



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

ICONICS

100 Foxborough Blvd.

Foxborough, MA 02035

Tel: 508-543-8600

Fax: 508-543-1503

E-Mail: support@iconics.com

Web: www.iconics.com

© ICONICS, Inc. All rights reserved.

Contents

New Features in TrendWorX32 Persistent Trending

Persistent Trending

Security

Persistent Trending OLE Automation Example

New Features in TrendWorX32 Persistent Trending

TrendWorX32 contains many new features that make the use of Persistent Trending much easier. These features include:

- Improved buffering management for load balancing.
- Faster client data-retrieval mechanism.
- Support for the Unicode version.
- New OLE automation methods to retrieve data in universal time coordinates (UTC) time format.
- Enhanced expression support
- Daylight savings time switch-over support

Note: TrendWorX32 up to and including Version 6.1x is using the Windows-based APIs for converting a time from local time to UTC or from UTC to local time. The Windows APIs do not correctly adjust the conversion for the daylight savings time if the timestamp to be converted was obtained from a different time period (i.e. it was obtained in "summer" time but now we have "winter" time). TrendWorX32 Version 7.0 has new time conversion utilities that account for this discrepancy. There are, however, some things to consider:

- These functions are used by default (no user enabling is required).
- This is what they do in order to convert each time stamp:
 - They check Windows for the PC Time Zone Settings and to see if automatic daylight savings is enabled.
 - They also obtain from Windows the times/dates for switching over. (These dates do not change; they are in the form of, e.g., the first Sunday of April.)
 - Depending on the previous information, the new functions properly adjust the conversion times to compensate for daylight savings time.

Persistent Trending

The primary purpose of Persistent Trending is to provide a real-time OLE Automation server for data collection in memory (RAM) with simultaneous data-logging capability to a user-defined CSV file. Client applications can request access from the Persistent Trending server (through OLE Automation) to its buffered data in order to enhance their functionality. The ICONICS TrendWorX32 Viewer ActiveX is already interfaced to the Persistent Trending server, and (if the server is installed on the client PC and is running) it TrendWorX32 Viewer will connect to it Persistent Trending to initialize its pens. Persistent Trending targets small to medium load applications, which require trending displays to be populated with available data upon initialization.

In general, Persistent Trending is responsible for:

- Collecting real-time data from OPC Servers in memory (RAM).
- Supplying Automation clients with the above buffered data.
- Logging data to a user-configurable .csv or .txt file.
- Managing a sophisticated **multithreaded** data-buffering system to guarantee data integrity with support for customer-configurable data-collection rates.

In terms of implementation, the Persistent Trending OLE Automation server's main features include:

- A highly customizable server.
- Fast data retrieval from memory buffers.
- Independent data collection rates for each group of points.
- A full set of exposed OLE automation methods.
- Built-in buffering system.
- Security integration.
- Support for calculated variables.
- Interface to OPC real-time servers.
- Easy Visual Basic integration.
- Embedded signal simulator.

The internal architecture of Persistent Trending allows the data to be collected at a different rate and logged to disk at a multiple of the data-collection rate in order to minimize the overall system load. Currently the minimal scan rate is set to be 10 milliseconds, and all data-collection rates that you set must be an integer multiple of the minimal scan rate. In addition, you can configure the desired number of samples to buffer (up to a maximum of 12,000) and the desired logging-to-disk interval. In addition, you can configure the file-management system for logging to disk.

Persistent Trend Automation Object Design
Configuring the Persistent Trending Automation Server
Menu Items
Groups
Support for Multiple-Tag Connection
Using the Persistent Trending OLE Automation Server

Persistent Trend Automation Object Design

The Persistent Trending application exposes, through OLE Automation, its document as a single-instance shared object. The object's name is "TWXBgTrend.Document," and it is registered in the system Registry. Once launched, the Persistent Trending server will register its document as the Active Object with the OLE Running Object Table (ROT), and therefore all client applications can access this instance rather than create a new instance each time they desire to connect to the Persistent Trending server.

