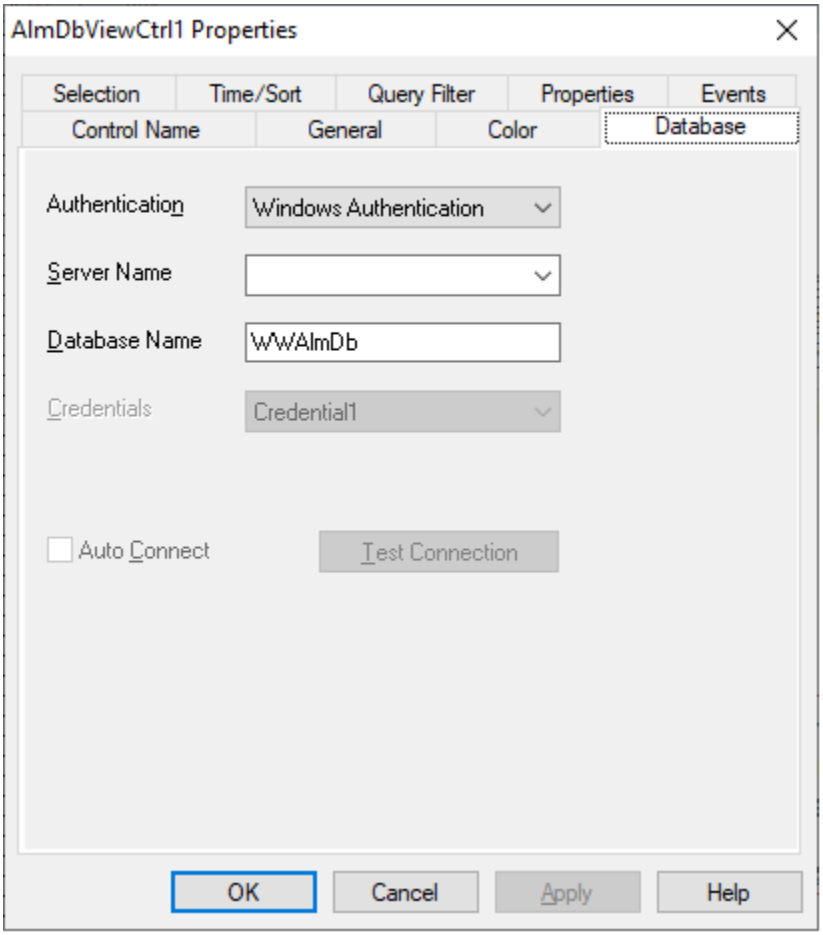
InTouch Alarm DB View control	Alarm Client Control
Forecolor	Text
Backcolor	Background

InTouch Alarm DB View control	Alarm Client Control
Alm	n/a
Return	RTN

You can set the alarm priority range breakpoints directly in the table in the **From Pri** column.

Transfer configuration of the Database tab

You can transfer the configuration of the **Database** tab options of the InTouch Alarm DB View control to the Alarm Client Control.



In the Alarm Client Control, you can configure the following options on the **Alarm Mode** page:

• **Server Name**	• **Database Name**
• **User**	• **Password**

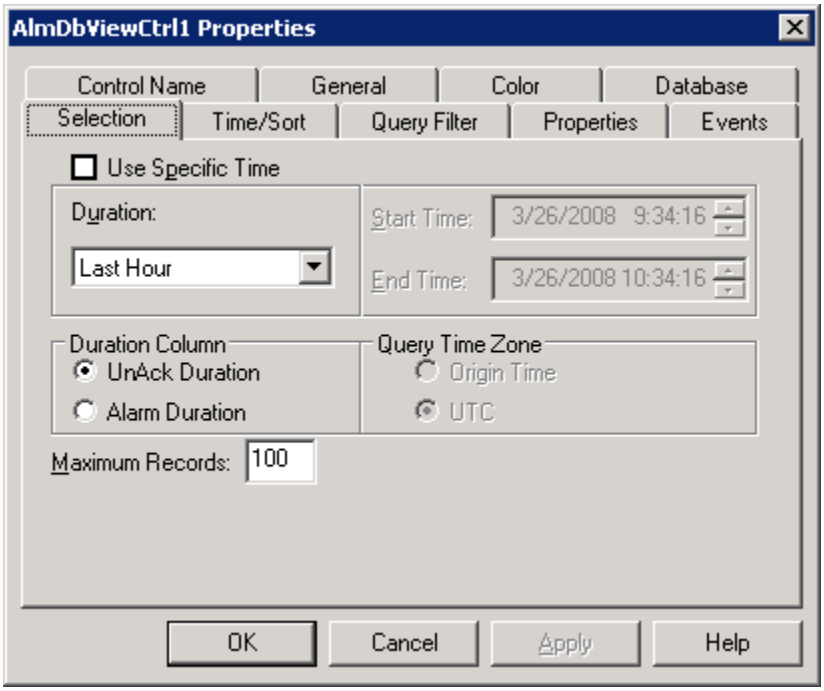
• **Credentials**	• **Test Connection**

In the Alarm Client Control, the **Auto Connect** option is called **Query on Startup**. You can configure it on the **Run-Time Behavior** page.

The configuration for the Alarm Database only appears if the **Client Mode** is set to **Historical Alarms**, **Historical Events**, or **Historical Alarms and Events**.

Transfer configuration of the Selection tab

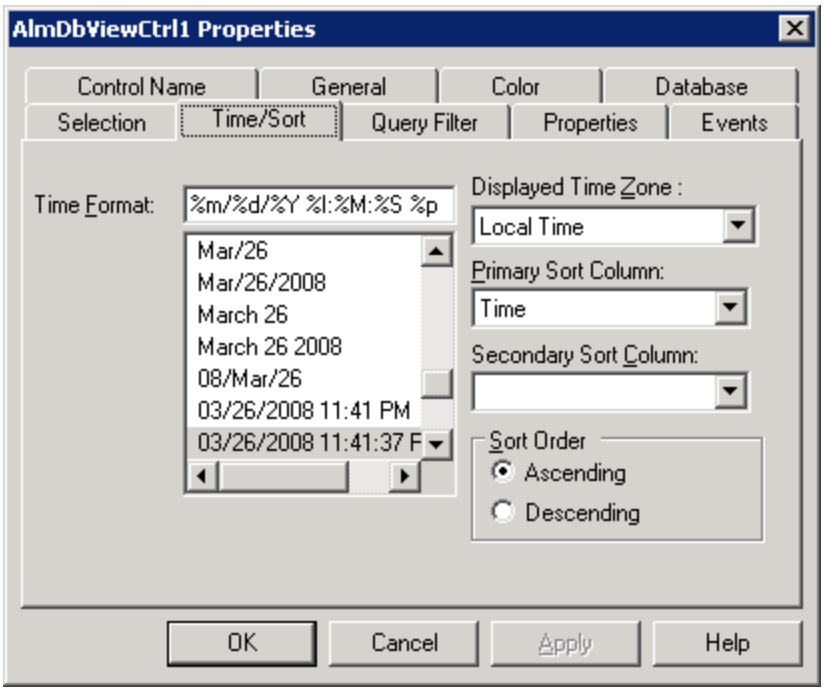
You can transfer the configuration of the **Selection** tab options of the InTouch Alarm DB View control to the Alarm Client Control.



InTouch option	Alarm Control option
Use Specific Time, Start Time, End Time	In the Alarm Client Control, you can set these options directly in the Time Range Picker control on the **Alarm Mode** page. When you select a time from either the start time or end time part of the Time Range Picker control, the Alarm Control is automatically set to use a specific time. To keep the specific start and end time, you must also clear **Update to Current Time**. When you refresh the Alarm Control grid at run time, the time range stays fixed to the given start and end time.
Duration	In the Alarm Client Control, you can set this option directly in the Time Range Picker control on the **Alarm Mode** page. When you select a duration from the center part of the Time Range Picker control, the Alarm Control is automatically set to use a time offset. To keep the duration, you must also select the **Update to Current Time** check box. When you refresh the Alarm Control grid at run time, the end time is set to the current time and the Alarm Control shows the alarms within the set duration.
UnAck Duration, Alarm Duration	In the Alarm Client Control, you cannot configure the Unack Duration and Alarm Duration settings. The Alarm Control grid shows both UnAck Duration and Alarm Duration in separate columns.
Query Time Zone	In the Alarm Client Control, you can configure the **Time Zone** setting on the **Time Settings** page.
Maximum Records	In the Alarm Client Control, you can configure the **Maximum Records** setting on the **Alarm Mode** page.

Transfer configuration of the Time/Sort tab

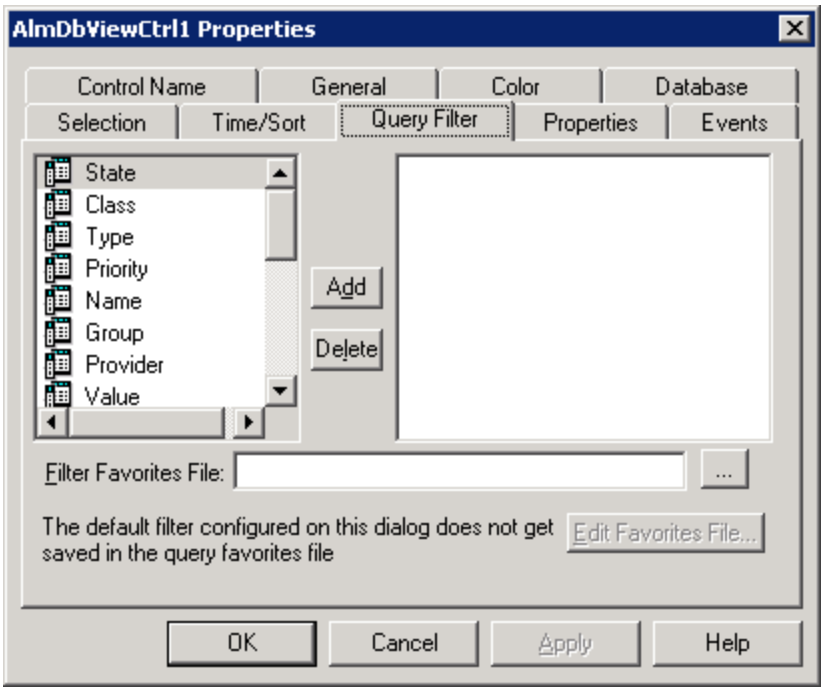
You can transfer the configuration of the **Time/Sort** tab options of the InTouch Alarm DB View control to the Alarm Client Control.



InTouch option	Alarm Control option
Time Format	In the Alarm Client Control, you can configure the **Time Format** setting on the **Time Settings** page.
Displayed Time Zone	In the Alarm Client Control, you can configure the **Time Zone** setting on the **Time Settings** page.
Primary Sort Column, Secondary Sort Column, Sort Order	In the Alarm Client Control, you can configure the sorting options on the **Column Details** page.

Transfer configuration of the Query Filter tab

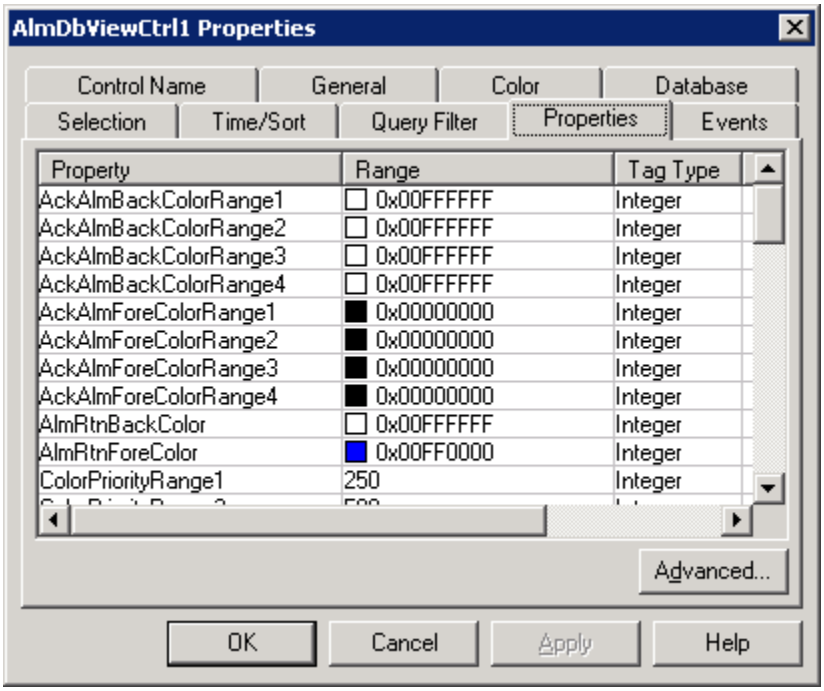
You can transfer the configuration of the **Query Filter** tab options of the InTouch Alarm DB View control to the Alarm Client Control.



In the Alarm Client Control, all query favorites and filter favorites are managed on one page and are interchangeable between different client modes. To access the Query Filter Favorites, open the **Query Filters** page.

Transfer configuration of the Properties tab

You can set the properties of the Alarm Client Control in the **Properties Editor** when the Alarm Control is selected on the canvas.



For more information on the exact mapping between the InTouch Alarm DB View control properties and Alarm

Client Control properties, see Map properties and methods.

The advanced property filtering feature does not exist in the Alarm Client Control. However, when you browse for properties of the Alarm Client Control from other elements with the **Galaxy Browser**, you can filter the properties. Also, the properties of the Alarm Client Control are logically grouped in the Properties Editor.

Transfer scripts configuration on the Events tab

You can configure scripts for events of the Alarm Client Control on the **Event** animation page. The events are the same as the events for the InTouch Alarm DB View control:

• Click	• Shutdown
• DoubleClick	• StartUp
• NewAlarm	

For more information, see Configure events.

Transfer query favorites configuration

You can only transfer query favorites configuration from InTouch to the Alarm Client Control by recreating the filters on the Query Filters page.

If you intend to use a the query filter in one of the current client modes, make sure you also include **Provider** and **Group** as filter criteria.

Map properties and methods

The following table shows all properties and methods of the InTouch Alarm Viewer control and InTouch Alarm DB View controls and their corresponding properties and methods of the Alarm Client Control.

InTouch alarm control	**Property or Method**
DisplayedTimeZone	TimeZone.TimeZone property
DisplayMode	ClientMode property
Duration	TimeSelector.TimeDuration property
EndTime	TimeSelector.EndDate property
EventBackColor	EventColor.BackGround property
EventColor	EventColor.ForeGround property

InTouch alarm control	Property or Method
EventForeColor	EventColor.ForeGround property
ExtendedSelection	RowSelection property
FilterFavoritesFile	No corresponding property. The file name is used as a parameter for the Favorites.Export() method and Favorites.Import() method methods.
FilterMenu	ContextMenu.Favorites property
FilterName	Favorite Property
FlashUnackAlarms	FlashUnAckAlarms property
FlashUnAckAlmColorRange1	AlarmColor.UnAck.Flash.ForeGround property
FlashUnAckAlmColorRange2	AlarmColor.UnAck.Flash.ForeGround property
FlashUnAckAlmColorRange3	AlarmColor.UnAck.Flash.ForeGround property
FlashUnAckAlmColorRange4	AlarmColor.UnAck.Flash.ForeGround property
Font	You can only set the font at design time, not at run time.
FreezeDisplay()	FreezeDisplay() method
FreezeMenu	ContextMenu.Freeze property
FromPriority	No corresponding property. Configure a Query Filter favorite at design time instead and use the Favorite property. For more information, see Favorite Property.
GetItem()	GetItem() method
GetLastError()	GetLastError() method
GetNext()	No corresponding property. Alarm records are retrieved one by one from the Alarm Database after the initial set of alarm records is retrieved. The initial set is defined by the **Maximum Records** setting.
GetPrevious()	No corresponding property. Alarm records are retrieved one by one from the Alarm Database after the initial set of alarm records is retrieved. The initial set is defined by the **Maximum Records** setting.
GetSelectedItem()	GetSelectedItem() method

InTouch alarm control	Property or Method
GridColor	GridColor property
GroupExactMatch	No corresponding property. Configure a Query Filter favorite at design time instead and use the Favorite property. For more information, see Favorite Property.
GroupName	No corresponding property. Configure a Query Filter favorite at design time instead and use the Favorite property. For more information, see Favorite Property.
MaxRecords	MaxDatabaseRecords property
MoveWindow()	MoveWindow() method
NewAlarmEventMode	NewAlarmEventMode property
Password	Database.Authentication property
PrimarySort	SortOrder.First property
ProviderExactMatch	No corresponding property. Configure a Query Filter favorite at design time instead and use the Favorite property. For more information, see Favorite Property.
ProviderName	No corresponding property. Configure a Query Filter favorite at design time instead and use the Favorite property. For more information, see Favorite Property.
QueryFavoritesFile	No corresponding property. The file name is used as a parameter for the Favorites.Export() method and Favorites.Import() method methods.
QueryFavoritesMenu	ContextMenu.Favorites property
QueryName	Favorite Property
QueryStartup	QueryStartup property
QueryTimeZone	TimeZone.TimeZone property
QueryType	ClientMode property
Refresh()	Requery() method
RefreshMenu	ContextMenu.Requery property
Requery()	Requery() method
RequeryMenu	ContextMenu.Requery property

InTouch alarm control	Property or Method
Reset()	Reset() method
ResetMenu	ContextMenu.Reset property
RetainSuppression	RetainHidden property
RowCount	RowCount property
RowSelection	RowSelection property
SecondarySort	SortColumn.Second property
SecondarySortColumn	SortColumn.Second property
SelectAll()	To select all records, see Select.All() method. To reverse the selection of all records, see Toggle.All() method.
SelectedCount	SelectedCount property
SelectGroup()	Select.Group() method
SelectItem()	To select a given alarm record, see Select.Item() method. To reverse the selection of a given alarm record, see Toggle.Item() method.
SelectPriority()	Select.Priority() method
SelectQuery()	Favorite Property
SelectTag()	Select.Tag() method
ServerName	Database.ServerName property
SetQueryByName	Favorite Property
SetSort()	SetSort() method
ShowContext()	Show.Context() method
ShowContextMenu	ShowContextMenu property
ShowDate	There is no equivalent functionality in the Alarm Client Control.
ShowFetch	No corresponding property. The buttons for retrieving sets of alarm records from the Alarm Database do not exist in the Alarm Client Control.

InTouch alarm control	Property or Method
ShowFilter()	Show.Favorite() method
ShowGrid	ShowGrid property
ShowHeading	ShowHeading property
ShowMessage	NoRecordsMessage.Enabled property
ShowQueryFavorites()	Show.Favorite() method
ShowSort()	Show.Sort() method
ShowStatistics()	Show.Statistics() method
ShowStatusBar	ShowStatusBar property
ShowSuppression()	Show.Hidden() method
SilentMode	HideErrors property
SortColumn	You can set three sort columns in the Alarm Client Control. To set the first column, see SortColumn.First property.
SortMenu	ContextMenu.Sort property
SortOnCol()	To set the first sort column, see SortColumn.First Property. To set the sort order of the first sort column, see SortOrder.First property.
SortOrder	SortOrder.First property
SpecificTime	UpdateToCurrentTime property
StartTime	TimeSelector.EndDate property
StatsMenu	ContextMenu.Statistics property
SuppressAll()	Hide.All() method
SuppressAllMenu	ContextMenu.HideAll property
SuppressedAlarms	HiddenAlarms property
SuppressGroup()	Hide.Group() method
SuppressionMenu	ContextMenu.Hidden property
SuppressOthersMenu	ContextMenu.HideOthers property

InTouch alarm control	Property or Method
SuppressPriority()	Hide.Priority() method
SuppressSelected()	Hide.Selected() method
SuppressSelectedGroup()	Hide.SelectedGroup() method
SuppressSelectedGroupsMenu	ContextMenu.HideSelectedGroups property
SuppressSelectedMenu	ContextMenu.HideSelected property
SuppressSelectedPrioritiesMenu	ContextMenu.HideSelectedPriorities property
SuppressSelectedPriority()	Hide.SelectedPriority() method
SuppressSelectedTagsMenu	ContextMenu.HideSelectedTags property
SuppressSelectedTag()	Hide.SelectedTag() method
SuppressTag()	Hide.Tag() method
SuppressVisible()	Hide.Visible() method
SuppressVisibleMenu	ContextMenu.HideVisible property
Time	Time.Type property and Time.Format property
TimeFormat	Time.Format property and Time.Type property
TitleBackColor	HeadingColor.BackGround property
TitleForeColor	HeadingColor.ForeGround property
ToPriority	No corresponding property. Configure a Query Filter favorite at design time instead and use the Favorite Property. For more information, see Favorite Property.
TotalAlarms	TotalRowCount property
TotalRowCount	TotalRowCount property
UnAckAlarms	UnAckAlarms property
UnAckAlmBackColor	AlarmColor.UnAck.BackGround property
UnAckAlmBackColorRange1	AlarmColor.UnAck.BackGround property
UnAckAlmBackColorRange2	AlarmColor.UnAck.BackGround property
UnAckAlmBackColorRange3	AlarmColor.UnAck.BackGround property

InTouch alarm control	Property or Method
UnAckAlmBackColorRange4	AlarmColor.UnAck.BackGround property
UnAckAlmColorRange1	AlarmColor.UnAck.ForeGround property
UnAckAlmColorRange2	AlarmColor.UnAck.ForeGround property
UnAckAlmColorRange3	AlarmColor.UnAck.ForeGround property
UnAckAlmColorRange4	AlarmColor.UnAck.ForeGround property
UnAckAlmForeColor	AlarmColor.UnAck.ForeGround property
UnAckAlmForeColorRange1	AlarmColor.UnAck.ForeGround property
UnAckAlmForeColorRange2	AlarmColor.UnAck.ForeGround property
UnAckAlmForeColorRange3	AlarmColor.UnAck.ForeGround property
UnAckAlmForeColorRange4	AlarmColor.UnAck.ForeGround property
UnAckOrAlarmDuration	No corresponding property. UnAck Duration and Alarm Duration are shown in the Alarm Control grid.
UnSelectAll()	UnSelectAll() method
UnSuppressAll()	UnhideAll() method
UnsuppressAllMenu	ContextMenu.UnhideAll property
UseDefaultAckComment	AckComment.UseDefault property
UserID	Database.UserID property
Visible	Visible property
WindowColor	WindowColor property

The Trend Client

To use the Trend Client, you should already have a basic understanding of the Integrated Development Environment (IDE), the Industrial Graphic Editor, and the AVEVA InTouch HMI before continuing. For more information, see the documentation for each of these products.

About the Trend Client

Use the Trend Client to chart attribute current values from Application Server and tag current values from the InTouch HMI software. This differs from the trend feature in the InTouch HMI because Trend Client fully supports ArchestrA and can initialize a trend pen with data from a historical source such as InTouch LogHistory/LGH files or the Historian.

Note: The InTouch trends are still included as part of the InTouch HMI.

Differences between the Trend Client control and the Historian Client Trend control

If you use the Historian Client Trend control, you need to be aware that there are several differences between the Trend Client Control and the Historian Client Trend control.

Description of the Historian Client Trend control

AVEVA Historian Client Trend is a client application that allows you to query tags from a Historian database and plot them on a graphical display. The Historian Client Trend supports two different chart types: a regular trend curve and an XY scatter plot.

Before you can use the Historian Client Trend control to query tag information from the database, the Historian Server must be running and you must have the necessary access privileges.

Description of the Trend Client control

In contrast to the Historian Client Trend control, the Trend Client control:

- Includes the real-time as well as historical trending functionality.
- Uses the data from a historical source such as InTouch Log History (LGH) files or from the Historian.
- Has fewer pens.
- Uses real-time attribute or tag data as well as historized data stored in the Historian or InTouch Log History (LGH) files.
- Does not require a separate license from the Application Server.
- Requires the InTouch HMI.
- Only supports the full and cyclic retrieval modes of the Historian when retrieving data.

- Does not support configuring the default pen colors (although you can change them after you add a pen).
- Does not provide scatter plots and target regions.

Data sources for the Trend Client

The Trend Client can use real-time data from Application Server and the InTouch HMI software and historical data from the Historian or InTouch LGH files. You identify the historical data source when you configure a pen. For more information, see Configuring Historical Sources.

At run time, the Trend Client initializes the trend with historical data from the data source (if configured) and then appends real-time data for the pens. If the pen is not associated with a historical data source, the trend initially shows no data. It then starts charting information as real-time data becomes available. If you refresh a pen configured with historical data, the chart re-queries the historical information, after which it appends the real-time data as it becomes available. Refreshing does not change a pen configured only for real-time data. If you refresh a trend that is charting both historical and real-time data, the historical information is recharted immediately, after which the trend charts real-time data as it becomes available.

The Trend Client supports the same syntax as the Industrial Graphics. It also supports the built-in functionality of the OwningObject property for trends configured with relative naming. For information about Industrial Graphics data configuration syntax and OwningObject functionality, see the Creating and Managing Industrial Graphics User Guide. For more information on relative and absolute references, see the Application Server User Guide.