In Visual Basic code, this is equivalent to:

Dim myTWXServer As Object

' get running object

Set myTWXServer = GetObject(, "TWXBgTrend.Document")

Every client application should always try to get the running instance of the server rather than create a new instance (except for the first time connection). The "TWXBgTrend.Document" OLE automation object supports the following OLE Automation properties and methods:

Properties

OLE Automation Methods

Properties

BOOL Visible	TRUE if the application window is visible; supports SET/GET
BSTR FullName	Returns the Full path and file name of the currently loaded file in TWXBgTrend; supports GET only
BOOL Maximized	TRUE if the application is maximized; supports GET/SET
BSTR Name	Returns the file name of the current file loaded in Persistent Trending; supports GET only
BSTR Path	Returns the path for the currently loaded Persistent Trending file; supports GET only
BOOL ReadOnly	Returns TRUE if the application does not allow any modifications; supports GET only

BOOL Runtime	Returns TRUE if the application is in runtime mode; supports GET only
BOOL Saved	Returns TRUE if all pending changes are saved or no new changes have been made; supports GET only
long GroupCount	Returns the number of groups configured in Persistent Trending; supports GET only

OLE Automation Methods

1. void Close(BOOL SaveChanges, BSTR File)

Call this method to shut down the application. If "SaveChanges" is TRUE, the last changes made will be saved to the user-supplied file.

2. void Open(BSTR File)

Call this method to open the user-supplied File in Persistent Trending.

3. void SaveAs(BSTR File)

Call this method to save the current Persistent Trending configuration file in "File."

4. void EnterRuntime()

Call this method to commit the Persistent Trending server into runtime mode. Data collection will start (if desired) automatically for the group, and data logging (if configured) will also be enabled.

5. void ExitRuntime()

Call this method to exit runtime mode.

6. void ActivateGroup(BSTR GroupName)

Call this method to activate a specific group (i.e. start data collection and data logging if configured). Note that each group configured in the Persistent Trending server can start data collection automatically upon entering runtime mode or when you desire by using this OLE automation method.

7. void DeActivateGroup(BSTR GroupName)

Call this method to deactivate a group. This will stop data collection and buffering in memory. If data logging is enabled for this group, all collected data up to the time of application of this method will be logged to disk.

8. long GetGroupRate(BSTR GroupName)

Call this method to obtain the data-collection rate for a specific group.

9. long GetGroupSamples(BSTR GroupName)

Call this method to obtain the number of samples the group is currently buffering in memory.

10. BOOL GetGroupActivate(BSTR GroupName)

Call this method to find out if the specific group has been configured to activate data collection and buffering upon Persistent Trending entering runtime mode.

11. long GetGroupPointCount(BSTR GroupName)

Call this method to obtain the number of signals configured for the specific group.

12. BOOL GetPointData(BSTR PointName, long Rate, long Samples, BOOL Stamp, VARIANT* Values, VARIANT* TDates, VARIANT* MSecs, VARIANT* Quality)

Call this method to retrieve buffered data in memory from Persistent Trending for a specific signal ("PointName"). "Rate" is the desired time resolution of the returned samples. "Samples" is the desired number of samples to be returned. The variants "Values," "TDates," "MSecs," and "Quality" represent variant arrays (of element types VARIANT), which contain the values, time and date stamps, millisecond resolution, and quality information about the retrieved samples.

"Values" variants utilize double values (VT_R8), and "TDates" variants utilize "DATE" types (VT_DATE). "Msecs" and "Quality" variants utilize VT_I4 types.

The Persistent Trending will allocate memory for these arrays, and the caller of this method must always FREE the associated memory. Also, if no samples are retrieved, the method will return FALSE and the variants will be empty.

Upon executing this method, the Persistent Trending server will search *all* groups configured, for the first group, which has the "PointName" signal configured. Once found, it will perform a time-stamped search to retrieve all values buffered in memory that satisfy the desired data-collection rate and number of returned samples, starting from the most current one. Persistent Trending will search for the requested data at time intervals reflecting the desired data resolution. The last valid sample within each search interval will be returned to the caller. If "bStamp" is set to TRUE, the actual time and date stamp of the retrieved sample will be modified to reflect the time and date stamp of the search interval.

13. BOOL GetGroups(VARIANT* GroupNames)

Call this method to obtain all the group names configured in the Persistent Trending server. They are returned in the array of BSTR's "GroupNames," as before Persistent Trending will allocate memory for these arrays, the caller of this method must always FREE the associated memory. If no groups are configured, the method returns FALSE.

14. BOOL GetGroupPoints(BSTR GroupName, VARIANT* GroupPoints)

Call this method to retrieve the signal names configured for a given group. They are returned in the array of BSTR's "GroupPoints," and before Persistent Trending will allocate memory for these arrays, the caller of this method must always FREE the associated memory. If no signals are configured, the method returns FALSE.

15. BOOL GetGroupLogging(BSTR GroupName)

Call this method to find whether data logging has been enabled for a specific group. The method returns TRUE if data logging is enabled.

16. BSTR GetGroupFile(BSTR GroupName)

Call this method to retrieve the current "FileName" to which a specific group is data logging.

17. BOOL GetGroupPointData(BSTR GroupName, BSTR PointName, long Rate, long Samples, BOOL Stamp, VARIANT* Values, VARIANT* TDates, VARIANT* MSecs, VARIANT* Quality)

Call this method to retrieve buffered data in memory from Persistent Trending for a specific signal ("PointName"). "Rate" is the desired time resolution of the returned samples. "Samples" is the desired number of samples to be returned. The variants "Values," "TDates," "MSecs," and "Quality" represent variant arrays (of element types "VARIANT"), which contain the values, time and date stamps, millisecond resolution, and quality information about the retrieved samples. "Values" variants utilize double values (VT_R8), "Tdates" variants utilize "DATE" types (VT_DATE), and "Msecs" and "Quality" utilize VT_I4 types.

Persistent Trending will allocate memory for these arrays, and the caller of this method must always FREE the associated memory. Also, if no samples are retrieved, the method will return FALSE and the variants will be empty.

Upon executing this method, the Persistent Trending server will search all Persistent Trend groups for a group called "GroupName," and then it will search for the "PointName" signal in the requested group. Then it will perform a time-stamped search to retrieve all values buffered in memory that satisfy the desired data-collection rate and the number of returned samples, starting from the most current one. Persistent Trending will search for the requested data at time intervals reflecting the desired data resolution. The last valid sample within each search interval will be returned to the caller. If "bStamp" is set to TRUE, the actual time and date stamp of the retrieved sample will be modified to reflect the time and date stamp of the search interval.

18. BOOL GetPointStat(BSTR GroupName, BSTR PointName, VARIANT* StatValues)

Call this method to retrieve statistical information about a specific signal belonging to a specific group. The returned array of variants "StatValues" will have the following as "VARIANT" elements:

- Min value
- Max value
- Average value
- Standard deviation
- Starting time for statistics
- Ending time for statistics

Persistent Trending will allocate memory for these arrays, and the caller of this method must always FREE the associated memory. Also, if no samples are retrieved, the method will return FALSE and the variants will be empty.

19. BOOL GetGroupPointAt(BSTR GroupName, BSTR PointName, DATE TDateAt, VARIANT* Value, VARIANT* TDateFound, VARIANT* Qual)

Call this method to retrieve a sample for a given time and date from Persistent Trending. All the returned information is in variants, and if the method succeeds, it returns TRUE.

20. BOOL GetGroupPointDataAt(BSTR GroupName, BSTR PointName, long Rate, long Samples, BOOL Stamp, DATE TDateFrom, VARIANT* Values, VARIANT* TDates, VARIANT* MSecs, VARIANT* Quality)

This method is the same as "GetGroupPointData," with the exception that now you can specify a starting time and date stamp in "TDateFrom" to retrieve data from Persistent Trending's memory buffers.