There are primary settings for specifying the time period you want to plot, such as a relative reference (the preceding two hours), or an absolute reference (July 15th at 10am for 2 hours) using scripting.

As an example, suppose you opened two trends created with the Trend Client. Both of them have a duration of 15 minutes, but the first one is only trending real-time data (meaning that the pen is associated with an expression but not with a historical tag) and the second is trending historical data (meaning that the pen is associated with both an expression and a historical tag). If you open both of these trends at the same time, what you see is described in the following table.

Time after starting both trends	What you see
0 minutes	The first trend shows no data. The second trend shows 15 minutes of historical data from the source.
5 minutes	The first trend shows 5 minutes of real-time data. The second trend shows 10 minutes of the historical data and 5 minutes of real-time data.
15 minutes	Both charts show 15 minutes of real-time data. All the historical data will have been pushed off the second trend by the new real-time data.

Configure the Trend Client

This section shows you how to place a Trend Client on the canvas and configure it. You can configure it either with the **Edit Animations** dialog box, or by changing individual properties in the Properties Editor

Place the Trend Client into an Industrial Graphic

You can easily place the Trend Client into an Industrial Graphic by placing it onto the canvas.

To place the Trend Client into an Industrial Graphic

1. Open the Industrial Graphic.

2. On the **Tools** panel, click the **Trend Client** icon. The cursor appears in insert mode.
3. Click on the canvas where you want to place the upper left corner of the Trend Client.

Remove the Trend Client from an Industrial Graphic

You can remove the Trend Client from the canvas.

To remove the Trend Client from an Industrial Graphic

1. Open the graphic you want to remove the Trend Client from.
2. Do either of the following:
   - Select the Trend Client and press **Delete**.
   - Right-click the Trend Client and select **Delete** from the menu.

Validate the configuration of the Trend Client

You can validate the configuration of a Trend Client. If the configuration has an error, an exclamation mark appears next to the **Animation** icon.

Examples of errors include:

- Syntax errors in expressions
- Data type mismatches
- Required values not specified
- Specified values out of valid range

To validate the configuration of a Trend Client

1. Open the symbol containing the Trend Client.
2. Double-click the Trend Client. The **Edit Animations** dialog box appears.
3. Click the **Validate** icon on the right side of the **Edit Animations** dialog box.

   The Trend Client is validated. Possible errors are highlighted in the right side of the dialog box.

Clear the configuration from the Trend Client

You can clear all configuration data from the configuration boxes of and reset the settings to the defaults.

To clear all configuration data

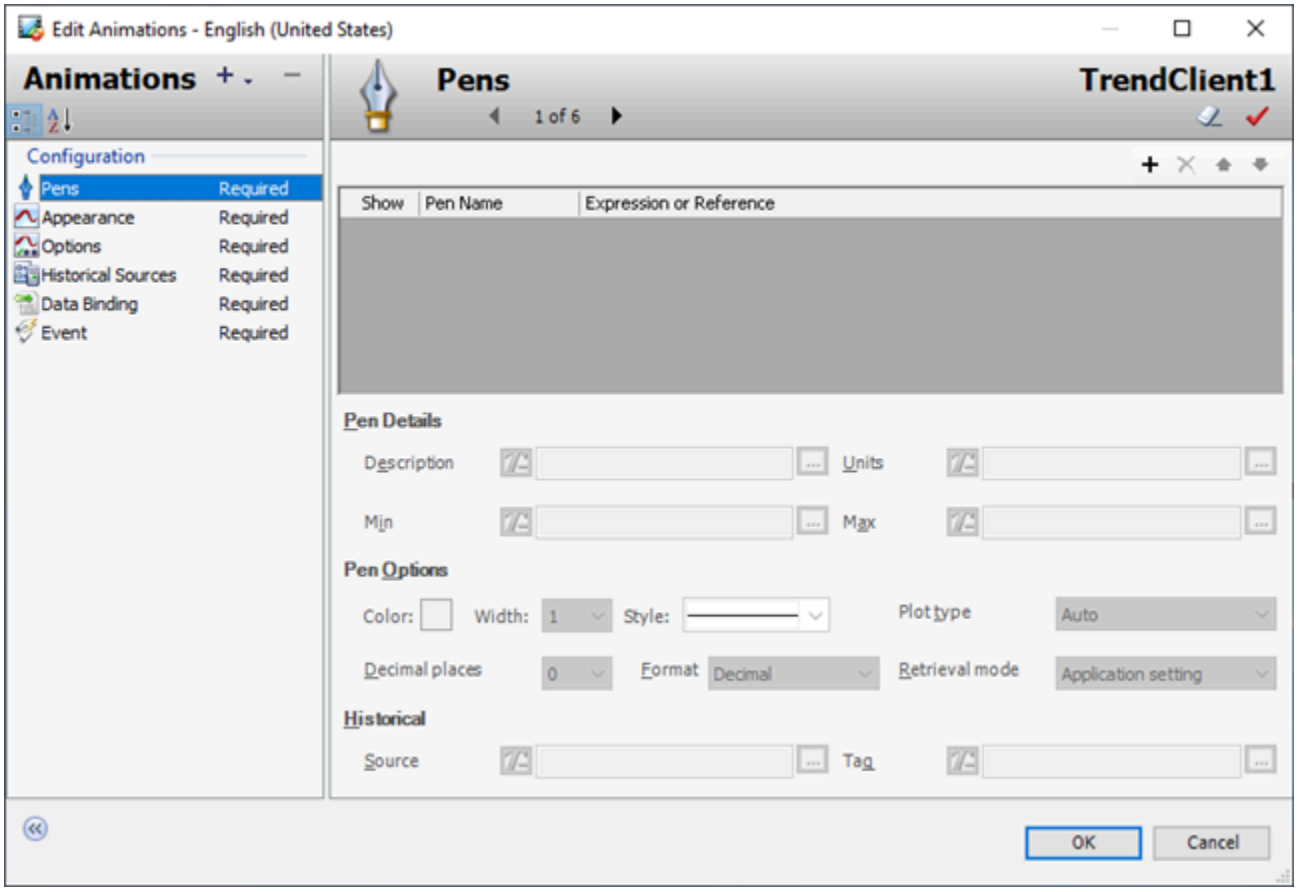
- In the configuration panel, click the **Clear animation** icon.

Add a pen

When you first add a Trend Client to the canvas, no pens are defined. To show a trend, you must add at least one pen and configure it so that the system can identify what trend information to chart. Each pen must have an associated expression or reference.

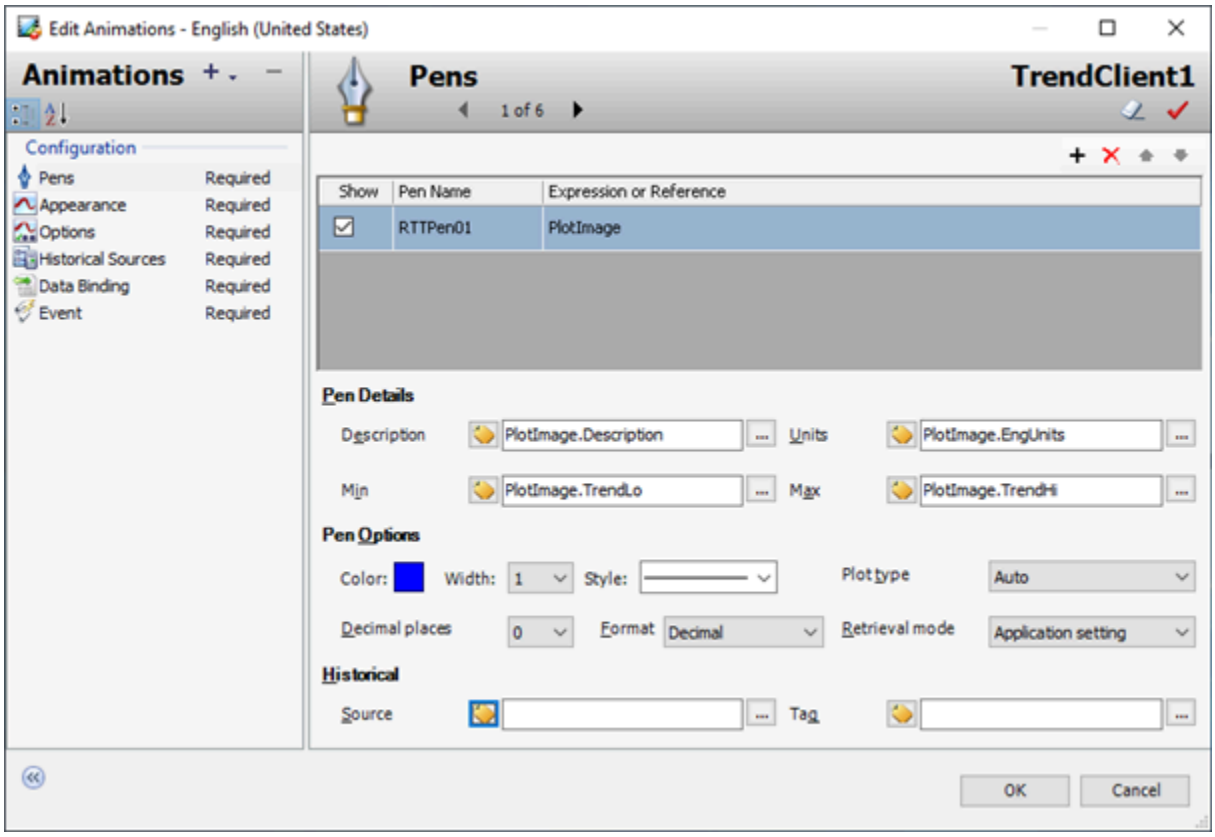
To add a pen

1. In the Industrial Graphic Editor, double-click the Trend Client. The **Edit Animations** dialog box appears with the pen configuration information in the right pane.



2. Click the **Add** icon on the right side of the **Edit Animations** dialog box.

A new pen appears in the configuration panel.

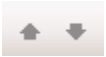


3. In the row for the pen, configure the information for the new pen.

Show	Select the Show box to show the pen in the trend.
Pen Name	Type the name of the pen. (If you leave this blank, the pen's expression is used as the chart label.)
Expression or Reference	Type an expression or reference. Click the ellipsis button to show the Galaxy Browser.

To reorder pens

- Click the Up/Down icons to reorder the pens you have added.



Connect a pen to an InTouch tag

You can connect the pen to an InTouch tagname through the **Expression or Reference** box. The InTouch tagname provides values at run time that control the animation and appearance of the pen.

This can be done by:

- Configuring a reference with the **intouch:tagname** syntax.
- Using a custom property and configuring the custom property in the embedded Industrial graphic in InTouch to reference an InTouch tag. For more information, see the InTouch HMI and ArchestrA Integration Guide.
- Configuring an application server attribute reference to the managed InTouchViewApp object that contains the InTouch tagnames as attributes. The InTouchViewApp object uses the functionality of an InTouchProxy object.
- Configuring an application server attribute reference to an InTouchProxy object that contains the InTouch tagnames as items. This is a special case of configuring an application server attribute reference.

To be able to browse for InTouch tags, you must first:

- Create a managed InTouch application by deriving an InTouchViewApp template and configuring it in WindowMaker.
- Derive an instance of the InTouchViewApp derived template.

The InTouch tags are represented by attributes of the InTouchViewApp object instance.

You may not always be able to reference the local node if you use this method of connecting to the InTouch HMI software. As a result, you may have problems with some applications. If this is the case, use the **intouch:tagname** syntax.

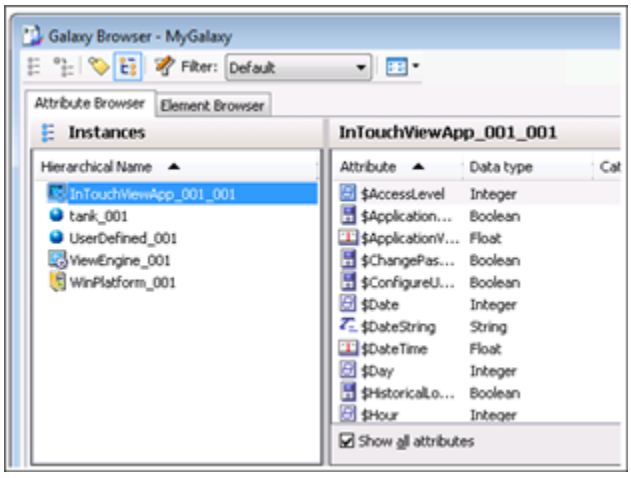
With the **intouch:tagname** syntax, the pen connects to the InTouch tag on the symbol's node. There are some restrictions on how you can use this syntax:

- Unlike in Application Server, you cannot use TRUE and FALSE as Boolean values. Use 1 and 0 instead.
- If you want to connect to an InTouch SuperTag with this syntax, use the following syntax instead:

```
attribute("intouch:SuperTag\Member")
```

To connect Trend Client to InTouch tags

1. In the Galaxy Browser, select the InTouchViewApp object that corresponds to the managed InTouch application. The right panel shows the InTouch tags.



2. Select a tag and click **OK**. The selected reference to an InTouch tag appears in **Expression or Reference**.

Configure pen details and options

After you add a pen and configure the expression or reference for the pen(s), you can set details and options for how the pen shows information on the trend chart. Pen details include the description, units, minimum and maximum for the pen. Pen options include the pen color, width, style, plot type, number of decimal places, format, and retrieval mode.

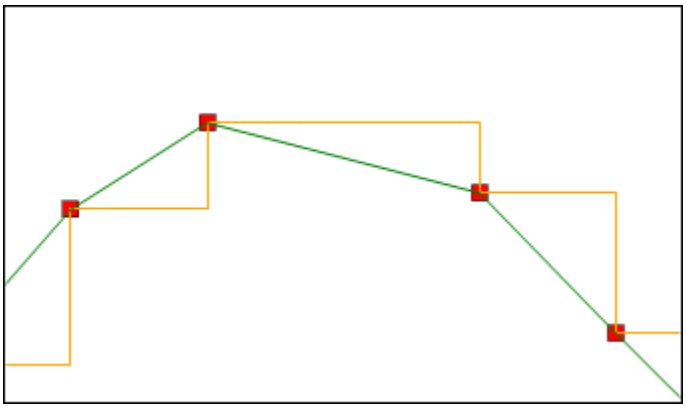
The default values for the pen details are formed by appending **.Description**, **.EngUnits**, **.TrendLo**, and **.TrendHi** to the pen expression. For example, if the pen expression is **Me.PV** for a field attribute named **PV** created on the user defined object, the default pen details are **Me.PV.Description**, **Me.PV.EngUnits**, **Me.PV.TrendLo**, and **Me.PV.TrendHi**. These are generally the best expressions to use for historized attributes. (If the expression is empty or invalid, the Trend Client uses default values of 0 and 100 for the minimum and maximum at run time.)

Note: For some input boxes on the configuration screens, you can specify if the configuration is a static value or a reference by setting the input mode. An input mode icon appears to the left of these boxes. Use static mode input to specify a literal static value such as "Temperature" or 3.141. Use reference mode to specify a reference to an attribute or symbol property such as Tank_001.PV. You can enclose static string values with double-quotes (such as "Description:" or "+Tank_001.Desc") with or without references in Reference mode. When you are in reference mode, you can also click the ellipsis button to the right of the box to show a selection tool such as the Galaxy Browser or the Tag Picker.

A line curve (plot type **Line**) is best suited for charting continuously-changing analog data. A step-line curve (plot type **Step**) is best suited for discrete data and box for analog data that is not continuous. When you select a plot type of **Auto**, the curve type is determined as follows:

- For tags retrieved from Historian 9.0 (or higher), the type is based on the tag's effective interpolation setting. Tags that use stair-step interpolation are trended as a step line, and tags that use linear interpolation are trended as a line.
- For all other tags, the curve type is based on the tag type: step line for integer tags, and line for real tags.

The following illustration shows the same data drawn using each type of curve. The line curve is shown in green, the step line curve is shown in orange, and the point curve is shown in red.



To configure the options for a pen

1. Select the pen you want to configure.
2. In the **Pen Details** area, configure what the pen shows.

Description	The pen’s description. The expression is evaluated and the result is used for the pen's description in the pen list (the legend) below the chart itself.
Units	The unit to chart in. This is the plot scale, rather than an instrument scaled from a percentage to engineering units.
Min	Type the minimum for the pen’s range.
Max	Type the maximum for the pen’s range.

3. In the **Pen Options** area, configure how the curve looks in the chart for the selected pen.

Color	The line color of the pen. Click the colored square to select the color from a palette or define a custom color.
Width	The thickness of the trend curve. Valid values are 0 through 10.
Style	The style of the trend curve; for example, a solid or dashed line.
Plot type	The type of trend curve to draw. Options are **Line**, **Step line**, **Point**, and **Auto**.
Decimal Places	The number of decimal places to show for the data value. This applies only to analog tags. Valid values are 0 through 15.
Format	The way the values for the pen appear, either in decimal format or scientific format.
Retrieval Mode	The data retrieval mode, either **Cyclic**, **Full**, or **Application Setting**.

Configure pens for historical sources and tags

When you start the Trend Client at run time with historical data and then plotting real-time data, you can configure a pen to initialize with a specific historical source and tag. For example, MyEngine.Engine.Historian.Connection is a common relative reference expression for the historical source on a template and, for a pen expression of Me.PV, Me.TagName.PV is a common relative reference expression for the historical tag.

If the historical tagname needs to use relative references, construct the string expression for the historical tagname that evaluates the relative part of the name and append a static part. For example, to reference a UDA named PV, use Me.Tagname+".PV" or a similar expression. You can define this expression on a symbol in a $Reactor template. There are instances of Historian tagnames are Reactor1.PV and Reactor2.PV. If you use the

expression in the form Me.PV, it resolves to the current value instead of the tagname. For example, if PV is associated with a temperature, Me.PV would return a temperature rather than the string for the tagname.

You can also use valid relative references such as Me, MyContainer, MyArea, MyEngine, and MyPlatform when configuring the historical tag-names as relative references. For example, to reference the UDA named "Level" in a user-defined object named $Reactor, you can configure the historical tagname as an expression: Me.TagName+".Level". At run time, when the object "Reactor_001" is viewed, the expression is resolved to "Reactor_001.Level" as the historical tag name.

To configure a pen for a historical source and tag

1. Select the pen you want to configure.
2. In the **Historical** area, configure the source and tag for the pen.

Source	The historical source to use for the pen. This history source must already be configured in the **Historical Sources** pane. For more information, see "Configure Historical Sources". If this is set to reference mode, click the ellipsis button to show the Galaxy Browser and use that to find the expression or reference.
Tag	The historical tag to use for the pen. If this is set to reference mode, click the ellipsis button to show the Tag Picker. For more information, see "Use the Tag Picker".

Note: To use the reference mode, you must have already configured a historical data source for the Trend Client. For more information, see "Configure Historical Sources".

Configure historical sources

To use a Trend Client to chart historical data, you must connect to a data source. You can connect to a Historian or an InTouch LGH file.

You can use one of the following authentication modes when connecting to AVEVA Historian:

- Windows Integrated Security (uses the running process credentials)
- Alternate Windows account (You can use the credentials stored in the Credential Manager. You can also specify the domain name, user name, and password).

  The alternate credentials are used to do a Windows impersonation before connecting to the database using Windows Integrated Security.
- SQL Server Authentication

  You can use the credentials stored in the Credential Manager. You specify the user name and password of a SQL Server account. These credentials are used to log in to the database.

  Note: Ask your administrator what type of user account you must use to access the server.

Server connections are specific to the instance of the client and must be re-defined for each instance.

When you start a Application Server application, you are not automatically logged on to every server that you configured before. You are only logged on to a server when a pen is configured to use that server.

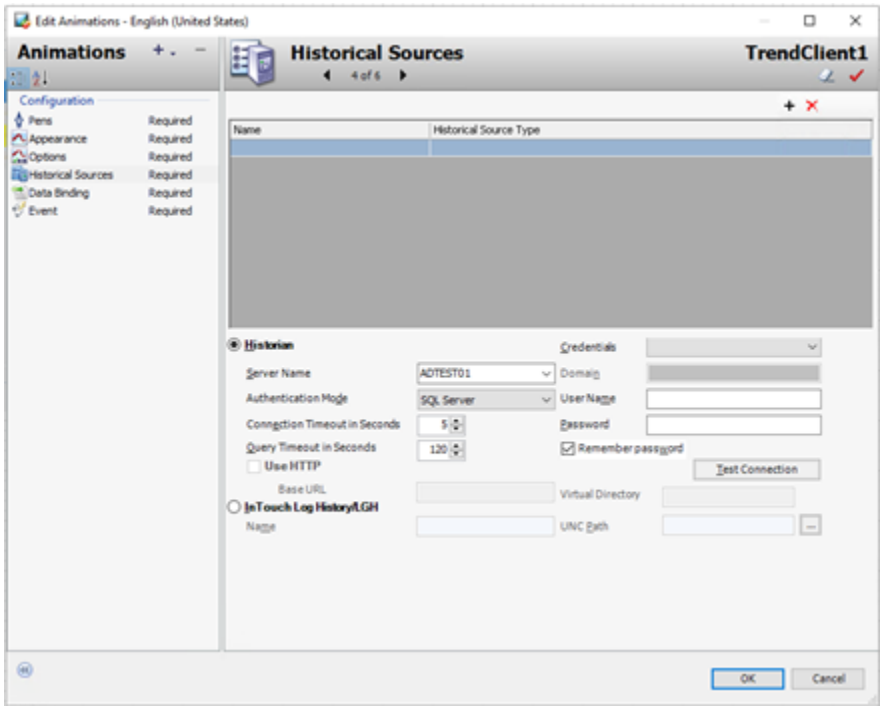
You need not run as an administrative user to operate in run time within an InTouch application. The user access control is enabled for you on all supported Windows operating systems.

Create a Historian connection

To create a Historian connection, you need your assigned Historian user name and password.

To create a Historian connection

1. In the Industrial Graphic Editor, double-click the Trend Client. The **Edit Animations** dialog box appears
2. Click **Historical Sources**. The **Historical Sources** screen appears in the right pane.



3. Click the **Add** icon on the right side of the **Edit Animations** dialog box.

   A new server row appears in the configuration panel.
4. In **Server Name**, type the name of the server to which you want to connect. You can select from a list of the servers by clicking the arrow to the right of the box.
5. In **Authentication Mode**, select one of the following authentication modes: Windows Integrated, Windows Account, or SQL Server.
   - Windows Integrated to use the authentication of the currently logged-on Windows user
   - Windows Account to use a given Windows user authentication
   - SQL Server to use SQL Server authentication mode
6. From the **Credentials** drop-down, select the credentials for authentication.The Credentials field is enabled only when you select **Windows account** or **SQL Server** authentication type. The Windows Integrated

authentication method uses the credentials of the user currently logged in, and disables the Credentials field.

Note:

- For standalone InTouch applications, the credentials are retrieved from the Application Manager. For managed InTouch applications, the credentials are retrieved from the Credential Manager of the Application Server. For more information, see Work with Credential Manager in the AVEVA™ InTouch HMI Application Development Guide.

- By default, all named credentials are added to the Administrators group. In addition, we recommend you to add groups that the users running the application belong. This will allow the users to access the credentials even when running the application in a non-administrator mode, such as, viewing the historical data or trends in WindowViewer.

You can also select **Use username and password** to enter the credentials.

- If you are using Windows Account authentication mode, type the domain, user name, and password in the Domain, User Name and Password boxes.
- If you are using SQL Server authentication mode, type user name and password in the User Name and Password boxes.

where:

Domain - The domain name in which your Windows account is validated.

If you are logging on to the server using Windows Account or SQL Server credentials, configure the following login details:

User Name - Your assigned Historian user name. If your system administrator has not assigned you a user name and password, you may use one of the default user accounts, which are automatically configured during a typical Historian installation.

Password - The password that is associated with the user name. Select **Remember password** to have the system remember your password.

7. In **Connection Timeout in Seconds**, type the connection timeout in seconds. Valid values are 1 to 600.
8. In **Query Timeout in Seconds**, type the query timeout in seconds. Valid values are 1 to 600.

   Note: You can test the connection to the server with the information you have entered by clicking **Test Connection**. If necessary, you can edit the information before adding the server to the list.
9. Click **OK**.

   Note: The Trend Client does not validate the connection when you click OK.

Create an InTouch LGH connection

You can get data from InTouch LGH files by connecting to them.

To create an InTouch LGH connection

1. In the Industrial Graphic Editor, double-click the Trend Client. The **Edit Animations** dialog box appears
2. Click **Historical Sources**. The **Historical Sources** screen appears in the right pane.

   
3. Click the **Add** icon on the right side of the **Edit Animations** dialog box. A new connection entry appears in the configuration panel.

4. Select **InTouch Log History/LGH.**
5. In **Name**, type the InTouch HMI software server.
6. In **UNC Path,** type the location of the InTouch LGH file using a UNC path name or a mapped drive path.
7. Click **OK**.

Note: The Trend Client does not validate the connection when you click OK.

Edit a server connection

You can edit an existing server connection.

To edit a server connection

1. On the **Historical Sources** screen, select the server you want to edit.
2. Edit the details for the server.
3. Click **OK**.

Remove a server connection

You can remove a server connection you no longer need. After you delete a server, you cannot undelete it.

To remove a server connection

1. On the **Historical Sources** screen, select the server you want to delete.
2. Click the **Delete** icon on the right side of the **Edit Animations** dialog box.

   The server is deleted.

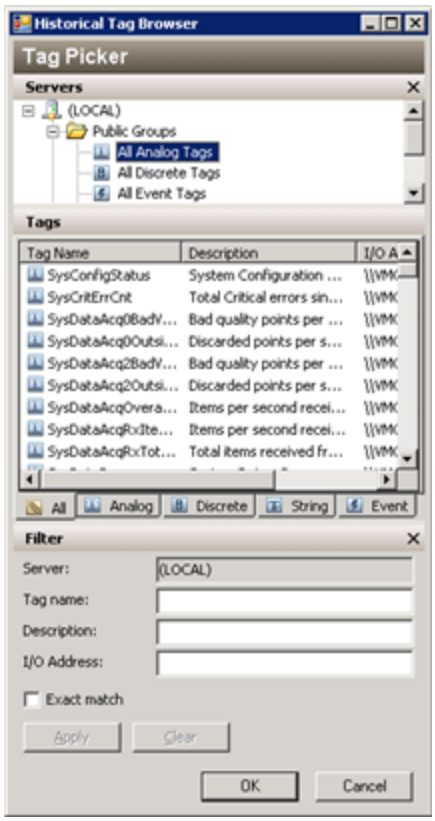
The Tag Picker

The Tag Picker shows which tag groups and tags exist in the Historian database. It shows all tags that are visible to the currently logged-on user based on his permissions.

Using the Tag Picker, you can quickly search the database for tags of a certain type or for tags that match a particular search pattern. You can then select the ones you want to include in the Trend Client. You cannot select a tag from an LGH file.

The Tag Picker includes the following panes:

- Servers pane
- Tags pane
- Filter pane



For instructions to import the Tag Picker control and embed it in a symbol, see "Chapter 12 Using Client Controls" of the *Creating and Managing Industrial Graphics User's Guide*.

The Servers pane

The **Servers** pane shows a list of Historian folders. The **Servers** pane allows you to navigate through the folder structure (namespace) of one or more Historians and select a group (folder) of tags.

The **Servers** pane shows the following items:

Category	Description
Servers	All objects that make up the basic Historian system, such as tags, I/O Servers, defined engineering units, storage locations, and so on.
Public Groups	All objects that are visible to all clients. If you have administrative permissions, you can create, rename, and delete groups in the public groups folder.
Private Groups	All objects that are visible to the user that is currently logged on. Users can create, rename, and delete groups in the private groups folder.

Show or hide the Servers pane

To show the Servers pane

- Right-click in the **Servers** pane and then click **Servers pane** so that a check mark appears.

To hide the Servers pane

Do one of the following:

- Right-click in the **Servers** pane and then click **Servers pane** so that no check mark appears.
- Click **Close**.

Add a group

You can add groups just as you would add a new folder in the Windows Explorer. For example, you can create the BoilerTags group under in the existing Private Groups group. You can also delete, cut, copy, paste, and drag objects from one folder to another.

To add a group

1. Right-click on the folder under which you want to create a group and then click **New Group**.

   A new folder appears in the Tag Picker.
2. Type a name for the folder and press **ENTER**.

Add a tag to a group

When you add tags to a new group, the original reference still appears in the default system group. Any tag can belong to any number of groups, and any group can contain any number of tags.

To add a tag to a group

1. Select the system group folder that contains the tag that you want to add to your new group.
2. In the **Tags** pane, select the tag to add.

3. Do any of the following:
   - Drag the tag from the **Tags** pane into the folder.
   - Use the **Copy** and **Paste** commands on the **Edit** menu to copy the tag to the target folder.
   - Right-click on the tag in the **Tags** pane. Use the **Copy** and **Paste** commands in the shortcut menu to copy the tag to the target folder.

Delete a group or tag reference

When you delete a private group or a tag reference in a private group, the group folder, any subfolders that the group folder may contain, and all references to tags are deleted. The tags themselves are not deleted, and the original references still appear in the default system group. You cannot delete public folders or the tag references contained in them.

To delete a group or tag

1. Select the group or tag in the pane.
2. Do one of the following:
   - Right-click on the group or tag and then click **DELETE**.
   - Press **DELETE**.

Rename a group

You can rename a group that you have created in the Tag Picker. However, you cannot rename a public folder.

To rename a group

1. Select the group in the pane.
2. Do one of the following:
   - Right-click on the group and then click **RENAME**.
   - Press **F2**.
3. Type a new name for the group and press **ENTER**.

View server details

You can view information such as the version number, time zone, and security mode for any Historian server in the **Servers** pane.

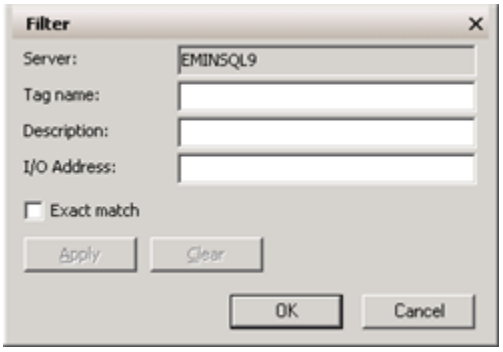
To view server details

1. In the **Servers** pane, right-click on a Historian server and then click **Server details**. The **Server Details** dialog box appears.

2. Click **OK**.

Filter the tags list

Use the **Filter** pane to reduce the tags listed in the **Tags** pane according to criteria that you specify. You can filter the tags according to name, description, and I/O address.



The filter mechanism allows for the following wildcard characters as part of the filter criteria:

Wildcard Character	Filter Function
%	Any string of zero or more characters.
_	Any single character.
[ ]	Any single character within the specified range or set. For example: • [a-f] • [abcdef]

Wildcard Character	Filter Function
[^]	Any single character not within the specified range or set. For example: • [^a - f] • [^abcdef]

For example, to find all tagnames ending with "level", type **%level**. Filter criteria are not case-sensitive.

When the **Servers** pane and the **Filter** pane are both visible, the filter conditions apply to the selected group in the **Servers** pane. When the **Servers** pane is hidden, the filter applies to all of the tags for the selected Historian.

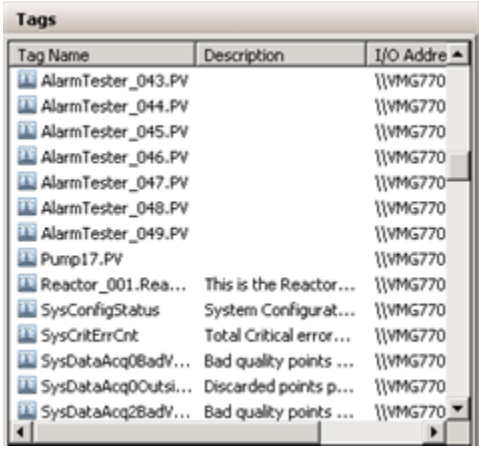
To apply a filter

1. In the **Server** box, select the server.
   This box is not available if the **Servers** pane is visible.
2. In the **Tag name** box, type the string to match for the tagname.
3. In the **Description** box, type the string to match for the description.
4. In the **I/O Address** box, type the string to match for the I/O address.
5. Select the **Exact match** check box to search for tags that exactly match the entire string that you provided for the tagname and description options.
   For example, if you specify "level" as the tagname and do not select **Exact match**, any tagname that contains the string "level" appears. For example, "**ReactLevel**," "**ProdLevel**," and **"$AccessLevel**."

   Note: The **Exact match** option does not apply to the I/O address.
6. Click **Apply** to apply the filter criteria.
7. Click **Clear** to clear the **Filter** pane.

The Tags pane

The **Tags** pane shows all the tags for the currently selected group in the **Servers** pane.



To select multiple tags in the list, hold **Ctrl** and/or **Shift** while clicking.

To view only tags of a certain type, click the appropriate tab at the bottom of the pane.

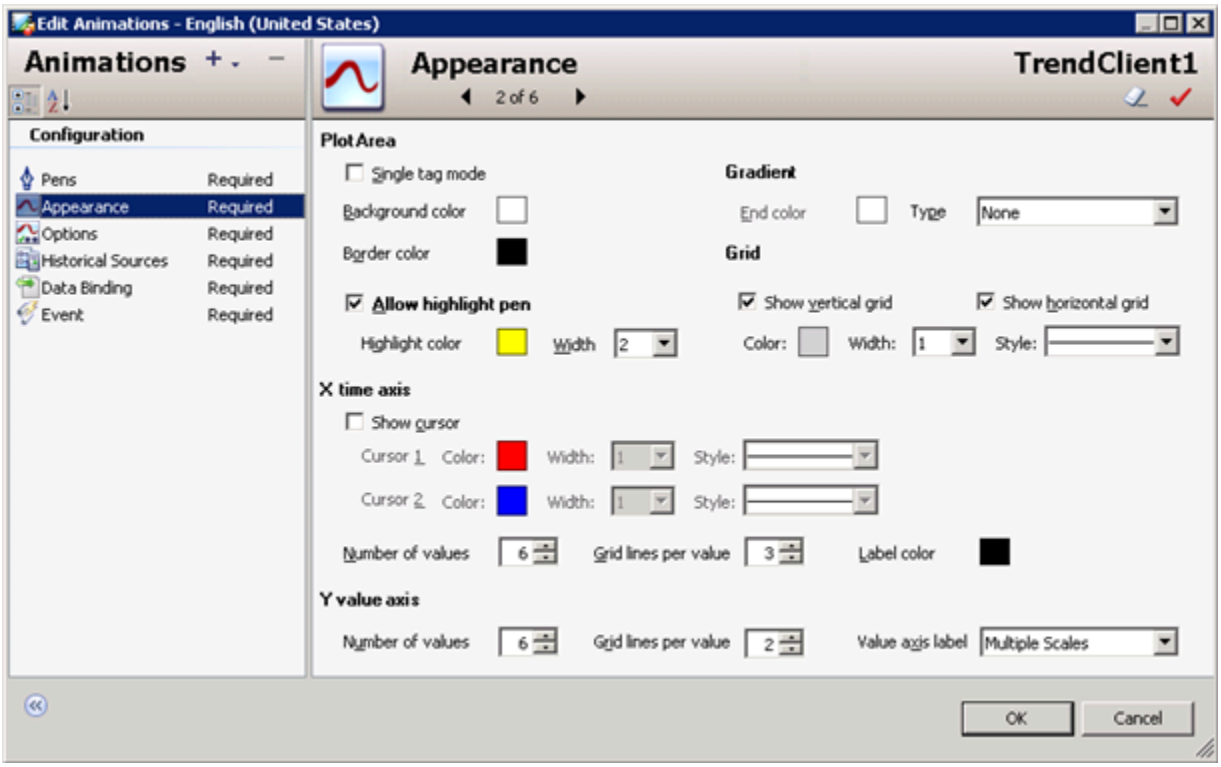
To sort the table by a particular column, click the column heading.

Configure the trend appearance

You can set a number of general options for the trend appearance, including gradients, border and plot area background color, highlights, and colors for the X and Y axes.

To configure the trend appearance

1. In the Industrial Graphic Editor, double-click the Trend Client. The **Edit Animations** dialog box appears
2. Click **Appearance**. The trend appearance information appears in the right pane.



3. Select **Single tag mode** to set the trend to single tag mode.

   When you initially create a tag list for a trend, all the tags are included in the display. Setting the trend to single tag mode allows you to exclude all tags but one from appearing in the trend chart, without removing them from the tag list.

   Clear **Single tag mode** to view multiple tags again in the chart.
4. Click **Background color** to configure the main color of the background of the plot area. If you are using a gradient fill, this is the starting color for the gradient.
5. Click **End color** to select the ending color for the gradient. The gradient starts with the main color and fades to the gradient end color.
6. Click **Type** to specify the starting point for the flow of the gradient. Valid values are **LeftRight**, **TopBottom**, **Center**, **DiagonalLeft**, **DiagonalRight**, **HorizontalCenter**, and **VerticalCenter**. For example, if you select green as main color, white as the gradient end color, and center as the gradient type, the center of the chart is green and fades to white towards the surrounding edges.

7. Click **Border Color** to configure the color of the border for the entire chart area.
8. Select **Allow highlight pen** to highlight the pen. Configure the color and width to be used for pen highlighting.

Highlight color	Click to select or configure a color for highlighting the pen curve.
Width	Specify the width (in pixels) of a highlighted curve.

9. In the **Grid** area, configure the grid options.

Show horizontal grid	Check to show the horizontal grid.
Show vertical grid	Check to show the vertical grid.
Color	Click to select or configure a color.
Width	The width, in pixels, of the border line.
Style	The style of the border line.

10. In the **X time axis** area, configure the properties for the horizontal axis.

Show horizontal cursor	Check to show the horizontal cursor.
Cursor 1 color	Click to select or configure the color for time axis cursor 1.
Width	The width of time axis cursor 1.
Style	The line style of time axis cursor 1.
Cursor 2 color	Click to select or configure the color for time axis cursor 2.
Width	The width of time axis cursor 2.
Style	The line style of time axis cursor 2.
Number of values	The number of values that are shown along the time axis. The values are shown at evenly-spaced points along the axis. The number of values remain the same even if you zoom in and out. The valid range is from 2 to 15, with a default of 6.
Grid lines per value	The number of grid lines that appear between each pen value plotted on the chart. The valid range is from 1 to 20, with a default of 3.
Label color	The color of the X axis label.

11. In the **Y value axis** area, configure the properties for the vertical axis.

Number of values	The number of values that are shown along the value axis. The timestamps are shown at evenly-spaced points along the axis. The number of values remain the same even if you zoom in and out. The valid range is from 2 to 15, with a default of 6.
Grid lines per value	The number of grid lines appearing between each pen value that is plotted on the chart. The valid range is from 1 to 20, with a default of 2.
Value axis label	The label of the value axis.

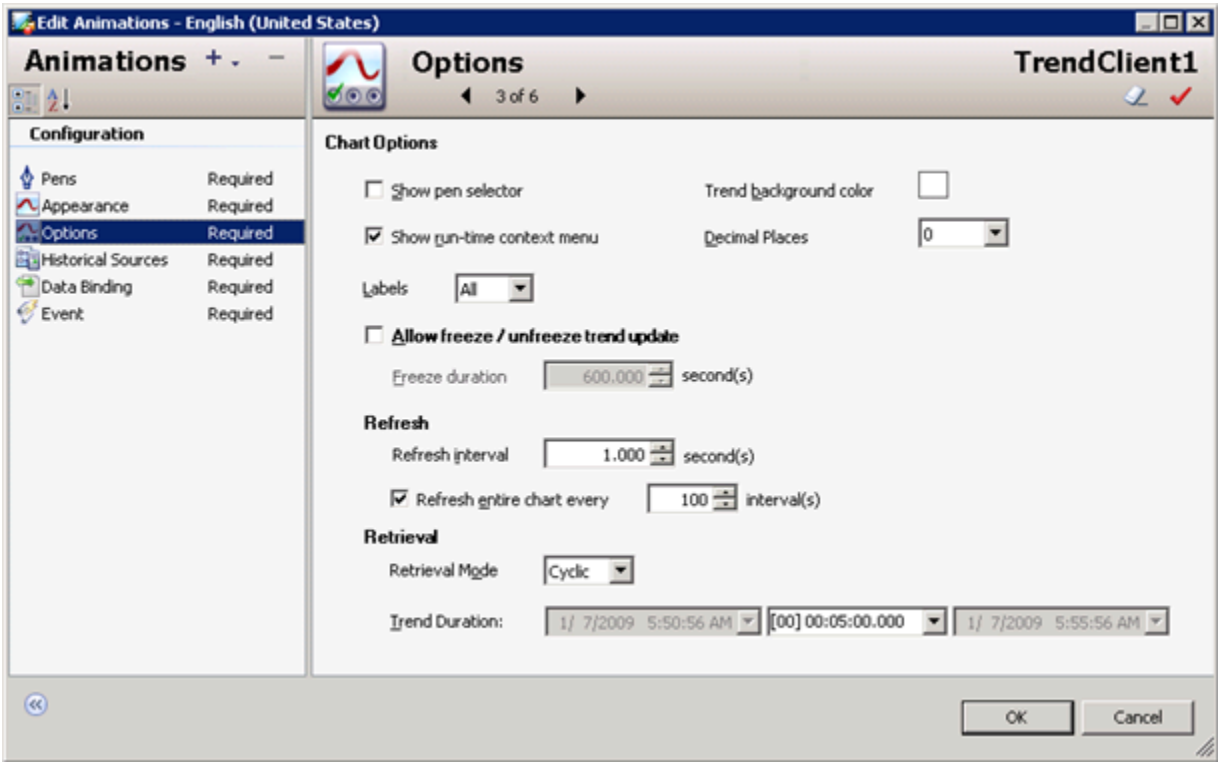
12. Click **OK**.

Set chart options

You can set a number of general chart options, including the trend background color, freeze options, refresh interval, and retrieval options.

To set the chart options

1. In the Industrial Graphic Editor, double-click the Trend Client. The **Edit Animations** dialog box appears.
2. Click **Options**. The options information appears in the right pane.



3. Select the **Show pen selector** check box to show the legend or list of pens at the bottom the trend.
4. Click **Trend background color** to configure the main color of the background of the entire chart area.
5. Select **Show run-time context menu** to show the context menu when you right-click the trend at run time.

6. Set **Decimal Places** to the number of decimal places to use in the trend.
7. Set **Labels** to All to display the chart label, X and Y axis scales, and cursor information, or None to suppress the labels in the plot area
8. Select **Allow freeze/unfreeze trend update** to enable Freeze on the run-time context menu so you can freeze the trend chart from live updates. When this option is selected (the default), you can also set the freeze duration. The freeze duration determines how long the chart remains frozen when you freeze the chart at run time. After the freeze time has elapsed, the chart resumes the live updates. If the freeze duration is set to 0, you must manually unfreeze the chart.
9. Set **Refresh interval** to the number of seconds between chart updates. You can optionally select **Refresh entire chart** to refresh the entire chart after the number of refresh intervals you specify.

   Note: Optimal performance for the Trend Client can be observed when, the default values for Refresh Interval is set at 1 second and Refresh entire chart every is set at 100 intervals. The Trend Client is refreshed as a combination of both values and if the refresh rate is faster than the rate at which data is retrieved, data may not be plotted correctly.
10. Set **Retrieval Mode** to **Full** or **Cyclic**. Full retrieval mode returns all stored data points for a tag within a trend period. Cyclic retrieval mode returns stored data at equal length intervals within a trend period. For more information, see Appendix A, "Understanding Data Retrieval".

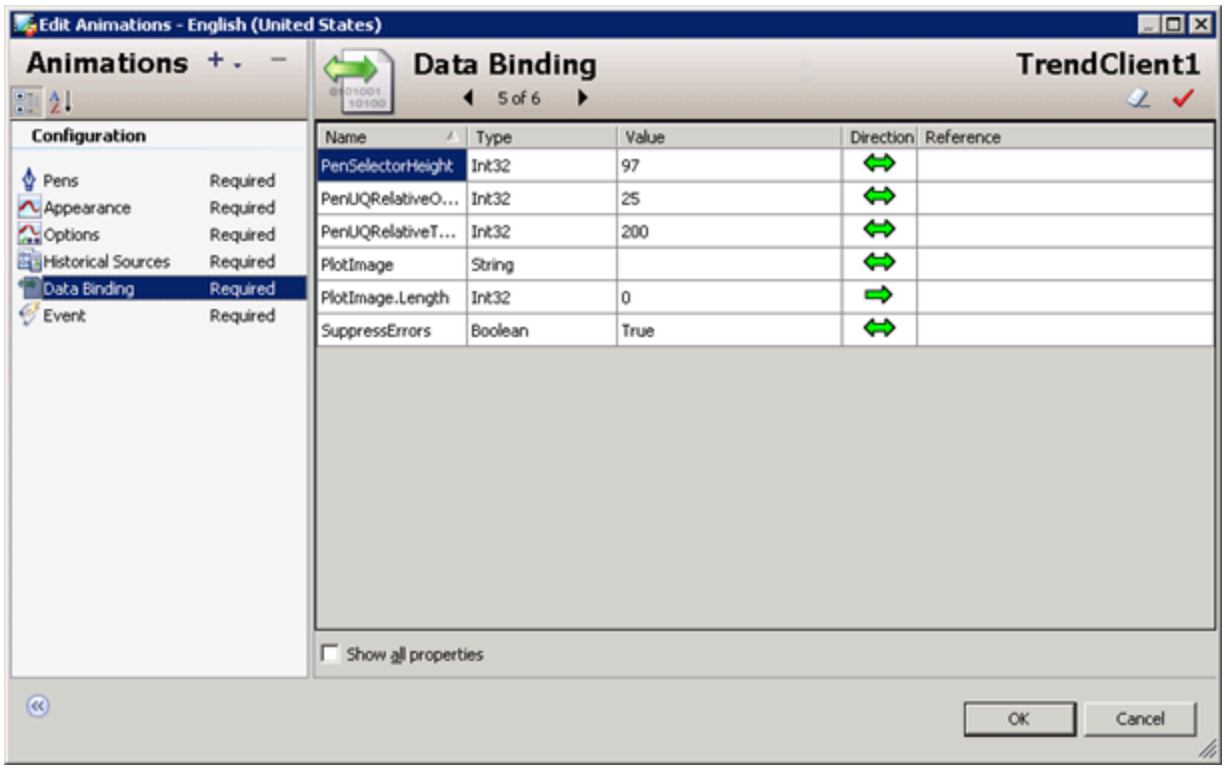
    Note: Retrieval mode is applicable for tags from Historian. The value set here is the default setting for a pen; however, you can override it by setting the pen retrieval mode to any value other than application setting.
11. Set **Trend Duration**. You can set the trend duration for a maximum period of 2 years. All trend durations are calculated backward from an end date of the current time (except for the options "Yesterday" and "Previous Hour"). You can pick a duration from the list or type a specific duration in the list box. The time notation is [DD] HH:MM:SS.fff, where DD=days, HH=hours, MM=minutes, SS=seconds, and fff=milliseconds. To change a value in the duration list box, use the arrow keys on your keyboard to position the mouse cursor to the left of the number you want to change, and then type the new number. You can only change the duration and not the start and end time.
12. Click **OK**.

Set data bindings

You can associate a reference to an attribute with the Data Binding pane instead of with scripting. Information can be passed in either direction or both directions as desired. Changes to the status of one can affect the other.

To set data bindings

1. In the Industrial Graphic Editor, double-click the Trend Client. The **Edit Animations** dialog box appears.
2. Click **Data Binding**. The data binding information appears in the right pane.



The Data Binding table contains the following read-only information:

Name	The name of the property.
Type	The .NET data type of the property.
Value	The value of the property.

3. Click the **Direction** icon to disable the link entirely or change the direction of the data flow. If the arrow points to the right (away from the name of the property), the status of the Trend Client property affects the attribute you're binding to. If the arrow points to the left (toward the name of the event), the status of the Trend Client property is affected by the attribute you're binding to. If the arrow points both ways, a change in the status of one affects the other.
4. Double-click **Reference** to enter reference information. Click the ellipsis button to open the Galaxy Browser and use the Element Browser. For more information, see Chapter 3, Working with Objects, in the Application Server User Guide.
5. Select **Show all properties** to show all properties for which there is data binding information.
6. Click **OK**.