21. BOOL GetPointDataAt(BSTR PointName, long Rate, long Samples, boolean Stamp, DATE TDateFrom, VARIANT* Values, VARIANT* TDates, VARIANT* MSecs, VARIANT* Quality)

This method is the same as "GetPointData," with the exception that now you can specify a starting time and date stamp in "TDateFrom" to retrieve data from Persistent Trending's memory buffers.

22. void Refresh()

Call this method to refresh the **Persistent Trending** window. This method should be called only once.

23. boolean GetPointDataUTC(BSTR GroupName, BSTR PointName, long Rate, long Samples, BOOL Stamp, VARIANT* Values, VARIANT* TDates, VARIANT* MSecs, VARIANT* Quality)

Call this method to retrieve buffered data in memory from Persistent Trending for a specific signal ("PointName"). "Rate" is the desired time resolution of the returned samples. "Samples" is the desired number of samples to be returned. The variants "Values," "TDates," "MSecs," and "Quality" represent variant arrays (of element types "VARIANT"), which contain the values, time and date stamps, millisecond resolution, and quality information about the retrieved samples. "Values" variants utilize double values (VT_R8), "Tdates"

variants utilize "DATE" types (VT_DATE), and "Msecs" and "Quality" utilize VT_I4 types.

Persistent Trending will allocate memory for these arrays, and the caller of this method must always FREE the associated memory. Also, if no samples are retrieved, the method will return FALSE and the variants will be empty.

Upon executing this method, the Persistent Trending server will search all Persistent Trend groups for a group called "GroupName," and then it will search for the "PointName" signal in the requested group. Then it will perform a time-stamped search to retrieve all values buffered in memory that satisfy the desired data-collection rate and the number of returned samples, starting from the most current one. Persistent Trending will search for the requested data at time intervals reflecting the desired data resolution. The last valid sample within each search interval will be returned to the caller. If "bStamp" is set to TRUE, the actual time and date stamp of the retrieved sample will be modified to reflect the time and date stamp of the search interval. Time stamps are returned in UTC time format.

24. boolean GetGroupPointDataUTC(BSTR GroupName, BSTR PointName, long Rate, long Samples, BOOL Stamp, VARIANT* Values, VARIANT* TDates, VARIANT* MSecs, VARIANT* Quality);

Call this method to retrieve buffered data in memory from Persistent Trending for a specific signal ("PointName"). "Rate" is the desired time resolution of the returned samples. "Samples" is the desired number of samples to be returned. The variants "Values," "TDates," "MSecs," and "Quality" represent variant arrays (of element types "VARIANT"), which contain the values, time and date stamps, millisecond resolution, and quality information about the retrieved samples. "Values" variants utilize double values (VT_R8), "TDates" variants utilize "DATE" types (VT_DATE), and "Msecs" and "Quality" utilize VT_I4 types.

Persistent Trending will allocate memory for these arrays, and the caller of this method must always FREE the associated memory. Also, if no samples are retrieved, the method will return FALSE and the variants will be empty.

Upon executing this method, the Persistent Trending server will search all Persistent Trend groups for a group called "GroupName," and then it will search for the "PointName" signal in the requested group. Then it will perform a time-stamped search to retrieve all values buffered in memory that satisfy the desired data-collection rate and the number of returned samples, starting from the most current one. Persistent Trending will search for the requested data at time intervals reflecting the desired data resolution. The last valid sample within each search interval will be returned to the caller. If "bStamp" is set to TRUE, the actual time and date stamp of the retrieved sample will be modified to reflect the time and date stamp of the search interval. Time stamps are returned in UTC time format.

25. boolean GetPointStatUTC(BSTR GroupName, BSTR PointName, VARIANT* StatValues);

Call this method to retrieve statistical information about a specific signal belonging to a specific group. The returned array of variants "StatValues" will have the following as "VARIANT" elements:

- Min value

- Max value
- Average value
- Standard deviation
- Starting time for statistics
- Ending time for statistics

Persistent Trending will allocate memory for these arrays, and the caller of this method must always FREE the associated memory. Also, if no samples are retrieved, the method will return FALSE and the variants will be empty. Time stamps are returned in UTC time format.