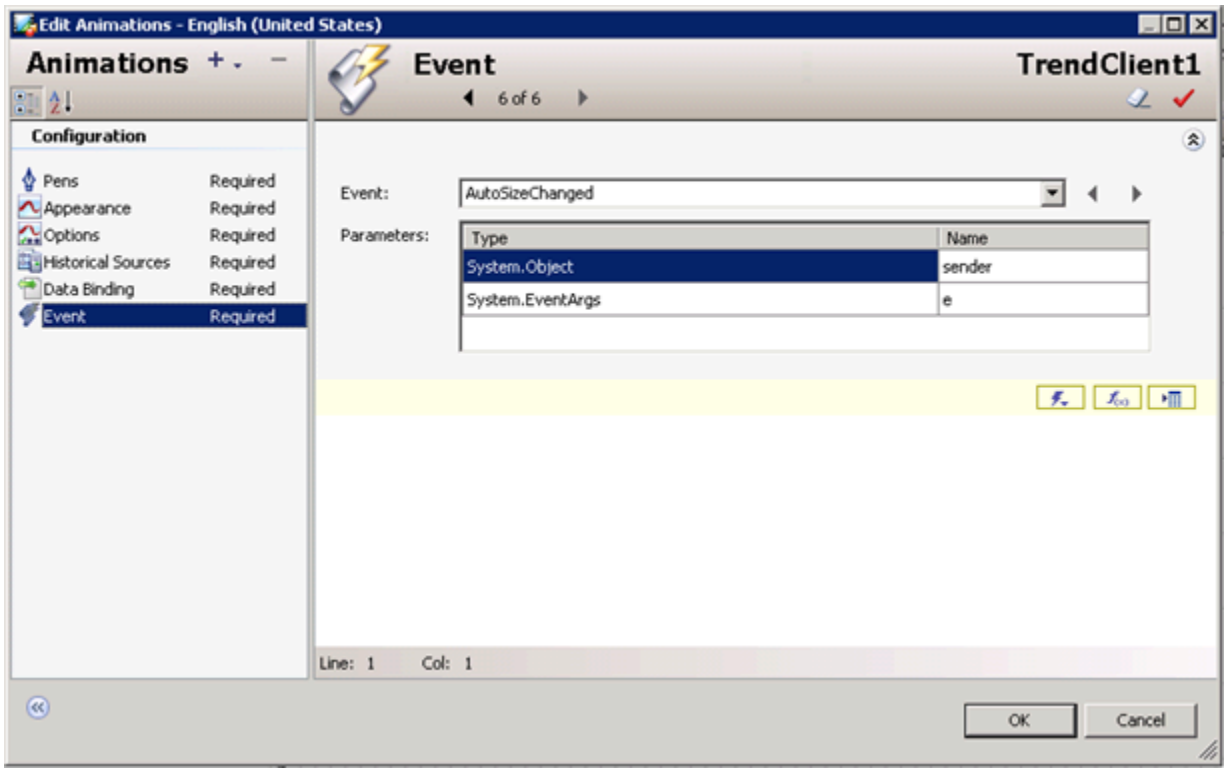
Handle trend events

You can use the custom event editor to build scripts to handle trend events.

To modify a trend event

1. In the Industrial Graphic Editor, double-click the Trend Client. The **Edit Animations** dialog box appears
2. Click **Events**. The events information appears in the right pane. For information on the different types of

events, see Trend Client Events.



3. Select an event in the **Event** list. The parameters information appears in the lower portion of the pane.
4. Type a script for this event in the window. You can use the editor buttons to gather information and parameters.

   Browse Event Parameters inserts the event parameters and any associated event parameter information at the current insertion point in the script.

   Display Script Function Browser shows the available script functions. Select a function and click **OK** to insert the function and parameters at the current insertion point in the script.

   Display Attribute Browser opens the Galaxy Browser.
5. Click **OK**. The script information is saved for the event.

Trend Client scripts

This section describes the properties, methods, and events for the Trend Client.

A Trend Client placed in a frame window supports panning and zooming. For more information on how to configure the ZoomPercent property, refer to the InTouch HMI Application Management and Extension Guide.

Trend Client properties

The following are the properties of the Trend Client:

- Chart.AddMultiplePens
- Chart.BackgroundColor
- Chart.Freeze
- Chart.FreezeDurationMS
- Chart.HidePenList
- Chart.Labels
- Chart.PenPrecision
- Chart.RefreshEntireChartIntervals
- Chart.RetrievalMode
- Chart.UpdateRateMS
- HistorySources
- Pen.Color
- Pen.Count
- Pen.Description
- Pen.Expression
- Pen.Format
- Pen.HistorySource
- Pen.HistoryTagName
- Pen.Index
- Pen.Name
- Pen.Precision
- Pen.RetrievalMode
- Pen.Style
- Pen.TrendHi
- Pen.TrendLo
- Pen.TrendType
- Pen.Units
- Pen.Visible
- Pen.Width
- PenSelectorHeight
- PenUQRelativeOpacity
- PenUQRelativeThickness
- PlotArea.BackgroundColor
- PlotArea.BorderColor
- PlotArea.GradientEndColor

- PlotArea.GradientType
- PlotArea.GridColor
- PlotArea.GridHorizontal
- PlotArea.GridStyle
- PlotArea.GridVertical
- PlotArea.GridWidth
- PlotArea.HighlightCurrentPen
- PlotArea.PenHighlightColor
- PlotArea.PenHighlightWidth
- PlotArea.SingleTagMode
- PlotImage
- ShowContextMenu
- SuppressErrors
- TimeAxis.Cursor1.Color
- TimeAxis.Cursor1.Pos
- TimeAxis.Cursor1.Style
- TimeAxis.Cursor1.Width
- TimeAxis.Cursor2.Color
- TimeAxis.Cursor2.Pos
- TimeAxis.Cursor2.Style
- TimeAxis.Cursor2.Width
- TimeAxis.LabelColor
- TimeAxis.NumGridPerValue
- TimeAxis.NumValues
- TimeAxis.ShowCursors
- TimeSelector
- TimeSelector.DurationMS
- TimeSelector.EndDate
- TimeSelector.StartDate
- TimeSelector.TimeDuration
- ToolTipText
- TrendVersion
- ValueAxis.Label
- ValueAxis.NumGridPerValue
- ValueAxis.NumValues
- Visible

This section describes all the scriptable properties available in the Trend Client.

You can access the properties of an individual pen by setting it as the current pen and then applying the pen

properties.

Chart.AddMultiplePens

The Chart.AddMultiplePens is a read-write Boolean property that suspends or enables a chart refresh while multiple pens are added to the chart.

Syntax

```
Chart.AddMultiplePens = bool;
Result = Chart.AddMultiplePens;
```

Remarks

The default is FALSE (allows the chart refresh while multiple pens are added to the chart).

You can set this property to TRUE, then add multiple properties using a script without refreshing the graph. After adding the final property value, the script returns this property to FALSE. The graph is automatically refreshed and shows all pens that have been added.

Chart.BackgroundColor

The Chart.BackgroundColor property is a read-write property that gets or sets the background color of the chart.

Syntax

```
Chart.BackgroundColor = color;
Result = Chart.BackgroundColor;
```

Remarks

The default is white.

Color is a .NET Framework data type. You can use various color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName(). For a list of the .NET color names and the corresponding hexadecimal codes, see .NET Colors. For more information on the color methods, see the Microsoft documentation for .NET Framework development.

Chart.Freeze

The Chart.Freeze property is a read-write Boolean property that stops (TRUE) or starts (FALSE) the trend display from updating.

Syntax

```
Chart.Freeze = bool;
Result = Chart.Freeze;
```

Remarks

The default is FALSE, which continues (or resumes) the trend display updates. Setting the property to TRUE stops the trend display updates.

Chart.FreezeDurationMS

The Chart.FreezeDurationMS property is a read-write integer property that freezes the chart from live updates for the specified duration in milliseconds.

Syntax

```
Chart.FreezeDurationMS = int;
Result = Chart.FreezeDurationMS;
```

Remarks

The default is 600000 (10 minutes).

Note: When freeze duration is set to a value other than 0, in run time after a freeze has been initiated, live updates appear on the Trend Client automatically after the configured time has elapsed. If Freeze Duration for chart is set to 0, the auto resume feature is disabled.

Chart.HidePenList

The Chart.HidePenList property is a read-write Boolean property that shows or hides the pen list in the chart.

Syntax

```
Chart.HidePenList = bool;
Result = Chart.HidePenList;
```

Remarks

The default is TRUE, which implies that the chart is hidden.

Chart.Labels

The Chart.Labels property is a read-write integer property that sets the visibility of labels (chart label, X and Y axes scales, and cursor information) in the chart. Supported options are All and None (where 0 = All and 1 = None).

Syntax

```
Chart.Labels = int;
Result = Chart.Labels;
```

Remarks

The default is 0 (All).

Chart.PenPrecision

The Chart.PenPrecision property is a read-write integer property that gets or sets the number of decimal places to show by default for the data value.

Syntax

```
Chart.PenPrecision = int;
Result = Chart.PenPrecision;
```

Remarks

The default is 0.

Chart.RefreshEntireChartIntervals

The Chart.RefreshEntireChartIntervals property is a read-write integer property that refreshes the entire chart at every specified number of intervals. (The interval is specified with the Chart.UpdateRateMS property.) If the value is '0' the chart only adds new data to the chart; however, it does not refresh the data already present on the chart.

Syntax

```
Chart.RefreshEntireChartInterval = int;
Result = Chart.RefreshEntireChartIntervals;
```

Remarks

The default is 100 intervals.

Chart.RetrievalMode

The Chart.RetrievalMode property is a read-write integer property that gets or sets the data retrieval mode for retrieving the data from IndustrialSQL Server at the chart level. Supported options are cyclic and full (where 0 = cyclic and 2 = full).

Syntax

```
Chart.RetrievalMode = int;
Result = Chart.RetrievalMode;
```

Remarks

The default is 0 (cyclic).

Chart.UpdateRateMS

The Chart.UpdateRateMS property is a read-write integer property that gets or sets the chart refresh interval in milliseconds.

Syntax

```
Chart.UpdateRateMS = int;
Result = Chart.UpdateRateMS;
```

Remarks

The default is 1000.

HistorySources

The HistorySources property is a read-only object property that gets a list of history sources. For more information about the HistorySources properties, see "Trend Client History Sources".

Syntax

```
Result = HistorySources;
```

Example

```
Dim histSources = TrendClient1.HistorySources;
Dim histSource = histSources.GetSource("<history source name>");
```

Remarks

No default.

Pen.Color

The Pen.Color property is a read-write property that gets or sets the color of the currently selected pen.

Syntax

```
Pen.Color = color;
Result = Pen.Color;
```

Remarks

The default is color.

Color is a .NET Framework data type. You can use various color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName(). For a list of the .NET color names and the corresponding hexadecimal codes, see .NET Colors. For more information on the color methods, see the Microsoft documentation for .NET Framework development.

Pen.Count

The Pen.Count property is a read-only integer property that gets the number of the pens in the pen list.

Syntax

```
Result = Pen.Count;
```

Remarks

The default is 0.

Pen.Description

The Pen.Description property is a read-write string property that gets or sets the description of the currently selected pen.

Syntax

```
Pen.Description = string;
Result = Pen.Description;
```

Remarks

If the pen expression was configured by browsing to the attribute in the Attribute Browser, the default value is obtained from the corresponding description property of the associated application server attribute. Otherwise, the default value is blank if the pen expression is manually configured.

Pen.Expression

The Pen.Expression property is a read-write string property that gets or sets the expression of the currently selected pen. The expression can be a Galaxy attribute, InTouch tag reference, or an expression.

Syntax

```
Pen.Expression = string;
Result = Pen.Expression;
```

Remarks

The default is blank.

Pen.Format

The Pen.Format property is a read-write integer property that gets or sets the format of the current pen (where 0 is decimal and 1 is scientific).

Syntax

```
Pen.Format = int;
Result = Pen.Format;
```

Remarks

The default is 0 (decimal).

Pen.HistorySource

The Pen.HistorySource property is a read-only string property that gets the history source of the currently selected pen. For more information about the HistorySources properties, see "Trend Client History Sources".

Note: This value is empty when no historical source is configured, if an expression is configured, or if no historical tag is configured.

Syntax

```
Result = Pen.HistorySource;
```

Remarks

The default is blank.

Pen.HistoryTagName

The Pen.HistoryTagName property is a read-write string property that gets or sets the historical tag name for the currently selected pen.

Syntax

```
Pen.HistoryTagName = string;
Result = Pen.HistoryTagName;
```

Remarks

The default is blank.

Pen.Index

The Pen.Index property is a read-only integer property that gets the index of the currently selected pen, or -1 if no pen is selected. The array is zero-based.

Syntax

```
Result = Pen.Index;
```

Remarks

The default is -1.

Pen.Name

The Pen.Name property is a read-write string property that gets or sets the name of the currently selected pen.

Syntax

```
Pen.Name = string;
Result = Pen.Name;
```

Remarks

The default is blank.

Pen.Precision

The Pen.Precision property is a read-write integer property that gets or sets the decimal precision of the currently selected pen.

Syntax

```
Pen.Precision = int;
Result = Pen.Precision;
```

Remarks

The default is the trend chart's pen precision property.

Pen.RetrievalMode

The Pen.RetrievalMode property is a read-write enumerated property that gets or sets the mode for retrieving the data from IndustrialSQL Server of the currently selected pen. Supported options are:

0 = Cyclic

2 = Full

12 = ApplicationSetting

Syntax

```
Pen.RetrievalMode = enum;
Result = Pen.RetrievalMode;
```

Remarks

The default is the trend chart's retrieval mode property (12).

Pen.Style

The Pen.Style property is a read-write integer property that gets or sets the line style of the currently selected pen.

Valid values are:

0 = Solid

1 = Dash

2 = DashDot

3 = DashDotDot

4 = Dot

Syntax

```
Pen.Style = int;
Result = Pen.Style;
```

Remarks

The default is 0.

Pen.TrendHi

The Pen.TrendHi property is a read-write double property that gets or sets the high value of the value range to be used for the currently selected pen.

Syntax

```
Pen.TrendHi = double;
Result = Pen.TrendHi;
```

Remarks

If the pen expression was configured by browsing to the attribute in the Attribute Browser, the default value is set to the corresponding TrendHi property of the associated application server attribute. Otherwise, the default value is 100 if the pen expression is manually configured.

Pen.TrendLo

The Pen.TrendLo property is a read-write double property that gets or sets the low value of the value range to be used for the currently selected pen.

Syntax

```
Pen.TrendLo = double;
Result = Pen.TrendLo;
```

Remarks

If the pen expression was configured by browsing to the attribute in the Attribute Browser, the default value is set to the corresponding TrendLo property of the associated application server attribute. Otherwise, the default value is 0 if the pen expression is manually configured.

Pen.TrendType

The Pen.TrendType property is a read-write integer property that gets or sets the plot type of the currently selected pen.

0 = Point

1 = Line

2 = StepLine

3 = Auto

When Pen.TrendType is set to Auto and there is no Historian default available, the Trend Client uses a plot type of **Line** for real tags and **StepLine** for integer tags.

Syntax

```
Pen.TrendType = int;
Result = Pen.TrendType;
```

Remarks

The default is 3.

Pen.Units

The Pen.Units property is a read-write string property that gets or sets the units of the currently selected pen.

Syntax

```
Pen.Units = string;
Result = Pen.Units;
```

Remarks

If the pen expression was configured by browsing to the attribute in the Attribute Browser, the default value is set to the corresponding EngUnits property of the associated application server attribute. Otherwise, the default value is blank if the pen expression is manually configured.

Pen.Visible

The Pen.Visible property is a read-write Boolean property that gets or sets the visible property for the currently selected pen.

Syntax

```
Pen.Visible = bool;
Result = Pen.Visible;
```

Remarks

The default is TRUE (the pen is visible on the chart). When this property is set to FALSE, the pen is hidden from the chart display.

Pen.Width

The Pen.Width property is a read-write integer property that gets or sets the line width (from 1 to 10) of the currently selected pen.

Syntax

```
Pen.Width = int;
Result = Pen.Width;
```

Remarks

The default is 1, when a pen is added to the pen list. If there is no pen in the pen list, the default is 0.

PenSelectorHeight

The PenSelectorHeight property is a read-write integer property that gets or sets the height of the pen selector in pixels.

Syntax

```
PenSelectorHeight = int;
Result = PenSelectorHeight;
```

Remarks

The default is 97.

PenUQRelativeOpacity

The PenUQRelativeOpacity property is a read-write integer property that gets or sets the line's relative opacity when the data quality is uncertain.

The two properties, relative opacity and relative thickness, create a visual distinction for values with uncertain quality. When the pen type is point, no visual change is shown on the trend.

Syntax

```
PenUQRelativeOpacity = int;
Result = PenUQRelativeOpacity;
```

Remarks

The default is 25%.

For example, if the pen has an opacity of 80% and the data quality is "uncertain," then the Trend Client calculates the opacity of the pen as 80% x 25% = 20% opaque.

PenUQRelativeThickness

The PenUQRelativeThickness property is a read-write integer property that gets or sets the line's relative thickness when data quality is uncertain.

The two properties, relative opacity, and relative thickness, create a visual distinction for values with uncertain quality. When the pen type is point, no visual change is shown on the trend.

Syntax

```
PenUQRelativeThickness = int;
Result = PenUQRelativeThickness;
```

Remarks

The default is 200%.

For example, if a line is 1 pixel wide and 200% thick, then the Trend Client calculates the thickness of the line with a data quality of "uncertain" as 1 pixel x 200% = 2 pixels wide.

PlotArea.BackgroundColor

The PlotArea.BackgroundColor property is a read-write property that gets or sets the color of the plot area of the graph.

Syntax

```
PlotArea.BackgroundColor = color;
Result = PlotArea.BackgroundColor;
```

Remarks

The default is white.

Color is a .NET Framework data type. You can use various color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName(). For a list of the .NET color names and the corresponding hexadecimal codes, see .NET Colors. For more information on the color methods, see the Microsoft documentation for .NET Framework development.

PlotArea.BorderColor

The PlotArea.BorderColor property is a read-write property that gets or sets the color of the plot area's border.

Syntax

```
PlotArea.BorderColor = color;
Result = PlotArea.BorderColor;
```

Remarks

The default is black.

Color is a .NET Framework data type. You can use various color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName(). For a list of the .NET color names and the corresponding hexadecimal codes, see .NET Colors. For more information on the color methods, see the Microsoft documentation for .NET Framework development.

PlotArea.GradientEndColor

The PlotArea.GradientEndColor property is a read-write property that gets or sets the gradient end color of the plot area of the graph.

Syntax

```
PlotArea.GradientEndColor = color;
Result = PlotArea.GradientEndColor;
```

Remarks

The default is white.

Color is a .NET Framework data type. You can use various color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName(). For a list of the .NET color names and the corresponding hexadecimal codes, see .NET Colors. For more information on the color methods, see the Microsoft documentation for .NET Framework development.

PlotArea.GradientType

The PlotArea.GradientType property is a read-write integer property that gets or sets the gradient type of the plot area of the graph.

0 = None (no gradient)

1 = LeftRight

2 = TopBottom

3 = Center

4 = DiagonalLeft

5 = DiagonalRight

6 = HorizontalCenter

7 = VerticalCenter

Syntax

```
PlotArea.GradientEndType = int;
Result = PlotArea.GradientEndType;
```

Remarks

The default is 0.

PlotArea.GridColor

The PlotArea.GridColor property is a read-write property that gets or sets the color of grid.

Syntax

```
PlotArea.GridColor = color;
Result = PlotArea.GridColor;
```

Remarks

The default is grey.

Color is a .NET Framework data type. You can use various color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName(). For a list of the .NET color names and the corresponding hexadecimal codes, see .NET Colors. For more information on the color methods, see the Microsoft documentation for .NET Framework development.

PlotArea.GridHorizontal

The PlotArea.GridHorizontal property is a read-write Boolean property that shows or hides the horizontal grid.

Syntax

```
PlotArea.GridHorizontal = bool;
Result = PlotArea.GridHorizontal;
```

Remarks

The default is TRUE, which implies that the horizontal grid is shown.

PlotArea.GridStyle

The PlotArea.GridStyle property is a read-write integer property that gets or sets the grid line style.

0 = Solid

1 = Dash

2 = DashDot

3 = DashDotDot

4 = Dot

Syntax

```
PlotArea.GridStyle = int;
Result = PlotArea.GridStyle;
```

Remarks

The default is 0.

PlotArea.GridVertical

The PlotArea.GridVertical property is a read-write Boolean property that shows or hides the vertical grid.

Syntax

```
PlotArea.GridVertical = bool;
Result = PlotArea.GridVertical;
```

Remarks

The default is TRUE, which implies that the vertical grid is shown.

PlotArea.GridWidth

The PlotArea.GridWidth property is a read-write integer property that gets or sets the width of the grid. Allowed values are 1 to 10.

Syntax

```
PlotArea.GridWidth = int;
Result = PlotArea.GridWidth;
```

Remarks

The default is 1.

PlotArea.HighlightCurrentPen

The PlotArea.HighlightCurrentPen property is a read-write Boolean property that highlights the currently selected pen.

Syntax

```
PlotArea.HighlightCurrentPen = bool;
Result = PlotArea.HighlightCurrentPen;
```

Remarks

The default is TRUE , which implies that the currently selected pen is highlighted.

PlotArea.PenHighlightColor

The PlotArea.PenHighlightColor property is a read-write property that gets or sets the pen’s highlight color.

Syntax

```
PlotArea.PenHighlightColor = color;
Result = PlotArea.PenHighlightColor;
```

Remarks

The default is yellow.

Color is a .NET Framework data type. You can use various color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName(). For a list of the .NET color names and the corresponding hexadecimal codes, see .NET Colors. For more information on the color methods, see the Microsoft documentation for .NET Framework development.

PlotArea.PenHighlightWidth

The PlotArea.PenHighlightWidth property is a read-write integer property that gets or sets the width of the pen's highlight color. Allowed values are 1 to 5.

Syntax

```
PlotArea.PenHighlightWidth = int;
Result = PlotArea.PenHighlightWidth;
```

Remarks

The default is 2.

PlotArea.SingleTagMode

The PlotArea.SingleTagMode property is a read-write Boolean property that sets whether to show only the currently selected pen or all pens.

Syntax

```
PlotArea.SingleTagMode = bool;
Result = PlotArea.SingleTagMode;
```

Remarks

The default is FALSE.

PlotImage

The PlotImage property is a read-write string property that gets or sets the plot background image for the chart.

Syntax

```
PlotImage = string;
Result = PlotImage;
```

Remarks

The value of this property is the folder path and filename for the image. Supported image types are .jpeg, .gif, .bmp, and .png.

The default is blank.

ShowContextMenu

The ShowContextMenu property is a Boolean property that shows the context menu in run time.

Syntax

```
ShowContextMenu = bool;
Result = ShowContextMenu;
```

Remarks

The default is TRUE, which implies that the context menu is shown in run time.

SuppressErrors

The SuppressErrors property is a read-write Boolean property that suppresses or allows error messages.

Syntax

```
SuppressErrors = bool;
Result = SuppressErrors;
```

Remarks

The default is TRUE which implies that the errors are suppressed.

Note: All errors are logged to Logger regardless of this setting.

TimeAxis.Cursor1.Color

The TimeAxis.Cursor1.Color property is a read-write property that gets or sets the color of the left time axis cursor.

Syntax

```
TimeAxis.Cursor1.Color = color;
Result = TimeAxis.Cursor1.Color;
```

Remarks

The default is red.

Color is a .NET Framework data type. You can use various color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName(). For a list of the .NET color names and the corresponding hexadecimal codes, see .NET Colors. For more information on the color methods, see the Microsoft documentation for .NET Framework development.

TimeAxis.Cursor1.Pos

The TimeAxis.Cursor1.Pos property is a read-write property that gets or sets the time position of the left time axis cursor.

Syntax

```
TimeAxis.Cursor1.Pos = datetime;
Result = TimeAxis.Cursor1.Pos;
```

Remarks

No default.

TimeAxis.Cursor1.Style

The TimeAxis.Cursor1.Style property is a read-write integer property that gets or sets the style of the left time axis cursor. Valid values are:

0 = Solid

1 = Dash

2 = DashDot

3 = DashDotDot

4 = Dot

Syntax

```
TimeAxis.Cursor1.Style = int;
Result = TimeAxis.Cursor1.Style;
```

Remarks

The default is 0.

TimeAxis.Cursor1.Width

The TimeAxis.Cursor1.Width property is a read-write integer property that gets or sets the width of the left time axis cursor. Allowed values are 1 to 10.

Syntax

```
TimeAxis.Cursor1.Width = int;
Result = TimeAxis.Cursor1.Width;
```

Remarks

The default is 1.

TimeAxis.Cursor2.Color

The TimeAxis.Cursor2.Color property is a read-write property that gets or sets the color of the right time axis cursor.

Syntax

```
TimeAxis.Cursor2.Color = color;
Result = TimeAxis.Cursor2.Color;
```

Remarks

The default is blue.

Color is a .NET Framework data type. You can use various color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName(). For a list of the .NET color names and the corresponding hexadecimal codes, see .NET Colors. For more information on the color methods, see the Microsoft documentation for .NET Framework development.

TimeAxis.Cursor2.Pos

The TimeAxis.Cursor2.Pos property is a read-write property that gets or sets the time position of the right time axis cursor.

Syntax

```
TimeAxis.Cursor2.Pos = datetime;
Result = TimeAxis.Cursor2.Pos;
```

Remarks

No default.

TimeAxis.Cursor2.Style

The TimeAxis.Cursor2.Style property is a read-write integer property that gets or sets the style of the right time axis cursor. Valid values are:

0 = Solid

1 = Dash

2 = DashDot

3 = DashDotDot

4 = Dot

Syntax

```
TimeAxis.Cursor2.Style = int;
Result = TimeAxis.Cursor2.Style;
```

Remarks

The default is 0.

TimeAxis.Cursor2.Width

The TimeAxis.Cursor2.Width property is a read-write integer property that gets or sets the width of the right time axis cursor. Allowed values are 1 to 10.

Syntax

```
TimeAxis.Cursor2.Width = int;
Result = TimeAxis.Cursor2.Width;
```

Remarks

The default is 1.

TimeAxis.LabelColor

The TimeAxis.LabelColor property is a read-write property that gets or sets the color of the time axis labels.

Syntax

```
TimeAxis.LabelColor = color;
```

```
Result = TimeAxis.LabelColor;
```

Remarks

The default is black.

Color is a .NET Framework data type. You can use various color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName(). For a list of the .NET color names and the corresponding hexadecimal codes, see .NET Colors. For more information on the color methods, see the Microsoft documentation for .NET Framework development.

TimeAxis.NumGridPerValue

The TimeAxis.NumGridPerValue property is a read-write integer property that gets or sets the number of grid lines that appear between each value shown on time axis. The valid range is 1 to 20.

Syntax

```
TimeAxis.NumGridPerValue = int;
Result = TimeAxis.NumGridPerValue;
```

Remarks

The default is 3.

TimeAxis.NumValues

The TimeAxis.NumValues property is a read-write integer property that gets or sets the number of time labels that are shown along the time axis. The valid range is 2 to 15.

Syntax

```
TimeAxis.NumValues = int;
Result = TimeAxis.NumValues;
```

Remarks

The default is 6.

TimeAxis.ShowCursors

The TimeAxis.ShowCursors property is a read-write Boolean property that shows or hides the time axis cursors.

Syntax

```
TimeAxis.ShowCursors = bool;
```

```
Result = TimeAxis.ShowCursors;
```

Remarks

The default is FALSE.

TimeSelector

The TimeSelector property is a read-only property that gets the Time Range Picker object used in the Trend Client. You can use it in scripting to shorten the code using its properties and methods.

For the individual properties and methods, see the following properties, or the methods starting at "TimeSelector.GetStartAndEndTimes".

Example 1

```
dim TRP as object;
TRP = TrendClient1.TimeSelector;
StartDate = TRP.StartDate;
EndDate = TRP.EndDate;
duration = TRP.TimeDuration;
```

Example 2

```
dim TRP as object;
TRP = TrendClient1.TimeSelector;
TRP.SetStartAndEndTimes(StartDate, EndDate, Duration);
```

Remarks

The return value is a TimeRangePicker.

TimeSelector.DurationMS

The TimeSelector.DurationMS property is a read-write integer property that gets the time duration measured in milliseconds.

The start time of the Trend Client (TimeSelector.StartDate) is calculated as the end time (TimeSelector.EndDate) minus the new time duration (TimeSelector.DurationMS).

When you set the value of the TimeSelector.DurationMS property, the TimeSelector.TimeDuration property is set to 0.

Syntax

```
result = TimeSelector.DurationMS;
TimeSelector.DurationMS = Value;
```

Remarks

The default value is 300000.

TimeSelector.EndDate

The TimeSelector.EndDate property is a read-only string property that gets the end date and time of the Trend Client.

The default value is the time the Trend Client is placed on the canvas. If the **Update to Current Time** option is enabled, the TimeSelector.EndDate property is updated with the current time.

Note: To set the end date and time of the Trend Client, use the TimeSelector.SetStartAndEndTimes method.

Syntax

```
result = Trend01.TimeSelector.EndDate;
```

TimeSelector.StartDate

The TimeSelector.StartDate property is a read-only string property that gets the start date and time of the Trend Client.

Syntax

```
result = TimeSelector.StartDate;
```

Remarks

The default value is the time the Trend Client is placed on the canvas minus the duration.

Note: To set the start date and time of the Trend Client, use the TimeSelector.SetStartAndEndTimes method.

TimeSelector.TimeDuration

The TimeSelector.TimeDuration property is a read-write enumerated property that gets or sets the enumerated duration of the time axis of the chart.

Syntax

```
TimeSelector.TimeDuration = enum;
Result = TimeSelector.TimeDuration;
```

The TimeSelector.TimeDuration property is a read-write integer property that gets or sets the time duration. The start time of the Trend Client (TimeSelector.StartDate) is calculated as the end time (TimeSelector.EndDate) minus the new time duration.

The TimeSelector.TimeDuration can have one of the following values:

Value	Description
0	Custom
1	The last minute.
2	The last five minutes.
3	The last ten minutes.
4	The last 15 minutes.
5	The last 30 minutes.
6	The last hour.
7	The last two hours.
8	The last four hours.
9	The last eight hours.
10	The last 12 hours.
11	The last 24 hours.
12	The last two days.
13	The last week.
14	The last two weeks.
15	The last month.
16	The last three months.
17	One minute.
18	Five minutes.
19	Ten minutes.
20	15 minutes.
21	30 minutes.
22	One hour.
23	Two hours.
24	Four hours.
25	Eight hours.

Value	Description
26	12 hours.
27	24 hours.
28	Two days.
29	One week.
30	Two weeks.
31	One month.
32	Three months.
33	Yesterday: 0:00:00 of the previous day to 0:00:00 of the current day.
34	Current day: 0:00:00 of the current day to the current time.
35	Previous hour: The start of the previous hour to the start of the current hour.
36	Current hour: The start of the current hour to the current time.

Syntax

```
result = TimeSelector.TimeDuration;
TimeSelector.TimeDuration = Value;
```

Example

```
Trend01.TimeSelector.TimeDuration = 5;
// The trend is now set to show the last 30 minutes.
```

Remarks

The default is 18 (5 minutes).

ToolTipText

The ToolTipText property is a read-write string property that gets or sets the pop-up text that appears when the mouse is hovered over the chart at run time.

Syntax

```
ToolTipText = string;
Result = ToolTipText;
```

Remarks

The default is blank.

TrendVersion

The TrendVersion property is a read-only string property that gets the version of the trend.

Syntax

```
Result = TrendVersion;
```

Remarks

The return value is 1.0.0 for the first release.

ValueAxis.Label

The ValueAxis.Label property is a read-write integer property that gets or sets which labels are shown on the value axis.

0 = MultipleScales

1 = SingleScale

2 = ValuesAtCursor

Syntax

```
ValueAxis.Label = int;
Result = ValueAxis.Label;
```

Remarks

The default is 0.

ValueAxis.NumGridPerValue

The ValueAxis.NumGridPerValue property is a read-write integer property that gets or sets the number of grid lines that appear between each value shown along the Y-axis. The valid range is 1 to 20.

Syntax

```
ValueAxis.NumGridPerValue = int;
Result = ValueAxis.NumGridPerValue;
```

Remarks

The default is 2.

ValueAxis.NumValues

The ValueAxis.NumValues property is a read-write integer property that gets or sets the number of value labels that are shown along the Y-axis. The valid range is 2 to 15.

Syntax

```
ValueAxis.NumValues = int;
Result = ValueAxis.NumValues;
```

Remarks

The default is 6.

Visible

The Visible property is a read-write Boolean property that shows or hides the Trend Client at run time.

Syntax

```
Visible = bool;
Result = Visible;
```

Remarks

The default is TRUE.

Trend Client history sources

The Trend Client has the following scriptable methods for the history sources:

- HistorySources.Add
- HistorySource.Authentication
- HistorySource.Connect
- HistorySources.Count

- HistorySource.Disconnect
- HistorySource.Domain
- HistorySources.GetSource
- HistorySources.Items
- HistorySource.Password
- HistorySources.Remove
- HistorySource.RetainPassword
- HistorySource.ServerName
- HistorySource.Type
- HistorySource.UNCPath
- HistorySources.Update
- HistorySource.UserID

HistorySources.Add

HistorySources.Add method adds a new history source.

Example

```
Dim histSources = Trend1.HistorySources;
Dim ret = histSources.Add("idc_insql12", "InSQL");
```

Syntax

```
[Result =] HistorySources.Add(string HistorySourceName, string Type);
```

Parameters

HistorySourceName

The name of the history source.

Type

The type of history source "InSQL" or "InTouch".

Return Value

If there is already a server with the given name in the list, the object for that server is returned. Otherwise, a new server with the given name is added to the list and the object for the new server is returned.

HistorySource.Authentication

The HistorySource.Authentication property is a read-write string property that gets or sets the authentication mode for the connection to the server. Available options are SQL Server, Windows Account, and Windows

Integrated.

Syntax

```
HistorySource.Authentication = string;
Result = HistorySource.Authentication;
```

Remarks

The default is Windows Integrated.

HistorySource.Connect

HistorySource.Connect method logs on to the current history source.

Example

```
Dim histSources = Trend1.HistorySources;
Dim histSource = histSources.GetSource("idc_insql12");
Dim statusMsg as string;
Dim ret = histSource.Connect(statusMsg);
```

Syntax

```
[Result =] HistorySource.Connect(string statusMessage);
```

Parameters

statusMessage

Information of the result of the log on attempt.

Return Value

Returns TRUE if the log on was successful; otherwise, returns FALSE. The server must be configured before calling the Connect method. Changes made to the server configuration after a connect do not take effect until after a disconnect and subsequent connect.

HistorySources.Count

The HistorySources.Count property is a read-only integer property that shows the number of history sources configured.

Syntax

```
Result = HistorySources.Count;
```

Remarks

The default is 0.

HistorySource.Disconnect

HistorySource.Disconnect method logs off the connection to the current history source.

Example

```
Dim histSources = Trend1.HistorySources;
Dim histSource = histSources.GetSource("idc_insql12");
histSource.Disconnect();
```

Syntax

```
[Result =] HistorySource.Disconnect();
```

Parameters

None.

Return Value

None.

Remarks

Repeated calls to this method are harmless and do not result in further state change events.

HistorySource.Domain

The HistorySource.Domain property is a read-write string property that gets or sets the domain name for the connection to the Historian.

Syntax

```
HistorySource.Domain = string;
Result = HistorySource.Domain;
```

Remarks

The default is an empty message value ("").

This property is not applicable if you are using an InTouch LGH file as the data source.

HistorySources.GetSource

HistorySources.GetSource method gets the named history source object for a server from the server list.

Example

```
Dim histSources = Trend1.HistorySources;
Dim histSource = histSources.GetSource("IDC_INSQL15");
```

Syntax

```
[Result =] HistorySources.GetSource(string HistorySourceName);
```

Parameters

HistorySourceName

The name of the history source.

Return Value

If the server exists, the object is returned; otherwise, a NULL is returned.

HistorySources.Items

The HistorySources.Items property is a read-only array property that shows the array of history source names.

Syntax

```
Result = HistorySources.Items;
```

Remarks

No default.

Example

```
Populate a string array uda of a userdefined instance with the server names configured in
the Trend Control, and then add them to a list box control on the canvas)
dim i as integer;
dim b as object;
ListBox1.Clear();
for i = 1 to TrendClient1.HistorySources.Count;
b = TrendClient1.HistorySources.Items[i];
UserDefined_001.items[i] = b.ServerName;
ListBox1.AddItem(UserDefined_001.Items[i]);
next;
```

HistorySource.Password

The HistorySource.Password property is a read-write string property that gets and sets the password for the connection to the server.

Syntax

```
HistorySource.Password = string;
Result = HistorySource.Password;
```

Remarks

The default is wwUser.

HistorySources.Remove

HistorySources.Remove method removes the specified history source from the list.

Example

```
Dim histSources = Trend1.HistorySources;
Dim ret = histSources.Remove("IDC_INSQL15");
```

Syntax

```
[Result =] HistorySources.Remove(string HistorySourceName);
```

Parameters

HistorySourceName

The name of the history source.

Return Value

This method returns TRUE if the instance was removed from the list. This method returns FALSE if the exact instance is not in the list, and the list remains unchanged.

HistorySource.RetainPassword

The HistorySource.RetainPassword property is a read-write Boolean property that indicates whether the password is stored in persistent storage.

Syntax

```
HistorySource.RetainPassword = bool;
```

```
Result = HistorySource.RetainPassword;
```

Remarks

The default is TRUE (the password is stored in persistent storage).

This property is not applicable if you are using an InTouch LGH file as the data source.

HistorySource.ServerName

The HistorySource.ServerName property is a read-write string property that gets or sets the name of the history source.

Syntax

```
HistorySource.ServerName = string;
Result = HistorySource.ServerName;
```

Remarks

The default is an empty message value ("").

HistorySource.Type

The HistorySource.Type property is a read-write string property that gets or sets the type of the history source. Possible values are "InSQL" (Historian Server) and InTouch (InTouch History Files).

Syntax

```
HistorySource.Type = string;
Result = HistorySource.Type;
```

Remarks

The default is "InSQL."

HistorySource.UNCPath

The HistorySource.UNCPath property is a read-write string property that gets or sets the UNC path of the InTouch Log History/LGH file.

Syntax

```
HistorySource.UNCPath = string;
Result = HistorySource.UNCPath;
```

Remarks

The default is an empty message value ("").

HistorySources.Update

HistorySources.Update method updates the specified history source in the list.

Example

```
Dim histSources = Trend1.HistorySources;
Dim ret = histSources.Update("IDC_INSQL15");
```

Syntax

```
[Result =] HistorySources.Update(string HistorySourceName);
```

Parameters

HistorySourceName

The name of the history source.

Return Value

Returns TRUE if the given instance is currently in the server list; otherwise, it returns FALSE.

HistorySource.UserID

The HistorySource.UserID property is a read-write string property that gets and sets the user ID for the Historian.

Syntax

```
HistorySource.UserID = string;
Result = HistorySource.UserID;
```

Remarks

The default is wwUser.

This property is not applicable if you are using an InTouch LGH file as the data source.

Trend Client methods

The following are the methods used by the Trend Client:

- AddHistorianSource
- AddPen
- ClearPens
- DeleteCurrentPen
- GetHistorianSource
- GetPenValAtX1
- GetStartAndEndTimes
- MoveNextPen
- MovePrevPen
- RefreshData
- RefreshTimes
- RemoveHistorianSource
- RemovePen
- ScaleAllPens
- ScaleAutoAllPens
- ScaleAutoPen
- ScaleDownAllPens
- ScaleDownPen
- ScaleMoveAllPensDown
- ScaleMoveAllPensUp
- ScaleMovePenDown
- ScaleMovePenUp
- ScalePen
- ScaleUpAllPens
- ScaleUpPen
- SetCurrentPen
- SetDuration
- SetStartAndEndTimes
- TimeSelector.GetStartAndEndTimes
- TimeSelector.RefreshTimes
- TimeSelector.SetStartAndEndTimes
- UpdateHistorianSource

The following section describes the scriptable methods available in the Trend Client.

AddHistorianSource

AddHistorianSource method adds a new history source.

Example

```
Dim b as object;
b = Trend1.AddHistorianSource("idc_insql12", "InSQL");
```

Syntax

```
[Result =] Trend1.AddHistorianSource(string HistorySourceName, string Type);
```

Parameters

HistorySourceName

The name of the history source.

Type

The type of history source "InSQL" or "InTouch".

Return Value

If there is already a server with the given name in the list, the object for that server is returned. Otherwise, a new server with the given name is added to the list and the object for the new server is returned.

Recommended Usage

Whenever the AddHistorianSource method is used in a script, such as the Predefined Script OnShow on a symbol, we strongly recommend that you add a corresponding call to RemoveHistorianSource. This can be added in a Predefined Script OnHide on a symbol. AddHistorianSource creates resources inside the InTouch view application that are not cleaned up when a window is closed during a view.exe session. To clean up these resources, you must use a call from a script to RemoveHistorianSource, or close the view.exe session and end the runtime application.

AddPen

The AddPen method adds a named pen to the trend.

Examples

```
dim b as boolean;
b = Trend1.AddPen("MyPen01","UserDefined_001.Value", "InSQL01","UserDefined_001.Value", 1);
b = Trend1.AddPen("MyPen02","InTouch:$Second", "InSQL01","$Second", 1);
b = Trend1.AddPen("MyPen03", "MyContainer.InletPump1+MyContainer.InletPump2","","",1);
```

Syntax

```
[Result=] AddPen(string PenName, string TagName, string HistorySource, string
HistoryTagName, int HistoryTagType);
```

Parameters

PenName

The name of the pen to add.

TagName

The name of the tag associated with the pen being added.

HistorySource

The history source for the pen being added.

HistoryTagName

The name of the history tag for the pen being added.

HistoryTagType

The type of history tag for the pen being added: 1 for Historian or 3 for LGH.

Return Value

Returns TRUE if the pen was successfully added; otherwise, returns FALSE.

Recommended Usage

Whenever the AddPen method is used in a script, such as the Predefined Script OnShow on a symbol, we strongly recommend that you add a corresponding call to ClearPens. This can be added in a Predefined Script OnHide on a symbol. AddPen creates resources inside the InTouch view application that are not cleaned up when a window is closed during a view.exe session. To clean up these resources, you must use a call from a script to the ClearPens method, or close the view.exe session and end the runtime application.

ClearPens

The ClearPens method removes all pens from the trend.

Example

```
dim b as boolean;
b = Trend1.ClearPens();
```

Syntax

```
[Result=] Trend1.ClearPens();
```

Parameters

None.

Return Value

Returns TRUE if the pens were successfully deleted from the trend; otherwise, returns FALSE.

DeleteCurrentPen

The DeleteCurrentPen method removes the currently selected pen from the trend.

Example

```
dim b as boolean;
b = Trend1.DeleteCurrentPen();
```

Syntax

```
[Result=] Trend1.DeleteCurrentPen();
```

Parameters

None.

Return Value

Returns TRUE if the pen was successfully removed; otherwise, returns FALSE.

GetHistorianSource

GetHistorianSource method gets the named history source object for a server from the server list.

Example

```
Dim b as object;
b = Trend1.GetHistorianSource("idc_insql15");
```

Syntax

```
[Result =] Trend1.GetHistorianSource(string HistorySourceName);
```

Parameters

HistorySourceName

The name of the history source.

Return Value

If the server exists, the object is returned; otherwise, a NULL is returned.

GetPenValAtX1

The GetPenValAtX1 method gets the value of the expression associated with the specified pen at the point at which its curve intersects the axis cursor the first time.

Example

```
double val = Trend1.GetPenValAtX1("MyPen01");
```

Syntax

```
[Result=] Trend1.GetPenValAtX1(string PenName);
```

Parameters

PenName

The pen you want to check the value of.

Return Value

Returns a double if the point is analog or discrete; returns the fixed position of the point if the point is message or other type. If the specified pen is shown in the chart multiple times, then the method uses the first instance that was added.

GetPenValAtX2

The GetPenValAtX2 method gets the value of the expression associated with the specified pen at the point at which its curve intersects the axis cursor the second time.

Example

```
double val = Trend2.GetPenValAtX2("MyPen01");
```

Syntax

```
[Result=] Trend2.GetPenValAtX2(string PenName);
```

Parameters

PenName

The pen you want to check the value of.

Return Value

Returns a double if the point is analog or discrete; returns the fixed position of the point if the point is message or other type. If the specified pen is shown in the chart multiple times, then the method uses the first instance that was added.

GetStartAndEndTimes

The GetStartAndEndTimes method gets the start and end times for the query.

Example

```
Dim SDate as System.DateTime;
Dim EDate as System.DateTime;
Dim b as System.Int32
b = Trend1.GetStartAndEndTimes(SDate, EDate);
StartTime = SDate.ToString();
EndTime = EDate.ToString();
Ret = b;
```

Syntax

```
[Result =] Trend1.GetStartAndEndTimes(DateTime StartTime, DateTime EndTime);
```

Parameters

StartTime

The start time for the query.

EndTime

The end time for the query.

Return Value

Returns the time range enumeration.

MoveNextPen

The MoveNextPen method sets the current pen focus to the next pen in the list of trended attributes or tags. If the current pen is the last pen, calling the MoveNextPen method sets the first pen as the current pen.

Example

```
dim b as boolean;
b = Trend1.MoveNextPen();
```

Syntax

```
[Result=] Trend1.MoveNextPen();
```

Parameters

None.

Return Value

Returns TRUE if the current pen focus was successfully set to the next pen (or wrapped around to the first pen in the list); otherwise, returns FALSE.

MovePrevPen

The MovePrevPen method sets the current pen focus to the previous pen in the list of trended attributes or tags. If the current pen is the first pen, calling the MovePrevPen method has no effect.

Example

```
dim b as boolean;
b = Trend1.MovePrevPen();
```

Syntax

```
[Result=] Trend1.MovePrevPen();
```

Parameters

None.

Return Value

Returns TRUE if the current pen focus was successfully set to the previous pen; otherwise, returns FALSE.

RefreshData

The RefreshData method refreshes the trend chart by retrieving new data for all pens.

Example

```
bool b = Trend1.RefreshData();
```

Syntax

```
[Result=] Trend1.RefreshData();
```

Parameters

None.

Return Value

Returns TRUE if the trend was successfully updated; otherwise, FALSE is returned.

RefreshTimes

The RefreshTimes method sets the time period for the query by updating the end time to current time and recalculates the start time based on the new end time and duration.

Example

```
dtag = 1;
Trend1.RefreshTimes(dtag);
```

Syntax

```
Trend1.RefreshTimes(bool TriggerEvent);
```

Parameters

TriggerEvent

If you set the Boolean parameter to TRUE, the OnChange event is triggered if the time is updated.

RemoveHistorianSource

RemoveHistorianSource method removes the specified history source from the list.

Example

```
Dim b as Boolean;
b = Trend1.RemoveHistorianSource("IDC_INSQL15");
```

Syntax

```
[Result] = Trend1.RemoveHistorianSource(string HistorySourceName);
```

Parameters

HistorySourceName

The name of the history source.

Return Value

This method returns true if the instance was removed from the list. This method returns false if the exact instance is not in the list, and the list remains unchanged.

RemovePen

The RemovePen method removes the specified pen from the trend chart.

Note: If the same pen is available multiple times, then only the first occurrence is removed.

Example

```
dim b as boolean;
b = Trend1.RemovePen("MyPen01");
```

Syntax

```
[Result=] Trend1.RemovePen(string PenName);
```

Parameters

PenName

The pen name to be removed.

Return Value

Returns TRUE if the pen was successfully removed from the trend chart; otherwise, returns FALSE.

ScaleAllPens

The ScaleAllPens method sets the Y-axis (value axis) scale for all pens.

Example

```
dim b as boolean;
b = Trend1.ScaleAllPens(-100, 100);
```

Syntax

```
[Result=] Trend1.ScaleAllPens(double Min, double Max);
```

Parameters

Min

Type the minimum value for the Y-axis.

Max

Type the maximum value for the Y-axis.

Return Value

Returns TRUE if the Y-axis scale is set successfully; otherwise, returns FALSE.

ScaleAutoAllPens

The ScaleAutoAllPens method sets a suitable Y-axis scale for all tags in accordance with the current minimum and maximum values.

Example

```
dim b as boolean;
b = Trend1.ScaleAutoAllPens();
```

Syntax

```
[Result=] Trend1.ScaleAutoAllPens();
```

Parameters

None.

Return Value

Returns TRUE if the Y-axis is successfully scaled; otherwise, returns FALSE.

ScaleAutoPen

The ScaleAutoPen method sets a suitable Y-axis scale for the currently selected pen in accordance with the current minimum and maximum values.

Example

```
dim b as boolean;
b = Trend1.ScaleAutoPen();
```

Syntax

```
[Result=] Trend1.ScaleAutoPen();
```

Parameters

None.

Return Value

Returns TRUE if the Y-axis is successfully scaled; otherwise, returns FALSE.

ScaleDownAllPens

The ScaleDownAllPens method increases the value range of all pens by one third.

Example

```
dim b as boolean;
b = Trend1.ScaleDownAllPens();
```

Syntax

```
[Result=] Trend1.ScaleDownAllPens();
```

Parameters

None.

Return Value

Returns TRUE if the value range is successfully increased by one third; otherwise, returns FALSE.

ScaleDownPen

The ScaleDownPen method increases the value range of the currently selected pen by one third.

Example

```
dim b as boolean;
b = Trend1.ScaleDownPen();
```

Syntax

```
[Result=] Trend1.ScaleDownPen();
```

Parameters

None.

Return Value

Returns TRUE if the scaling is successful; otherwise, returns FALSE.

ScaleMoveAllPensDown

The ScaleMoveAllPensDown method moves the value scale down for all pens.

Example

```
dim b as boolean;
b = Trend1.ScaleMoveAllPensDown();
```

Syntax

```
[Result=] Trend1.ScaleMoveAllPensDown();
```

Parameters

None.

Return Value

Returns TRUE if the scaling is successful; otherwise, returns FALSE.

ScaleMoveAllPensUp

The ScaleMovesAllPensUp method moves the value scale up for all pens.

Example

```
dim b as boolean;
b = Trend1.ScaleMoveAllPensUp();
```

Syntax

```
[Result=] Trend1.ScaleMoveAllPensUp();
```

Parameters

None.

Return Value

Returns TRUE if the scaling is successful; otherwise, returns FALSE.

ScaleMovePenDown

The ScaleMovePenDown method moves the value scale down for the currently selected pen.

Example

```
dim b as boolean;
b = Trend1.ScaleMovePenDown();
```

Syntax

```
[Result=] Trend1.ScaleMovePenDown();
```

Parameters

None.