26. boolean GetGroupPointAtUTC(BSTR GroupName, BSTR PointName, DATE TDateAt, VARIANT* Value, VARIANT* TDateFound, VARIANT* Qual)

Call this method to retrieve a sample for a given time and date from Persistent Trending. All the returned information is in variants, and if the method succeeds, it returns TRUE. Time stamps are returned in UTC time format.

27. boolean GetGroupPointDataAtUTC(BSTR GroupName, BSTR PointName, long Rate, long Samples, boolean Stamp, DATE TDateFrom, VARIANT* Values, VARIANT* TDates, VARIANT* MSecs, VARIANT* Quality)

This method is the same as "GetGroupPointData," with the exception that now you can specify a starting time and date stamp in "TDateFrom" to retrieve data from Persistent Trending's memory buffers. Time stamps are returned in UTC time format.

28. boolean GetPointDataAtUTC(BSTR PointName, long Rate, long Samples, boolean Stamp, DATE TDateFrom, VARIANT* Values, VARIANT* TDates, VARIANT* MSecs, VARIANT* Quality)

Call this method to retrieve buffered data in memory from Persistent Trending for a specific signal ("PointName"). "Rate" is the desired time resolution of the returned samples. "Samples" is the desired number of samples to be returned. The variants "Values," "TDates," "MSecs," and "Quality" represent variant arrays (of element types VARIANT), which contain the values, time and date stamps, millisecond resolution, and quality information about the retrieved samples.

"Values" variants utilize double values (VT_R8), and "TDates" variants utilize "DATE" types (VT_DATE). "Msecs" and "Quality" variants utilize VT_I4 types.

The Persistent Trending will allocate memory for these arrays, and the caller of this method must always FREE the associated memory. Also, if no samples are retrieved, the method will return FALSE and the variants will be empty.

Upon executing this method, the Persistent Trending server will search *all* groups configured, for the first group, which has the "PointName" signal configured. Once found, it will perform a time-stamped search to retrieve all values buffered in memory that satisfy the desired data-collection rate and

number of returned samples, starting from the most current one. Persistent Trending will search for the requested data at time intervals reflecting the desired data resolution. The last valid sample within each search interval will be returned to the caller. If "bStamp" is set to TRUE, the actual time and date stamp of the retrieved sample will be modified to reflect the time and date stamp of the search interval. Time stamps are returned in UTC time format.

29. boolean GetGroupPointInfo(BSTR GroupName, BSTR PointName, VARIANT* HighRange, VARIANT* LowRange, VARIANT* EngUnits, VARIANT* Description);

This method returns point information for a given signal, such as "HighRange" and "LowRange," "EngUnits," and "Description," given a Persistent Trend "GroupName."

30. boolean GetPointInfo(BSTR PointName, VARIANT* HighRange, VARIANT* LowRange, VARIANT* Description, VARIANT* EngUnits);

This method returns point information for the first signal, such as "HighRange" and "LowRange," "EngUnits," and "Description," that matches "PointName."

ProjectWorX32 Support

Several new OLE Automation methods have been added to facilitate ProjectWorX32 deployment and WebHMI enhanced support:

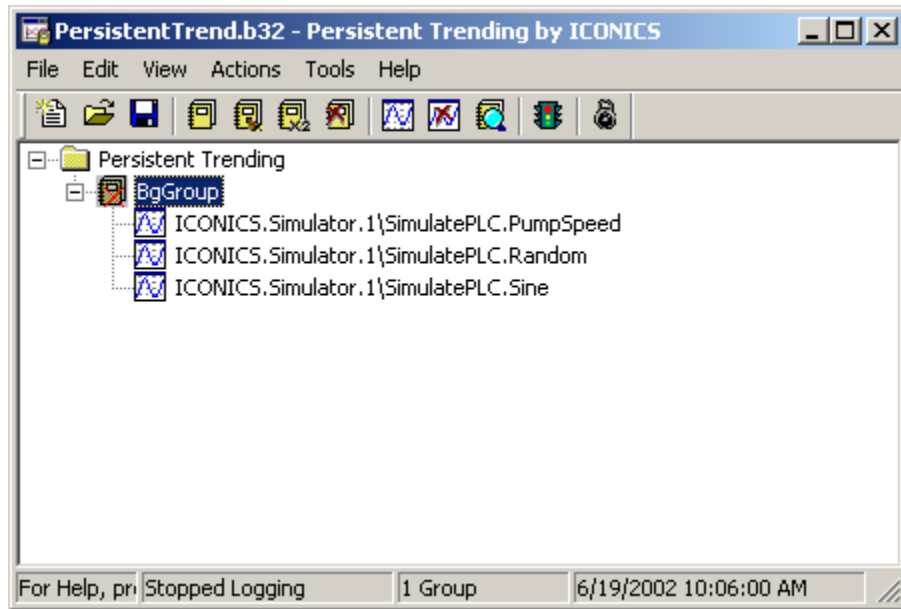
1. **ReplaceStringInString**(
 StringToReplace as String,
 OldSubstring as String,
 NewSubstring as String
) as String
2. **ReplaceHost**(
 OldHostName as String,
 NewHostName as String
) as Long
3. **ReplaceFilePath**(
 OldSubstring as String,
 NewSubstring as String
) as Long
4. **ReplaceTag**(
 OldSubstring as String,
 NewSubstring as String
) as Long
5. **ReplaceHostEx**(

```
OldHostNameSubstring as String,  
NewHostNameSubstring as String  
MatchCase as Boolean,  
MatchWholeWord as Boolean  
) as Long
```

The **ReplaceHostEx** with parameters **MatchCase=False & MatchWholeWord=True** works exactly the same as the **ReplaceHost()** function.

Configuring the Persistent Trending OLE Automation Server

The Persistent Trending OLE Automation server can be easily configured to request data from various OPC servers, and can evaluate expressions as well as internal simulation points.



TrendWorX32 Persistent Trending

Menus

The **Menu Bar** in Persistent Trending contains the following menus:

- **File menu**
- **Edit menu**
- **View menu**
- **Actions menu**
- **Tools menu**
- **Help menu**

File Menu

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

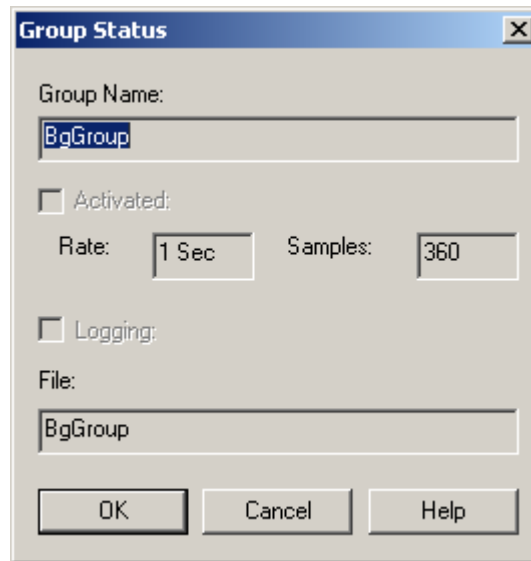
Command	Shortcut Key	Function
New	CTRL+N	Opens a new Persistent Trending window.
Open	CTRL+O	Opens an existing Persistent Trending window.
Save	CTRL+S	Saves the open Persistent Trending window.
Save As		Allows you to select the name under which the open Persistent Trending window should be saved.
Exit		Closes Persistent Trending.