Return Value

Returns TRUE if the scaling is successful; otherwise, returns FALSE.

ScaleMovePenUp

The ScaleMovePenUp method moves the value scale up for the currently selected pen.

Example

```
dim b as boolean;
b = Trend1.ScaleMovePenUp();
```

Syntax

```
[Result=] Trend1.ScaleMovePenUp();
```

Parameters

None.

Return Value

Returns TRUE if the scaling is successful; otherwise, returns FALSE.

ScalePen

The ScalePen method sets the Y-axis scale for the currently selected pen.

Example

```
dim b as boolean;
b = Trend1.ScalePen(-100, 100);
```

Syntax

```
[Result=] Trend1.ScalePen(double Min, double Max);
```

Parameters

Min

Type the minimum value for the Y-axis.

Max

Type the maximum value for the Y-axis.

Return Value

Returns TRUE if the Y-axis scale is set successfully; otherwise, returns FALSE.

ScaleUpAllPens

The ScaleUpAllPens method decreases the value range of all pens by one fourth.

Example

```
dim b as boolean;
b = Trend1.ScaleUpAllPens();
```

Syntax

```
[Result=] Trend1.ScaleUpAllPens();
```

Parameters

None.

Return Value

Returns TRUE if the value range is successfully reduced by one fourth; otherwise, returns FALSE.

ScaleUpPen

The ScaleUpPen method decreases the value range of the currently selected pen by one fourth.

Example

```
dim b as boolean;
b = Trend1.ScaleUpPen();
```

Syntax

```
[Result=] Trend1.ScaleUpPen();
```

Parameters

None.

Return Value

Returns TRUE if the value range is successfully reduced by one fourth; otherwise, returns FALSE.

SetCurrentPen

The SetCurrentPen method sets the specified pen as the currently selected pen. If the pen is available multiple times, then the first occurrence is selected.

Example

```
dim b as boolean;
b = Trend1.SetCurrentPen("MyPen01");
```

Syntax

```
[Result=] Trend1.SetCurrentPen(string PenName);
```

Parameters

PenName

The name of the pen being set.

Return Value

Returns TRUE if the pen was set successfully; otherwise, returns FALSE.

SetDuration

The SetDuration method sets the duration of the trend. Calling this method first sets the end time to the current time and then the start time to current time minus the specified duration. Default duration is 5 minutes. Maximum duration is 24 hours.

Example

```
dim b as boolean;
b = Trend1.SetDuration("00:05:00");
```

Syntax

```
[Result=] Trend1.SetDuration(string DateTimeDuration);
```

Parameters

DateTimeDuration

The time of the duration in HH:MM:SS format.

Return Value

Returns TRUE if the duration is set successfully; otherwise, returns FALSE.

SetStartAndEndTimes

The SetStartAndEndTimes method sets the start and end times for the query.

Example

```
Dim b as Boolean;
b = Trend1.SetStartAndEndTimes("08/31/2008 15:33:43","09/01/2009 15:33:43",0);
```

Syntax

```
[Result =] Trend1.SetStartAndEndTimes(DateTime StartTime, DateTime EndTime, integer
Duration);
```

Parameters

StartTime

The start time for the query. Only considered if the duration is set to Custom. For other durations, the start time is calculated automatically based on the end time and duration.

EndTime

The end time for the query. Only considered if the duration is set to Custom or an option from 17 to 32

(OneMinute to ThreeMonths). Otherwise, the end time is set based on the duration.

Duration

The time1 range duration. If the duration is set to Custom, the specified start and end times are used. For other duration options, the time indicated by the duration is used, and the start and/or end times are updated as necessary. For more information on valid values for the duration, see "TimeSelector.TimeDuration".

TimeSelector.GetStartAndEndTimes

The TimeSelector.GetStartAndEndTimes method gets the start and end times for the query.

Syntax

```
Trend01.GetStartAndEndTimes(DateTime StartTime, DateTime EndTime);
```

Parameters

StartTime

String attribute, custom property, or element property to retrieve the start time.

EndTime

String attribute, custom property, or element property to retrieve the end time.

Example

```
dim SDate as string;
dim EDate as string;
Trend01.TimeSelector.GetStartAndEndTimes(SDate, EDate);
StartDate = SDate;
EndDate = EDate;
```

TimeSelector.RefreshTimes

The TimeSelector.RefreshTimes method sets the time period for the query by updating the end time to current time and recalculates the start time based on the new end time and duration.

If you set the Boolean parameter to TRUE, the OnChange event is triggered if the time is updated.

Only use this method if the **Update to Current Time** option is cleared or the **UpdateToCurrentTime** property is FALSE. Otherwise, this method does not work.

Syntax

```
Trend01.TimeSelector.RefreshTimes(TriggerEvent);
```

Example

```
dtag = 1;
Trend01.TimeSelector.RefreshTimes(dtag);
```

TimeSelector.SetStartAndEndTimes

The TimeSelector.SetStartAndEndTimes method sets the start and end times for the query.

You must specify one of the following parameter combinations:

- Start time and end time. Set the Duration parameter to Custom (0).
- Start time and duration. Set the EndTime parameter to "".
- End time and duration. Set the StartTime parameter to "".
- Start time, duration, and end time. The Trend Client shows an error message if start time plus duration is not equal to end time.

Syntax

```
Trend01.SetStartAndEndTimes(DateTime StartTime, DateTime EndTime, integer Duration);
```

Parameters

StartTime

The start time for the query. Only considered if the duration is set to Custom (0) by the Duration parameter. For other durations, the start time is calculated automatically based on the end time and duration.

EndTime

The end time for the query. Only considered if the duration is set to Custom or an option from 17 to 32 (OneMinute to ThreeMonths) by the Duration parameter. Otherwise, the end time is set based on the duration.

Duration

Duration enum. For more information on possible duration values, see "TimeSelector.TimeDuration".

Example

```
Trend01.TimeSelector.SetStartAndEndTimes("08/31/2008 15:33:43","09/01/2009 15:33:43",0);
```

UpdateHistorianSource

UpdateHistorianSource method updates the specified history source in the list.

Example

```
Dim a as object;
Dim b as Boolean;
a = Trend1.GetHistorianSource ("IDC_INSQL15");
a.Authentication = "Windows Integrated";
b = Trend1.UpdateHistorianSource("IDC_INSQL15");
```

Syntax

```
[Result =] Trend1.UpdateHistorianSource(string HistorySourceName);
```

Parameters

HistorySourceName

The name of the history source.

Return Value

Returns TRUE if the given instance is currently in the server list; otherwise, it returns FALSE.

Trend Client events

The following events are relevant to the Trend Client.

- CurrentPenChanged
- DatesChanged
- MouseClick
- PenDisplayChanged
- PenlistChanged
- ShutDown
- SizeChanged
- StartUp
- StateChanged

CurrentPenChanged

The CurrentPenChanged event is triggered when a different pen is selected.

Syntax

```
TrendClient.CurrentPenChanged();
```

DatesChanged

The DatesChanged event is triggered when a date/time for the trend chart changes because there is a scripted change to the chart's duration.

Syntax

```
TrendClient.DatesChanged();
```

MouseClick

The MouseClick event is triggered when there is a mouse click in the trend region.

Syntax

```
TrendClient.MouseClick();
```

PenDisplayChanged

The PenDisplayChanged event is triggered when the display options for a pen in the pen list are changed.

Syntax

```
TrendClient.PenDisplayChanged();
```

PenlistChanged

The PenlistChanged event is triggered when a pen is added to or removed from the pen list.

Syntax

```
TrendClient.PenlistChanged();
```

ShutDown

The ShutDown event is triggered when the client is closed.

Syntax

```
TrendClient.ShutDown();
```

SizeChanged

The SizeChanged event is triggered when a resize event is received.

Syntax

```
TrendClient.SizeChanged();
```

StartUp

The StartUp event is triggered when the client is started.

Syntax

```
TrendClient.StartUp();
```

StateChanged

The StateChanged event is triggered when a change has been made to the configuration for a pen in the pen list.

Syntax

```
TrendClient.StateChanged();
```

Colors in Trend Client

Trend Client and ActiveFactory use different color rendering schemes. ActiveFactory uses BGR and ABGR color rendering,

where:

A = Transparency (for ABGR only)

B = Blue

G = Green

R = Red

Information on how colors are rendered in ActiveFactory appears in Chapter 19, Common Properties, Methods, Events, Enums, and Data Types, of the ActiveFactory Software User's Guide.

In contrast, in the Trend Client, color is a .NET Framework data type. You can use various color methods to set the color, such as a predefined color name, FromARGB(), FromKnownColor(), and FromName(). For example, to set the pen color on a Trend Client Trend named **RealtimeTrend1** to **DarkGreen**, you use the following script statement:

```
RealtimeTrend1.Pen.Color = System.Drawing.Color.FromName("DarkGreen");
```

For more information on the color methods, see the Microsoft documentation for .NET Framework development.

.NET colors

The following table is an overview of the color .NET color names with hexadecimal code.

Color with Hex Code	Color with Hex Code	Color with Hex Code
AliceBlue #F0F8FF	AntiqueWhite #FAEBD7	Aqua #00FFFF
Aquamarine #7FFFD4	Azure #F0FFFF	Beige #F5F5DC
Bisque #FFE4C4	Black #000000	BlanchedAlmond #FFEBCD

Color with Hex Code	Color with Hex Code	Color with Hex Code
Blue #0000FF	BlueViolet #8A2BE2	Brown #A52A2A
BurlyWood #DEB887	CadetBlue #5F9EA0	Chartreuse #7FFF00
Chocolate #D2691E	Coral #FF7F50	CornflowerBlue #6495ED
Cornsilk #FFF8DC	Crimson #DC143C	Cyan #00FFFF
DarkBlue #00008B	DarkCyan #008B8B	DarkGoldenrod #B8860B
DarkGray #A9A9A9	DarkGreen #006400	DarkKhaki #BDB76B
DarkMagenta #8B008B	DarkOliveGreen #556B2F	DarkOrange #FF8C00
DarkOrchid #9932CC	DarkRed #8B0000	DarkSalmon #E9967A
DarkSeaGreen #8FBC8B	DarkSlateBlue #483D8B	DarkSlateGray #2F4F4F
DarkTurquoise #00CED1	DarkViolet #9400D3	DeepPink #FF1493
DeepSkyBlue #00BFFF	DimGray #696969	DodgerBlue #1E90FF
Firebrick #B22222	FloralWhite #FFFAF0	ForestGreen #228B22
Fuchsia #FF00FF	Gainsboro #DCDCDC	GhostWhite #F8F8FF
Gold #FFD700	Goldenrod #DAA520	Gray #808080
Green #008000	GreenYellow #ADFF2F	Honeydew #F0FFF0
HotPink #FF69B4	IndianRed #CD5C5C	Indigo #4B0082
Ivory #FFFFF0	Khaki #F0E68C	Lavender #E6E6FA
LavenderBlush #FFF0F5	LawnGreen #7CFC00	LemonChiffon #FFFACD
LightBlue #ADD8E6	LightCoral #F08080	LightCyan #E0FFFF
LightGoldenrodYellow #FAFAD2	LightGray #D3D3D3	LightGreen #90EE90
LightPink #FFB6C1	LightSalmon #FFA07A	LightSeaGreen #20B2AA
LightSkyBlue #87CEFA	LightSlateGray #778899	LightSteelBlue #B0C4DE
LightYellow #FFFFE0	Lime #00FF00	LimeGreen #32CD32
Linen #FAF0E6	Magenta #FF00FF	Maroon #800000
MediumAquamarine #66CDAA	MediumBlue #0000CD	MediumOrchid #BA55D3

Color with Hex Code	Color with Hex Code	Color with Hex Code
MediumPurple #9370DB	MediumSeaGreen #3CB371	MediumSlateBlue #7B68EE
MediumSpringGreen #00FA9A	MediumTurquoise #48D1CC	MediumVioletRed #C71585
MidnightBlue #191970	MintCream #F5FFFA	MistyRose #FFE4E1
Moccasin #FFE4B5	NavajoWhite #FFDEAD	Navy #000080
OldLace #FDF5E6	Olive #808000	OliveDrab #6B8E23
Orange #FFA500	OrangeRed #FF4500	Orchid #DA70D6
PaleGoldenrod #EEE8AA	PaleGreen #98FB98	PaleTurquoise #AFEEEE
PaleVioletRed #DB7093	PapayaWhip #FFEFD5	PeachPuff #FFDAB9
Peru #CD853F	Pink #FFC0CB	Plum #DDA0DD
PowderBlue #B0E0E6	Purple #800080	Red #FF0000
RosyBrown #BC8F8F	RoyalBlue #4169E1	SaddleBrown #8B4513
Salmon #FA8072	SandyBrown #F4A460	SeaGreen #2E8B57
SeaShell #FFF5EE	Sienna #A0522D	Silver #C0C0C0
SkyBlue #87CEEB	SlateBlue #6A5ACD	SlateGray #708090
Snow #FFFAFA	SpringGreen #00FF7F	SteelBlue #4682B4
Tan #D2B48C	Teal #008080	Thistle #D8BFD8
Tomato #FF6347	Transparent #FFFFFF	Turquoise #40E0D0
Violet #EE82EE	Wheat #F5DEB3	White #FFFFFF
WhiteSmoke #F5F5F5	Yellow #FFFF00	YellowGreen #9ACD32

Script differences between Trend Client and ActiveFactory Trend control

The Trend Client maintains the same support for run-time reference binding as all Industrial graphic elements.

The Trend Client properties and methods are very similar to the ActiveFactory Trend control properties and methods, as shown in the following table:

ActiveFactory Trend property or method	Trend Client property or method
(no equivalent)	AddHistorianSource
AddTag	AddPen
AddMultipleTags	Chart.AddMultiplePens
BackColor	Chart.BackgroundColor
(no equivalent)	Chart.Freeze
(no equivalent)	Chart.FreezeDurationMS
(no equivalent)	Chart.HidePenList
(no equivalent)	Chart.Labels
DefaultTagPrecision	Chart.PenPrecision
(no equivalent)	Chart.RefreshEntireChartIntervals
RetrievalOptionsRetrievalMode	Chart.RetrievalMode
LiveModeRate	Chart.UpdateRateMS
ClearTags	ClearPens
DeleteCurrentTag	DeleteCurrentPen
(no equivalent)	GetHistorianSource
CurrentTagValAtX1	GetPenValAtX1
(no equivalent)	GetStartAndEndTimes
(no equivalent)	HistorySource.Authentication
(no equivalent)	HistorySource.Connect
(no equivalent)	HistorySource.Disconnect
(no equivalent)	HistorySource.Domain
(no equivalent)	HistorySource.Password
(no equivalent)	HistorySource.RetainPassword
(no equivalent)	HistorySource.ServerName
(no equivalent)	HistorySource.Type

ActiveFactory Trend property or method	Trend Client property or method
(no equivalent)	HistorySource.UNCPath
(no equivalent)	HistorySource.UserID
Servers	HistorySources
(no equivalent)	HistorySources.Add
(no equivalent)	HistorySources.Count
(no equivalent)	HistorySources.GetSource
(no equivalent)	HistorySources.Items
(no equivalent)	HistorySources.Remove
(no equivalent)	HistorySources.Update
MoveNextTag	MoveNextPen
MovePrevTag	MovePrevPen
CurrentTagColor	Pen.Color
(no equivalent)	Pen.Count
(no equivalent)	Pen.Description
(no equivalent)	Pen.Expression
CurrentTagFormat	Pen.Format
(no equivalent)	Pen.HistorySource
(no equivalent)	Pen.HistoryTagName
CurrentTagIndex	Pen.Index
(no equivalent)	Pen.Name
(no equivalent)	Pen.Precision
CurrentTagPrecision	Pen.RetrievalMode
CurrentTagRetrievalStyle	Pen.Style
(no equivalent)	Pen.TrendHi
(no equivalent)	Pen.TrendLo

ActiveFactory Trend property or method	Trend Client property or method
CurrentTagTrendType	Pen.TrendType
(no equivalent)	Pen.Units
(no equivalent)	Pen.Visible
CurrentTagPenWidth	Pen.Width
CurrentTagRetrievalMode	PenSelectorHeight
(no equivalent)	PenUQRelativeOpacity
(no equivalent)	PenUQRelativeThickness
BackColor	PlotArea.BackgroundColor
BorderColor	PlotArea.BorderColor
PlotGradientEndColor	PlotArea.GradientEndColor
PlotGradient	PlotArea.GradientType
GridColor	PlotArea.GridColor
GridHorizontal	PlotArea.GridHorizontal
(no equivalent)	PlotArea.GridStyle
GridVertical	PlotArea.GridVertical
(no equivalent)	PlotArea.GridWidth
HighlightCurrentTag	PlotArea.HighlightCurrentPen
(no equivalent)	PlotArea.PenHighlightColor
(no equivalent)	PlotArea.PenHighlightWidth
SingleTagMode	PlotArea.SingleTagMode
PlotImage	PlotImage
RefreshData	RefreshData
(no equivalent)	RefreshTimes
(no equivalent)	RemoveHistorianSource
RemoveTag	RemovePen

ActiveFactory Trend property or method	Trend Client property or method
ScaleAllTags	ScaleAllPens
ScaleAutoAllTags	ScaleAutoAllPens
ScaleAutoTag	ScaleAutoPen
ScaleDownAllTags	ScaleDownAllPens
ScaleDownTag	ScaleDownPen
ScaleMoveAllTagsDown	ScaleMoveAllPensDown
ScaleMoveAllTagsUp	ScaleMoveAllPensUp
ScaleMoveTagDown	ScaleMovePenDown
ScaleMoveTagUp	ScaleMovePenUp
ScaleTag	ScalePen
ScaleUpAllTags	ScaleUpAllPens
ScaleUpTag	ScaleUpPen
SetCurrentTag	SetCurrentPen
SetDuration	SetDuration
(no equivalent)	SetStartAndEndTimes
AllowContextMenu	ShowContextMenu
SupressErrors	SuppressErrors
XCursor1Color	TimeAxis.Cursor1.Color
XCursor1Pos	TimeAxis.Cursor1.Pos
(no equivalent)	TimeAxis.Cursor1.Style
(no equivalent)	TimeAxis.Cursor1.Width
XCursor2Color	TimeAxis.Cursor2.Color
XCursor2Pos	TimeAxis.Cursor2.Pos
(no equivalent)	TimeAxis.Cursor2.Style

ActiveFactory Trend property or method	Trend Client property or method
(no equivalent)	TimeAxis.Cursor2.Width
NumDataPointLabels	TimeAxis.LabelColor
NumTimeAxisGridPerValue	TimeAxis.NumGridPerValue
NumTimeAxisValues	TimeAxis.NumValues
(no equivalent)	TimeAxis.ShowCursors
(no equivalent)	TimeSelector
(no equivalent)	TimeSelector.DurationMS
(no equivalent)	TimeSelector.EndDate
(no equivalent)	TimeSelector.GetStartAndEndTimes
(no equivalent)	TimeSelector.RefreshTimes
(no equivalent)	TimeSelector.SetStartAndEndTimes
(no equivalent)	TimeSelector.StartDate
TimeSelector.TimeDuration	TimeSelector.TimeDuration
ToolTipText	ToolTipText
(no equivalent)	TrendVersion
(no equivalent)	UpdateHistorianSource
NumXValueAxisLabels	ValueAxis.Label
NumYAxisGridPerValue	ValueAxis.NumGridPerValue
NumYAxisValues	ValueAxis.NumValues
Visible	Visible

Data retrieval

The Trend Client supports both Cyclic and Full retrieval modes when retrieving data from Historian. This appendix explains how both retrieval modes work.

Retrieval modes

Different retrieval modes enable you to access data stored on an Historian in different ways. For example, if you retrieve data over a long trend duration, you can use equal length data collection intervals to minimize response time. For a shorter period, you might want to retrieve all stored values to produce more accurate trends.

Although Historian with a version of 9.0 or higher supports a variety of retrieval modes, the Trend Client only supports Cyclic and Full data retrieval.

Cyclic retrieval

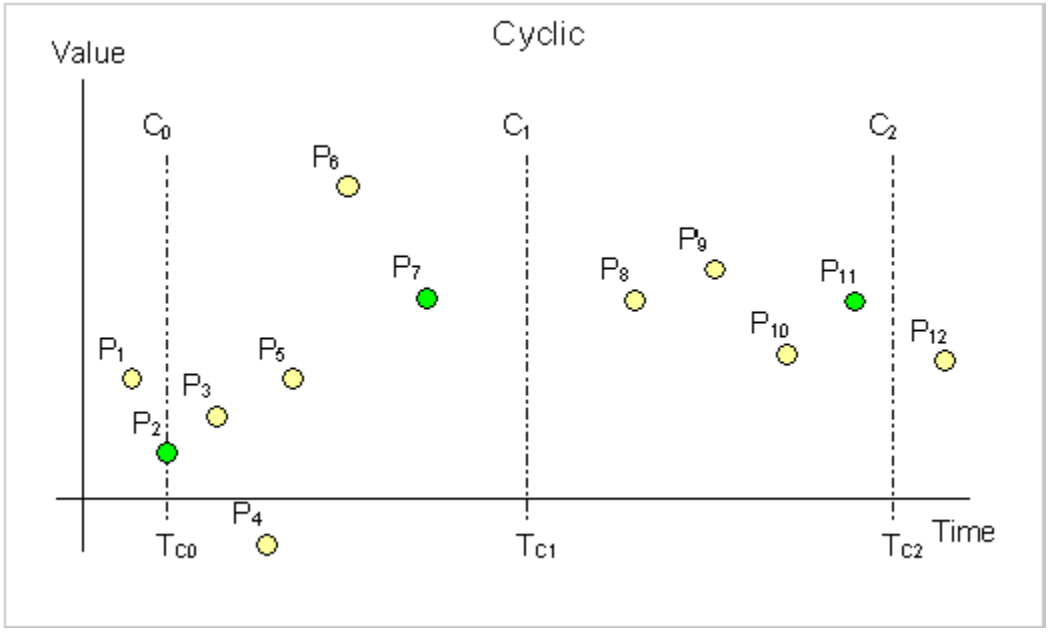
Cyclic retrieval mode returns stored data over a trend duration at equal length cyclic intervals. In Cyclic retrieval mode, a single value is returned at the beginning of each cyclic interval.

If no data is stored at the start of a cyclic interval, the last recorded value before the start of an interval is returned. If two consecutive intervals have the same value, no value is returned for the second interval.

Cyclic retrieval works with all types of tags and produces a virtual rowset, which may or may not correspond to the actual data rows stored on the Historian.

The length of a cyclic interval is dynamic and determined by the duration of a trend and the resolution of a trend in pixels: 1 cyclic interval for every 2 pixels.

The following graph shows how tag values are retrieved from the Historian using Cyclic retrieval mode.



Data is retrieved over a period starting at T_{C0} and an ending at T_{C2}. Each dot in the graph represents an actual data point stored in the Historian. The Historian can return data from three cyclic intervals at T_{C0}, T_{C1}, and T_{C2}.

The following data points are returned by Cyclic retrieval mode:

- At T_{C0}: P_2, because it falls on the start boundary of an interval
- At T_{C1}: P_7, because it is the last data point before the start of the next cyclic interval
- At T_{C2}: none because P7 and P11 are the same value

Cyclic retrieval mode is fast and consumes little Historian resources. However, cyclic retrieval mode may not accurately reflect stored data because important process values (data gaps, value spikes) may fall outside of the retrieval intervals and not appear as values shown in the trend.

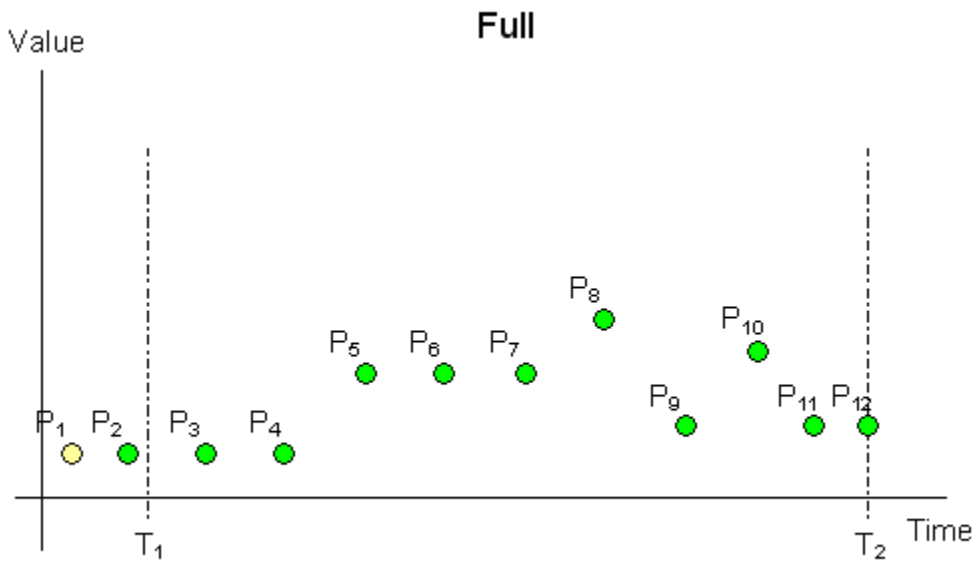
Full retrieval

In Full retrieval mode, all stored data points are returned, regardless of whether a value or quality has changed since the last value. This mode enables the same value and quality pair (or NULL value) to be returned consecutively with their actual timestamps. It works with all types of tags.