Edit Menu

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

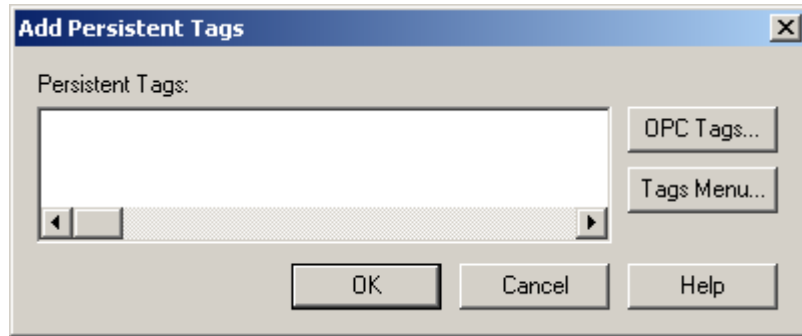
Command	Shortcut Key	Function
Duplicate	CTRL+D	Duplicates a Persistent Trending group. Persistent Trending will duplicate the group by inserting a new group and appending _x to the group's name, in addition to duplicating all of the signals that are already configured.

Delete	DEL	Deletes a Persistent Trending group.
Group Status		Displays the Group Status dialog box information for a selected group. Right-clicking a group and selecting Group Status from the menu also opens this dialog box.



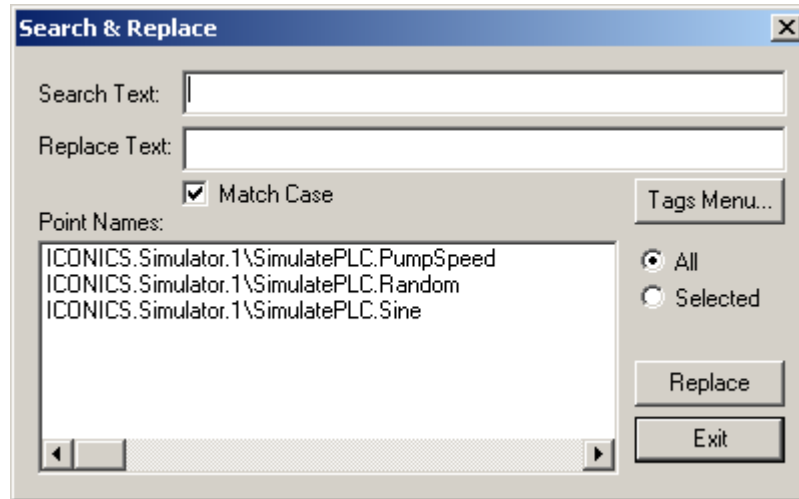
Group Status Dialog Box

Add Group	CTRL+G	Adds a group to the opened Persistent Trend.
Edit Group	CTRL+E	Opens the Persistent Trending dialog box for the group, which enables you to change any of the original configuration settings.
Add Tags	CTRL+P	Enables you to create signal connections to already configured Persistent Trending groups. From the Add Persistent Tags dialog box, you can add OPC tags by clicking OPC Tags , or add expressions and simulated variables by clicking Tags Menu . Once you add all the tags you wish to configure, click OK to add these tags to the selected Persistent Trending group.