By using Full retrieval in conjunction with storage without filtering (that is, no delta or cyclic storage mode is applied at the historian), you can retrieve all values that originated from the plant floor data source or from another application.

Full retrieval best represents the process measurements recorded by the Historian. However, it imposes a higher load on the server, the network, and the client system because a very large number of records may be returned for longer time periods.

The following illustration shows how values are returned for Full retrieval:



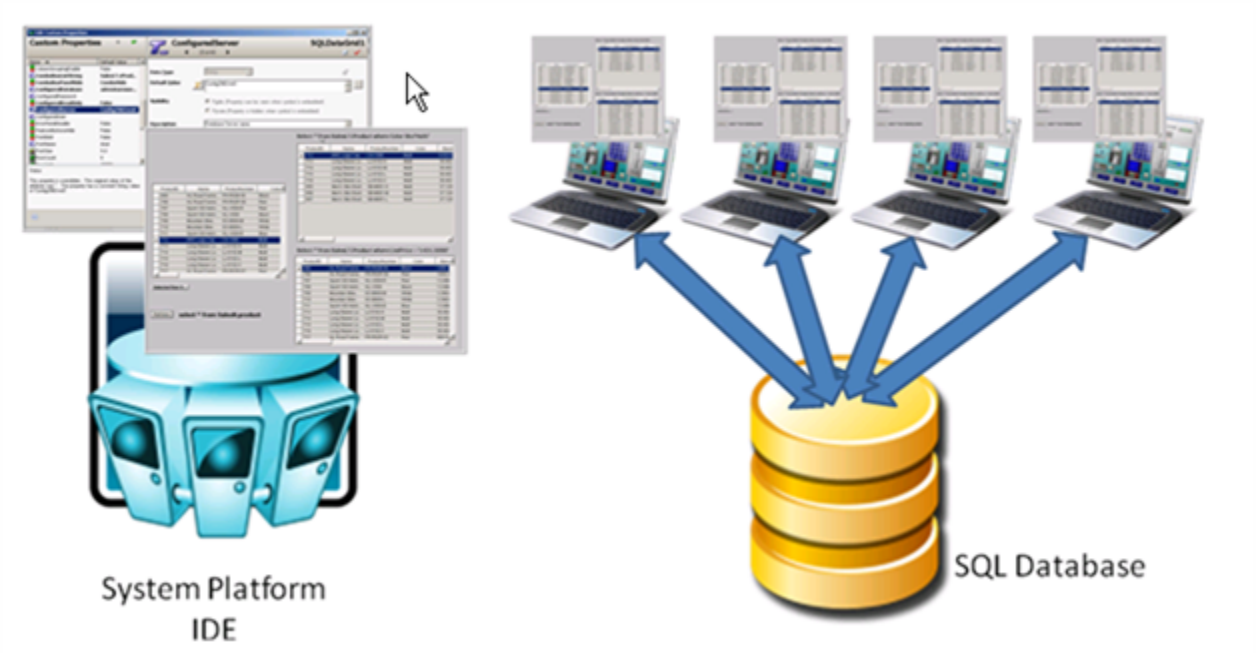
Data is retrieved in full mode with a start time of T_1 and an end time of T_2. Each dot in the graphic represents an actual data point stored on the historian. From these points, the following are returned:

- P_2, because there is no actual data point at T_1
- P_3 through P_{12}, because they represent stored data points during the time period

The SQLDataGrid graphic

The SQLDataGrid is a graphic that enables you to interact with data stored in SQL databases. You can use the SQLDataGrid graphic in an InTouch application that you create in a published or managed mode.

The following picture shows the main components associated with the SQLDataGrid.



You can perform the following activities with the SQLDataGrid graphic:

- Insert or modify data in selected database tables and views.
- Delete rows from a database table (select the row and press the **Delete** key).
- Select data from existing tables and views.
- Run a custom SQL query.

This section provides the following information:

- The minimum configuration required by the SQLDataGrid graphic to retrieve data from a SQL database.
- The set of custom properties that must be configured to authenticate users and connect to the database.
- A list of the custom properties that define the parameters associated with a specific database and query to be retrieved.

Secure access to the SQLDataGrid

The SQL Data Grid is a powerful tool which allows InTouch WindowViewer to connect to various SQL databases. This tool allows for reading as well as modifying SQL databases, including their data, tables, table schemas, and stored procedures. Given this, it is important that the SQLDataGrid and any SQL database(s) it will access are

configured securely.

Security best practices

1. Grant each user the least privileges required within each SQL database

   Grant users in the SQL database the least privileges required to accomplish their job. This can include limiting their access to specific databases as well as preventing modification to the database. The user should only have the privilege needed to perform their required actions. In the SQLDataGrid, the user will only be able to perform actions that their role allows them to. For SQL authentication and Windows impersonation, this is the user whose credentials have been provided to the SQLDataGrid. For Windows integrated authentication, this is the currently logged-on user.
2. Consider granting users read-only access within each SQL database

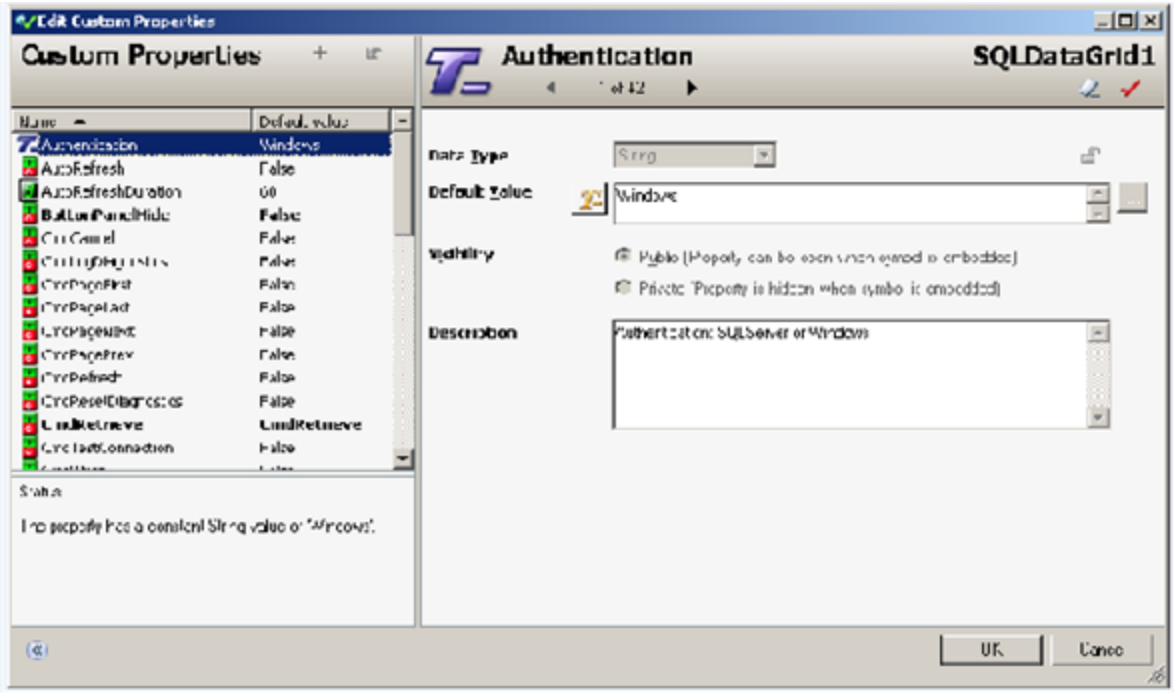
   Grant users read-only access in the SQL database when possible. This will prevent unwanted modification of the database. The SQLDataGrid can operate with a user who has read-only privilege and will only allow the user to retrieve and view the data.
3. Configure SQLDataGrid to use Windows Integrated Authentication

   Configuring the SQLDataGrid to use Windows integrated authentication helps to secure your application. Using SQL authentication and Windows impersonation require the storage of user credentials and increases the attack surface of the application. With Windows Integrated Security, the SQLDataGrid will connect to the SQL database using the context of the currently logged on Windows OS user.
4. Be aware of what SQLDataGrid can do

   As an application developer/maintainer, it’s your responsibility to be fully aware that the SQLDataGrid is not a read-only control, but instead it allows users to execute full SQL statements on the database, including UPDATE, CREATE, DROP, etc. There is a potential risk involved with using the SQLDataGrid if it, or the SQL database it is accessing are not configured securely.

The SQLDataGrid graphic

The SQLDataGrid graphic appears in the **Visualization** folder of the System Platform IDE. Embed the graphic in an existing window or a new window. When you double-click the graphic, the **Edit Custom Properties** dialog box appears.



For detailed information on working with graphics, see the *ArchestrA and InTouch Integration Guide*.

Configure authentication modes

This section describes how to use the custom properties of the SQLDataGrid graphic to authenticate a user to the database. You can assign static values to these custom properties or link them to attributes or tag names in your application.

Note: The value fields are not case sensitive.

To configure an authentication mode for the SQLDataGrid graphic

1. In the **Custom Properties** pane, select **Authentication**.
2. In the **Default Value** box, type the name of a valid security type. Acceptable values are:
   - Windows
   - SQL Server

   A more detailed description of these authentication modes follows this procedure.
3. In the **Custom Properties** pane, select **ConfiguredUser**.
4. In the **Default Value** box, type a user name.
5. In the **Custom Properties** pane, select **ConfiguredPassword**.
6. In the **Default Value** box, type the password associated with the user name.

Windows authentication

If you use Windows as a value for the **Authentication** custom property and do not designate custom properties for a domain, user, or password, the SQLDataGrid graphic uses the credentials of the user who is currently logged on to the node running InTouch.

If you select Windows as a value for the **Authentication** custom property and designate custom properties for a domain, user, and password, the SQLDataGrid graphic uses the specified Windows domain name, user name, and password to connect to the SQL Server database.

SQL Server

This authentication mode uses the specified SQL Server user name and password to connect to the SQL Server database.

Configure the database connection

This section describes the procedure to configure the custom properties required to establish a connection to a specific database and its associated data. You can assign static values to these custom properties or link them to attributes or tag names in your application.

Note: The value fields are not case sensitive.

If you are using a non-default port for the SQL Server database, see the SQLData Object help file for configuration information.

To configure the database connection

1. In the **Custom Properties** pane, select **ConfiguredServer**.
2. In the **Default Value** box, type the network name of the computer hosting the database.
3. In the **Custom Properties** pane, select **ConfiguredDatabase**.
4. In the **Default Value** box, type the name of the target database.
5. In the **Custom Properties** pane, select **SQLQuery**.
6. In the **Default Value** box, type the name of the target table, view or query.

To populate the SQLDataGrid, click **Runtime**, and then click **Retrieve**.

Configure SQLDataGrid graphic custom properties

This section lists custom properties in functional groups. For a detailed description of each custom property, see Custom Properties for the SQLDataGrid Graphic.

The default run-time window shows the complete graphic. However, you can hide the interface controls and display only the grid.

You can use custom properties to display additional information about the status of the query, the dataset, or to provide interactive buttons that initiate specific actions

In addition to the table, view, and query input, you can specify additional parameters by using custom properties and control them with scripting.

The following table lists the various custom properties you can insert in your scripts by functional groups.

Authentication	Authenticate users to databases	**Authentication**

		ConfiguredDomain
		ConfiguredPassword
		ConfiguredUser
Database Connectivity	Designate database connectivity functions	
		AutoRefresh
		AutoRefreshDuration
		CmdCancel
		CmdRefresh
		CmdRetrieve
		CmdTestConnection
		CmdWrite
		ComboBoxListString
		ConfiguredDatabase
		ConfiguredReadOnly
		ConfiguredServer
		RowCount
		RowLimit
		SelectedRowNumber
		SQLQuery
		SQLQueryCurrent
		SQLQueryPassThrough
		SQLQueryPending
Diagnostic	Perform diagnostic functions	
		CmdLogDiagnostics
		CmdResetDiagnostics
		ErrorPanelDisable
		StatusDesc

		StatusError
		StatusIsError
Appearance	Modify the appearance of the run-time SQLDataGrid that contains data from the database	
		ButtonPanelHide
		CmdPageFirst
		CmdPageLast
		CmdPageNext
		CmdPagePrev
		ColumnAggregateEnable
		ColumnFiltering
		ColumnGroupingEnable
		ComboBoxPanelHide
		FontBold
		FontName
		FontSize
		SupportThemes
		SuppressAuthenticationWarning
		WriteButtonHide

Scripts with the SQLDataGrid

When you use the **SQLDataGridUserCtl** to create a new graphic, you can script all the properties and events of the control.

Associate grid control actions with InTouch scripts and animations

You can hide the buttons of the SQLDataGrid graphic and trigger data retrieval from outside the grid interface. For example, you can configure a button to generate a dataset from an InTouch application.

To do this, use the **CmdRetrieve** property and associate it to an InTouch tag. The tag is linked to the InTouch button.

To configure dataset retrieval from InTouch

1. In WindowMaker, open the InTouch window.
2. Double-click the SQLDataGrid graphic. The **Edit Custom Property** dialog box appears.
3. In the **Custom Properties** panel, select the **ButtonPanelHide** property.
4. In the **Default Value** field, type **True**.
5. Select the **CmdRetrieve** property.
6. In the **Default Value** field, select an InTouch tag by one of the following actions:
   a. Type a tag name. If the tag does not exist, you are prompted to define a new tag.
   b. Click the ellipsis to open the Tagname Dictionary and select a tag.

   For this example, type **CmdRetrieve** in the field.
7. Click **OK** and define the tag as a Memory Discrete tag.
8. When you finish defining the tag, close the Tagname Dictionary and click **OK** on the **Custom Properties** panel.
9. Create a button on your InTouch window. For this example, its caption is **Retrieve**.
10. Link the **CmdRetrieve** tag to the button.
11. Switch to run-time mode and click **Retrieve**.

Access data from a selected row in the SQLDataGrid

You can use the SQLDataGrid graphic to read the content of a selected row and associate it to attributes outside the graphic. You can configure scripts to read the data from a selected row and programmatically associate that data to tags in an InTouch application or to attributes in the Galaxy.

The following script example shows how to retrieve data from a selected row. Notice that the **SelectedRowData** property is available in the SQLDataGrid client control and not in the SQLDataGrid graphic.

```
dim rd1 as System.Data.DataRow;
rd1 = SQLDataGridUserCtrl1.SelectedRowData;
if(rd1 <> null) then
InTouch:Text9 = rd1.ItemArray[4].ToString();
'This assigns the 4th column value for the selected row, to the InTouch "Text9" tag.
endif;
```

The SQLDataGrid in runtime

During run time, you can select the database table, view, or query that will populate the SQLDataGrid graphic and change the appearance of the data.

You can use the run-time application to perform the following tasks:

- Move columns
- Group columns
- Aggregate column values
- Filter columns

- Sort columns

If you have the appropriate permissions, you can also make the following changes to the database:

- Change the order of columns in an existing row.
- Insert a row into a table.
- Update a row in a table.
- Delete a row from a table.

Note: You cannot update or delete a row in a table unless that table has a primary key defined. You can add new rows to a table without a primary key as long as that is the only action you are performing on the table.

About the SQLDataGrid in runtime

During run time, you can select the database table, view, or query that will populate the SQLDataGrid graphic and change the appearance of the data.

You can use the run-time application to perform the following tasks:

- Move columns
- Group columns
- Aggregate column values
- Filter columns
- Sort columns

If you have the appropriate permissions, you can also make the following changes to the database:

- Change the order of columns in an existing row.
- Insert a row into a table.
- Update a row in a table.
- Delete a row from a table.

Note: You cannot update or delete a row in a table unless that table has a primary key defined. You can add new rows to a table without a primary key as long as that is the only action you are performing on the table.

View data

You can use the run-time version of the SQLDataGrid graphic to view data in the following formats:

- Data from a specific table
- Data from a specific view
- Data that is retrieved by a SQL query

You can resize the column width of any table, view, or query result.

To view specific data from a SQL query

1. In the **Table, View or Query** box, type a SQL query.
2. Click **Retrieve**. The appropriate dataset appears.

Populate the Table, View, or Query list

You can also populate the Table, View or Query box with items that you frequently use, so that you do not repeatedly have to type entries.

To populate the Table, View, or Query box

1. In the **Edit Custom Properties** dialog box, click **ComboBoxListString**.
2. In the **Default Value** box, type any of the following items, separated by semicolons (;):
   - Multiple table names
   - Multiple SQL statements
   - A combination of table names and SQL statements
3. Click **OK**.
4. When you go to run time, click the arrow to the right of the **Table, View or Query** box. Each item that you entered is displayed on a separate line.

Hide and show window issues

If you hide the InTouch window that contains the SQLDataGrid and then show it again, the last data values contained in the window might not appear. Use the following procedure to ensure that the data values always appear.

To ensure that grid values appear when you hide and show the InTouch window

1. In WindowMaker, click **Special > Configure > WindowViewer**.
2. Under **WindowViewer Memory**, clear the check box **Always load windows from disk**.

Move columns

You can rearrange the order of the columns to meet your viewing needs.

To move columns

1. Click the heading of the column that you want to move.
2. Holding your mouse key down, drag the column heading to another position within the table.

Group columns

The group-by function lets you sort data at more specific levels. This example shows a view that groups customer names and their business addresses.

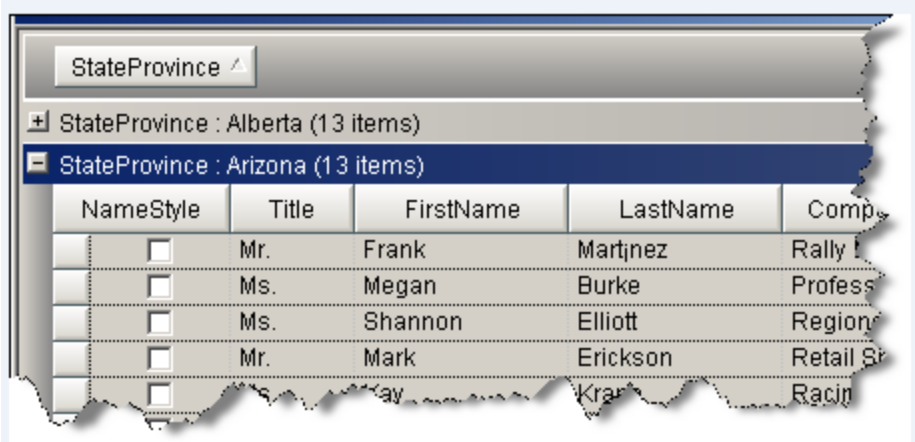
Note: The ColumnGroupingEnable custom property must be True before you can use this function.

To use the Group-by function

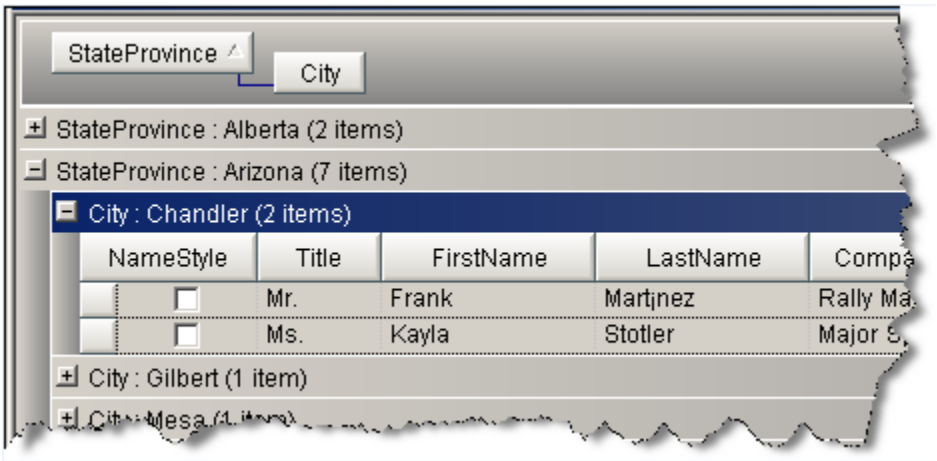
1. In the **Table, View or Query** box, type the name of a table or view or click the arrow to display a list of pre-determined items. For details about pre-determined items, see View Data.
2. Click **Group-by**. The **Drag a column header** area appears.

3. Drag a column heading to the upper area.

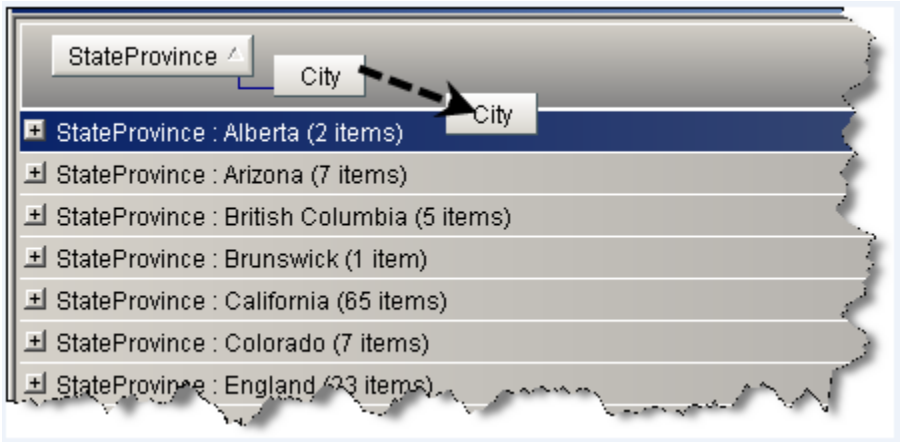
4. To view the data groups, press **Enter** to expand any of the **StateProvince** items.



5. To see a more specific sorting of the data, drag another column heading to the upper sorting area.



6. If you want to sort the data at an even finer level, you can repeat steps 3 and 4 and drag another column heading to the sort area.
7. If you want to remove one of the column headings from the sorting area, drag it down to the data area.



8. To return the original dataset, click **Group-by**.

View aggregate values

You can perform a numeric operation on data within a single column of the table by clicking the **Aggregate** button.

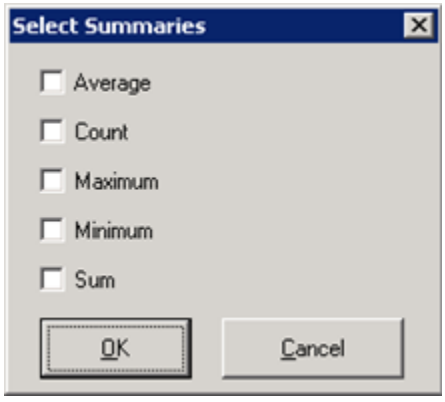
Note: Before you can use this function, the ColumnAggregateEnable custom property must be True.

You can perform the following functions on column data. Not all operations may be available for a column because of the data type.

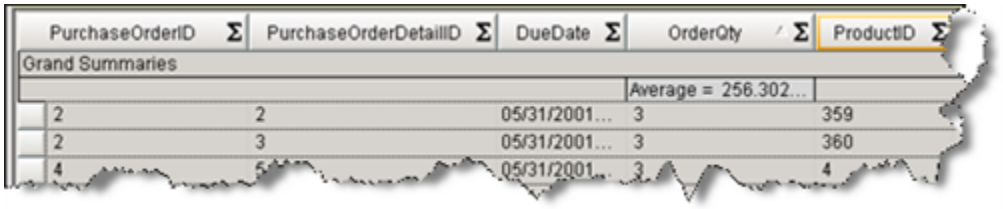
- Average: Calculates the average number of items in the column.
- Count: Lists the number of rows in the column.
- Maximum: Lists the highest number in the column.
- Minimum: Lists the lowest number in the column.
- Sum: Adds the numbers in each row and provides a total.

To view aggregate values

1. Click **Aggregate**. The summation sign (S) appears at the head of each column.
2. Click the summation sign at the top of the **OrderQty** column. The **Select Summaries** dialog box appears.



3. Select **Average** and click **OK**. The value appears just below the **Grand Summaries** row. You can re-size the column width to view the entire number This following figure shows the average value for **OrderQty**.



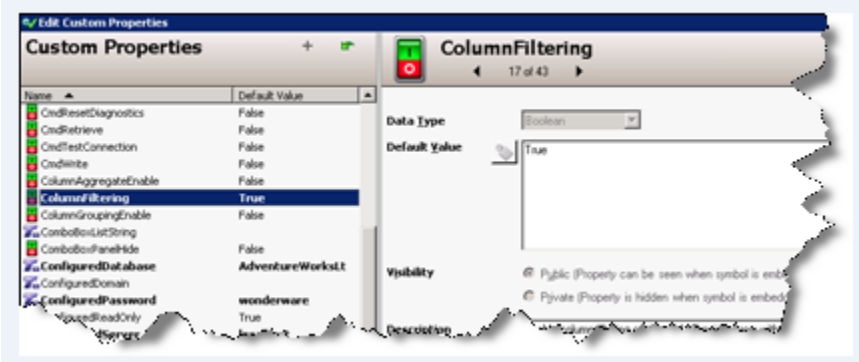
4. To hide the **Grand Summaries** row follow these steps:
   a. Click the summation sign.
   b. Clear the **Average** check box in the **Select Summaries** dialog box
   c. Click **OK**.
5. To hide the summation signs, click **Aggregate**.

Progressive column filtering

You can configure the SQLDataGrid to support progressive column filtering. Type the first letters of your search, and the matching values appear in the grid.

To configure and use progressive filtering

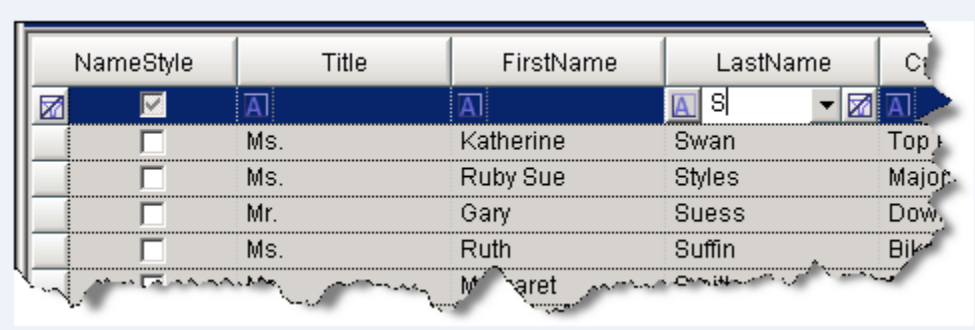
1. Switch your SQLDataGrid window to WindowMaker mode.
2. Double-click the graphic and select the custom property **ColumnFiltering**.
3. Set the **ColumnFiltering** default value to **True**.



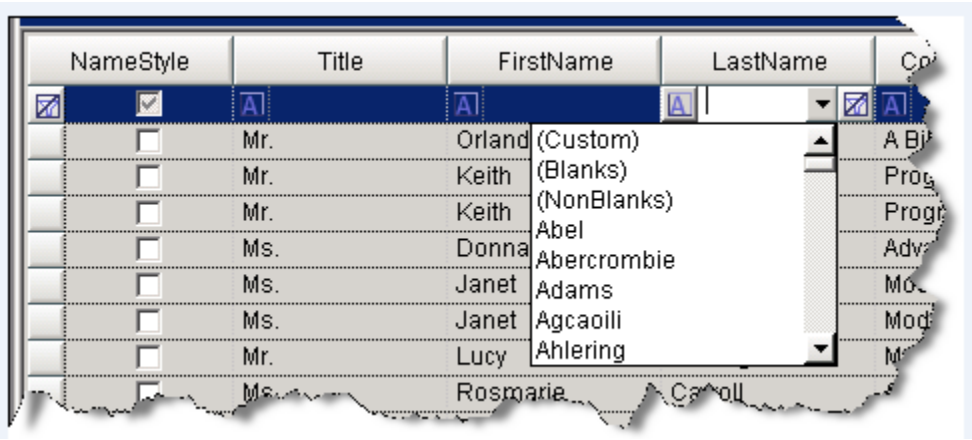
4. Click **OK** and run the InTouch application.
5. To return a dataset, click **Retrieve**. In this example, the dataset contains a table view of customer names and addresses.

6. Move your cursor over the field below the **LastName** column heading. A list arrow appears.
7. You can filter the values by any of the following actions:

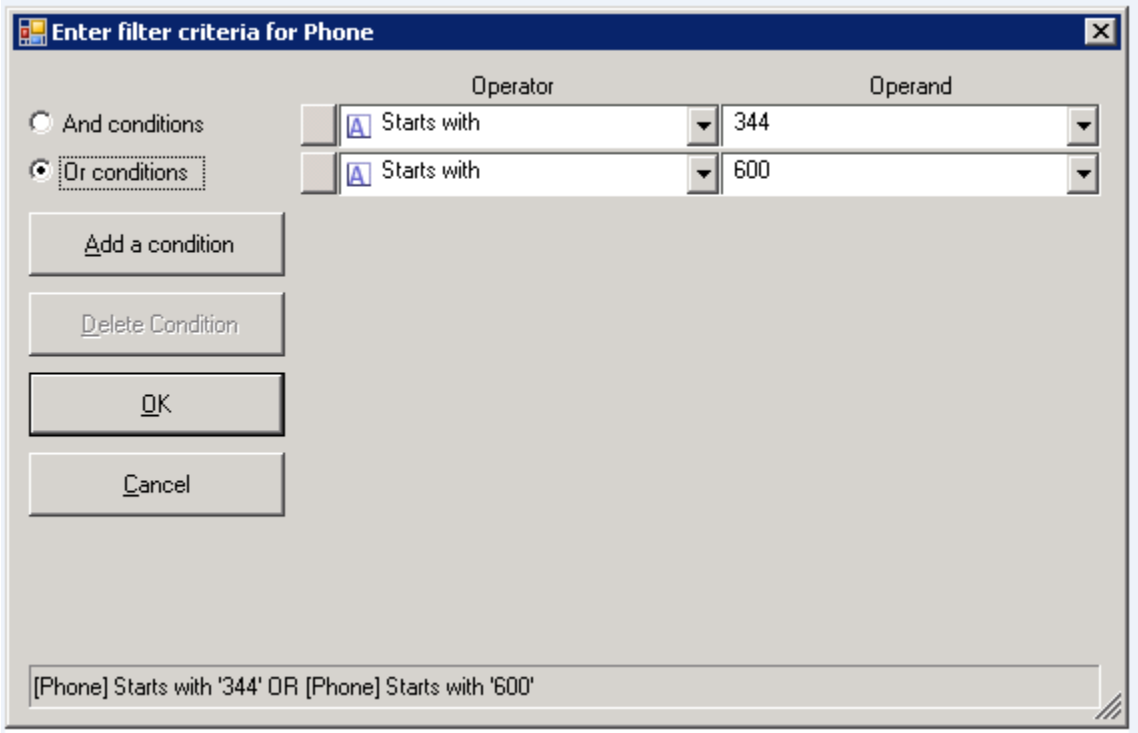
- Click the list arrow and type the first letter of the value If you type additional letters, your sort is more precise.



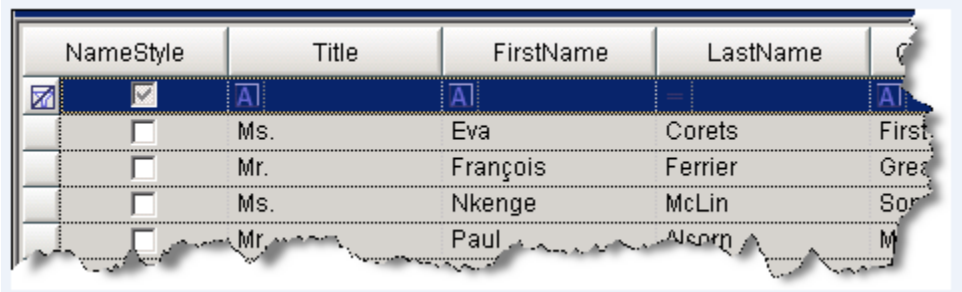
- Click the list arrow and select an item from the list.



- If you select the **(Custom)** item from the list, you can further refine your search in the **Custom** dialog box. This example shows a search for two different area codes.



8. To perform a different search, clear your search criteria by clicking the cancel icon at the left of the filtering row.



Discard changes

If you need to discard the changes made to grid values, click **Retrieve**. This action reloads the table values from the database and discards any changes. You can also initiate the action by using a script that sets the **CmdRetrieve** property to **True**.

View errors

You can manually display or close the **Status** panel in the lower part of the grid display area. The status panel can also automatically appear when an error occurs. However, after the panel appears, it remains on view.

If a command object is successful, the message "Success" appears in the Status panel. If the panel is closed, "Success" messages do not appear for successful command objects.

To close the panel, click the double-arrow icon at the right of the panel. To open the panel on demand, right-click and select **Show Status**.

Note: The error panel display does not change the overall area occupied by the grid graphic.

Write changes to the database

You can write changes to the database if the following conditions are met:

- The database query (or table) can be updated.
- The **ConfiguredReadOnly** custom property value is **False**.
- SQL Server security allows the security mode configured at the grid to perform an update.
- Changes made in the grid are consistent with the database schema.

Changes made in the grid are written to the database using the Write command. Changes are handled as a single database transaction. If any change fails, the entire set of changes is rolled back.

If you have sufficient privileges, you can change a value and click **Write**. You can also use a script to set the **CmdWrite** property. This action causes the grid contents to be evaluated for changes and for the changes to be

written to the database.

Tip: After you modify cells in the grid and write them back to the database, you should also verify that the data is correctly written. In certain cases, such as a database table with an automatically incrementing primary key or interdependent cells, the grid does not show the true contents of the database table after you write back to it. Also, no error message is shown. It is recommended that you execute a retrieve or auto refresh command after writing to the database to make sure it is showing the current data.

Views and updates

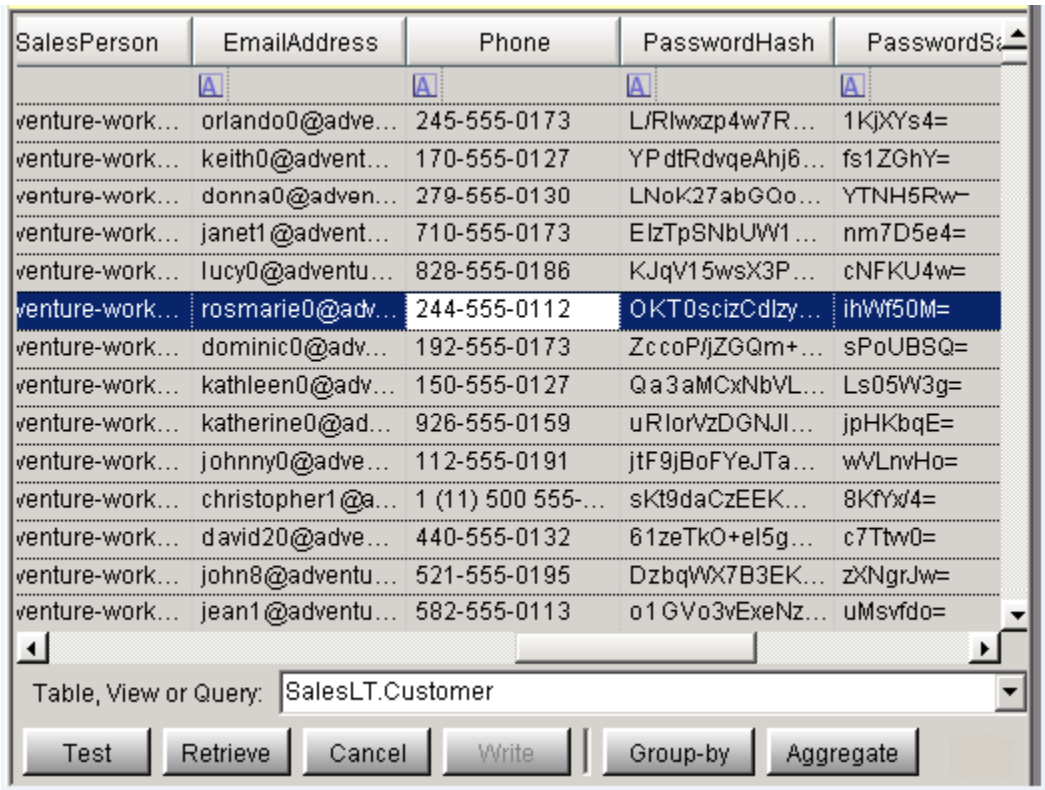
Some views cannot be updated for the following reasons:

- The views contain table joins on keys other than primary keys.
- The views include security settings that prohibit updates.

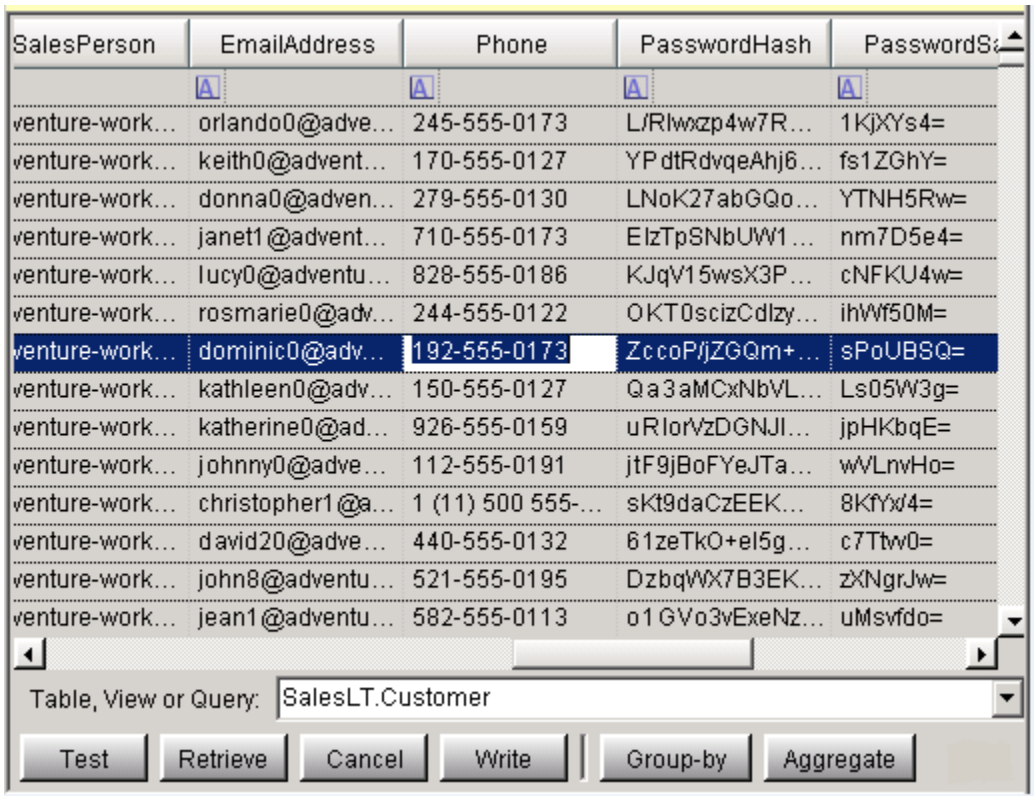
If the write process fails, you can view the status in the **Error** panel. You can also monitor the **StatusDesc**, and **StatusError** properties by using text fields on the InTouch window for run-time display.

To make a change to data in the data grid

1. Ensure that the **ConfiguredReadOnly** custom property value is **False**.
2. Click **Runtime**.
3. Enter a SQL statement or the name of a database table or view.
4. Click **Retrieve**. This example shows data from the customer table in AdventureWorks. You will modify a phone number.
5. Select the phone number that you must modify. The area becomes editable.



6. Change the last four digits of the phone number to 0122.
7. Click an area outside the edit area. The **Write** button becomes enabled.



8. Click **Write**.
9. To verify the change, click **Retrieve**.

Database connection management

An important feature of the SQLDataGrid is the ability to optimize the management of multiple simultaneous connections to the database. This section describes how the infrastructure handles the database connection pool and sharing for the SQLDataGrid.

Pool and shares connections

Multiple instances of the SQLDataGrid configured with identical connection strings use a shared connection pool and SQL command queue when they are active in the same instances of the InTouch WindowViewer. To limit the size of the shared pool, you can use the connection pool property **MaxConnections**. For configuration options, see Custom Properties for the SQLDataGrid Graphic.

Physical connections to the SQL database are opened based on command requests and are closed when no requests are pending. All SQL commands are queued and processed based on availability of connections and processing time.

The SQLDataGrid asynchronously processes all SQL commands. Asynchronous command processing ensures availability of interaction with the InTouch WindowViewer during time-consuming command processing.

Any database connection that is established in a script within a graphic using the SQLData Script Library also

takes advantage of connection pooling. Similar connection strings from either the SQLDataGrid or the script in the same InTouchView application share a common pool.

Note: Connection sharing and connection pooling require that the same security mode and user credentials be used for all SQLDataGrid graphics within the InTouchView application that shares the connection or the pool.

Custom properties for the SQLDataGrid graphic

The following table lists all available Custom Properties. Data types are shown in the Custom Properties panel and are not included in this table.

Property Name	Description
Authentication	Designates whether database user authentication is SQL Server or Windows. **Writable at Run Time**: Yes **Default**: Windows
AutoRefresh	Automatically refreshes data appearing in the SQLDataGrid. When True, the query that repopulates the SQLDataGrid runs after the **AutoRefreshDuration** time elapses. **Writable at Run Time**: Yes **Default**: False
AutoRefreshDuration	Designates the requested time interval in seconds for AutoRefresh. The actual refresh rate can be slower than the specified rate. **Writable at Run Time**: Yes **Default**: 60
ButtonPanelHide	Hides grid buttons when the value is True. **Writable at Run Time**: Yes **Default**: False
CmdCancel	Cancels the last database query. **Writable at Run Time**: Yes **Default**: False
CmdClear	Clears the contents of the SQLDataGrid. **Writable at Run Time**: Yes **Default**: False

Property Name	Description
CmdCopy	When True, copies the current selection to the clipboard. **Writable at Run Time**: Yes **Default**: False
CmdLogDiagnostics	Writes the current DCM diagnostic values to the logger. **Writable at Run Time**: Yes **Default**: False
CmdPageFirst	Moves the SQLDataGrid display to the first row. **Writable at Run Time**: Yes **Default**: False
CmdPageLast	Moves the SQLDataGrid display to the last row. **Writable at Run Time**: Yes **Default**: False
CmdPageNext	Moves the SQLDataGrid display down one page. **Writable at Run Time**: Yes **Default**: False
CmdPagePrev	Moves the SQLDataGrid display up one page. **Writable at Run Time**: Yes **Default**: False
CmdPaste	When True, pastes the current text contained on the clipboard to the grid. **Writable at Run Time**: Yes **Default**: False
CmdRefresh	Refreshes the current SQL query data. Any pending changes are discarded. **Writable at Run Time**: Yes **Default**: False
CmdResetDiagnostics	Resets the current DCM diagnostic values and clears the message and event counters. **Writable at Run Time**: Yes **Default**: False

Property Name	Description
CmdRetrieve	Loads the selected table, view, or query to the SQLDataGrid. Any changes to a previous dataset are discarded. **Writable at Run Time**: Yes **Default**: False
CmdTestConnection	Tests the database connection for the specified server, database, security type; and if applicable, user name and password. The result is placed in the Error panel. **Writable at Run Time**: Yes **Default**: False
CmdWrite	Submits changes made in the SQLDataGrid to the database. **Writable at Run Time**: Yes **Default**: False
ColumnAggregateEnable	Specifies whether the column aggregate feature is exposed by default. **Writable at Run Time**: Yes **Default**: True.
ColumnFiltering	Enables column filtering and provides a row with the progressive filtering entry field. **Writable at Run Time**: Yes **Default**: False
ColumnGroupingEnable	Specifies whether the column grouping feature of the SQLDataGrid is exposed by default. **Writable at Run Time**: Yes **Default**: False
ComboBoxListReadOnly	If False, you can make changes to the edit box above the combo list box. **Writable at Run Time**: Yes **Default**: False
ComboBoxListString	Enables a list of table or view names, separated by semicolons to be viewable in the **Runtime** combo box. **Writable at Run Time**: Yes **Default**: Blank

Property Name	Description
ComboBoxPanelHide	Hides the combo box and panel. **Writable at Run Time**: Yes **Default**: False
CommandTimeout	The number of seconds that the SQLDataGrid waits for a pending command to run before timing out. **Writable at Run Time**: Yes **Default**: 30
ConfiguredDatabase	The name of the database that connects to the SQLDataGrid. **Writable at Run Time**: Yes **Default**: Blank
ConfiguredDomain	The domain to use with impersonation if the authentication mode is Windows. **Writable at Run Time**: Yes **Default**: Blank
ConfiguredPassword	SQL Server domain if the security type is Windows. **Writable at Run Time**: Yes **Default**: ********
ConfiguredReadOnly	Prevents the SQLDataGrid from making changes to the database. **Writable at Run Time**: Yes **Default**: True
ConfiguredServer	The name of the computer hosting the database. **Writable at Run Time**: Yes **Default**: Blank
ConfiguredUser	The SQL Server user name if security type is SQL. This property is optional when security type is Windows. **Writable at Run Time**: Yes **Default**: Blank
ConnectionTimeout	The number of seconds that the SQLDataGrid tries to connect to a database before timing out. **Writable at Run Time**: Yes **Default**: 15

Property Name	Description
ContextMenuEnable	When True and you right-click the grid, the context menu appears. When False and you right-click the grid, the context menu does not appear. **Writable at Run Time**: Yes **Default**: True
ErrorPanelDisable	Disables the error message display. **Writable at Run Time**: Yes **Default**: False
ErrorPanelShow	When True, the error panel is visible. When False, the error panel is hidden. **Writable at Run Time**: Yes **Default**: False
FeatureButtonsHide	Hides the **Aggregate** and **Group-by** buttons. **Writable at Run Time**: Yes **Default**: False
FontBold	Specifies whether to display the SQLDataGrid font in bold text. **Writable at Run Time**: Yes **Default**: False
FontName	Specifies the name of the font. **Writable at Run Time**: Yes **Default**: Arial
FontSize	Specifies the point size of the font. **Writable at Run Time**: Yes **Default**: 9.0
InstanceName	A name that you choose to identify a specific SQLDataGrid instance. The name appears in the **ConnectionName** diagnostic. **Writable at Run Time**: Yes **Default**: Blank

Property Name	Description
MaxConnections	The number of active connections allowed in the connection pool for a given connection string. **Writable at Run Time**: Yes **Default**: 100
RowCount	The row count returned in the last successful data request. **Writable at Run Time**: No **Default**: 0
RowLimit	Modifies the query to return a specified number of records. To prevent the database query from being modified, set the **SQLQueryPassThrough** custom property to True. **Writable at Run Time**: Yes **Default**: 20,000
RowsChanged	If True, one or more rows have been changed in the grid (added, odified, or deleted). If False, no rows were changed in the grid. **Writable at Run Time**: No **Default**: False
SelectedRowNumber	The number of the selected grid row. **Writable at Run Time**: Yes **Default**: 0
SQLQuery	Specifies the SQL query, table, or view data to be retrieved to the SQLDataGrid. If **SQLQueryPassThrough** is True, you must add the SELECT statement along with the table or view name. For example, SELEC*T * FROM person.contact*, rather than *person.contact*. **Writable at Run Time**: Yes **Default**: Blank
SQLQueryCurrent	Specifies the SQL command that was run to show the current data in the SQLDataGrid. **Writable at Run Time**: No **Default**: Blank

Property Name	Description
SQLQueryPassThrough	If True, enables the SQL statement specified in the **SQLQuery** custom property to be passed directly to the database without modification. When this property is True and the **SQLQuery** custom property is set only to a table name or view, the grid does not insert "SELECT * FROM" in front of the table or view name. Additionally, if this property is True, the top "n" is not inserted into a SELECT statement (where "n" is the value of the RowLimit custom property). **Writable at Run Time**: Yes **Default**: False
SQLQueryPending	Indicates a pending query. **Writable at Run Time**: No **Default**: False
StatusDesc	Specifies the error description from the last query. If no error exists, the string is empty. **Writable at Run Time**: No **Default**: Blank
StatusError	Specifies the error number from the last request. **Writable at Run Time**: No **Default**: 0
StatusIsError	Indicates whether an error occurred in the last query. True = Error False = no error **Writable at Run Time**: No **Default**: False
SupportThemes	Specifies the SQLDataGrid window is formatted with Windows XP or Vista visual themes. You should always set the value of this custom property to True. This custom property is not yet supported. **Writable at Run Time**: No **Default**: True

Property Name	Description
SuppressAuthenticationWarning	If user specifies any values in the ConfiguredUser and ConfiguredPassword properties, then a warning textbox is displayed in runtime. The warning informs the user of the possible security issue of storing the database user credentials locally. If True, hides the textbox regardless of configuration. **Writable at Run Time**: No **Default**: True
WriteButtonHide	Hides the **Write** button used to send changes to the database. **Writable at Run Time**: Yes **Default**: False

AVEVA Group plc
High Cross
Madingley Road
Cambridge
CB3 0HB
UK

Tel +44 (0)1223 556655

www.aveva.com
To find your local AVEVA office, visit **www.aveva.com/offices**