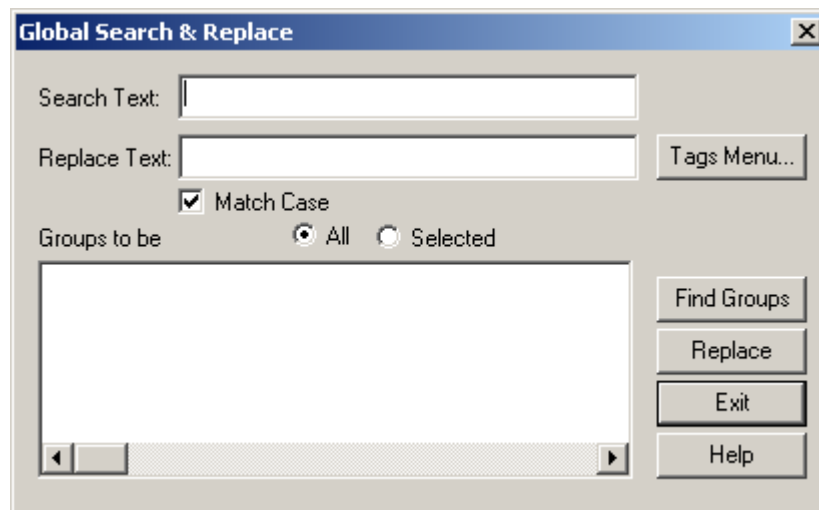
Add Persistent Tags Dialog Box

Delete Tags	CTRL+SHIFT+P	Enables you to delete signals from already configured Persistent Trending Groups. From the Delete Persistent Tags dialog box, you can choose which signals to delete. Clicking OK to remove these tags from the selected Persistent Trending group.
Copy Tags		Opens the Copy Tags dialog box, which lists all of the tags available in the selected group. After you select tags, they are copied to be pasted into another group.
Paste Tags		Pastes all tags existing on the clipboard in the selected group.
Group Search	CTRL+H	Opens the Search and Replace dialog box. You can select all the tags to which you want to apply the search and replace. Define a search string (which can be the fully configured tag or a portion of it), define a replacement string, and then click Replace .



Search and Replace Dialog Box

Global Search	CTRL+F	Activates the Global Search and Replace dialog box. You can search for a given string across all or selected configured groups by clicking Find Groups . Select a replacement string and the groups to which you want to apply the new string, and then click the Replace to continue.
---------------	--------	---



Global Search and Replace Dialog Box

View Menu

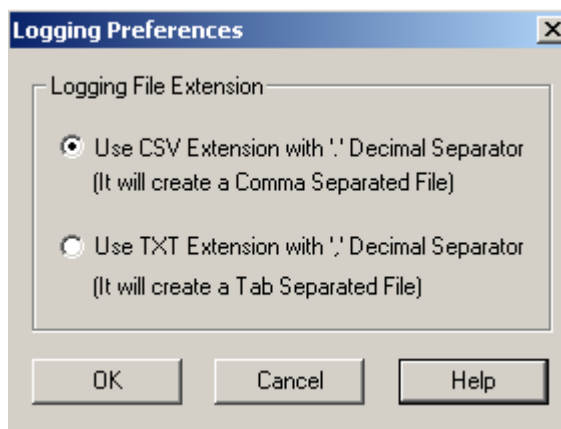
The **View** menu contains the following commands:

Command	Shortcut Key	Function
Toolbar	CTRL+T	When checked, shows the Persistent Trending toolbar as shown below.



Toolbar

Status Bar	CTRL+B	When checked, shows the status bar, which indicates the status of the application, the number of groups, and the time and date.
Select Language	CTRL+SHIFT+U	Opens the Select Language dialog box, which enables you to select a language that TrendWorX32 will automatically load if the corresponding language resource .dll is present in the TrendWorX32 Bin directory. Persistent Trending will automatically switch its menus and dialogs to the selected language, and it will also start up in that language.
Logging Preferences	SHIFT+L	Enables you to log to a .csv file or a .txt file. When using the .csv file, decimal places will be indicated by the '.'. When using the .txt file, decimal places will be indicated by the ','. This is useful for those using Persistent Trending on systems set up for European standards (using ',' as a decimal separator).



Logging Preferences Dialog Box

Actions Menu

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

Command	Shortcut Key	Function
Start Logging	CTRL+R	Starts logging. This command is only available when the Persistent Trend is not logging.
Stop Logging	CTRL+L	Stops logging. This command is only available when the Persistent Trend is currently logging.
Activate Group	CTRL+A	Depending on the current state of a particular group, activates it or deactivates the selected group. This will start or stop all data collection and logging for the selected group. A check mark indicates that the group is activated.

Tools Menu

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

Command	Shortcut Key	Function
Security Login	F6	Prompts for login to the ICONICS Security Server.
Security Configuration	F7	Configures the setup for the ICONICS Security Server.
Working Directory	F8	Sets the working directory where all files associated with Persistent Trending are kept.

Help Menu

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

Command	Shortcut Key	Function
Help Topics	F1	Launches the online Help tutorial for Persistent Trending.
About Application		Displays both technical support information and the ICONICS About Box, which contains registration and serial number information as well as system resource information, such as memory and disk space available.

Groups

The key concept in the configuration of the Persistent Trending server is the **Group**. A group represents a logical collection of signals for common processing and buffering.

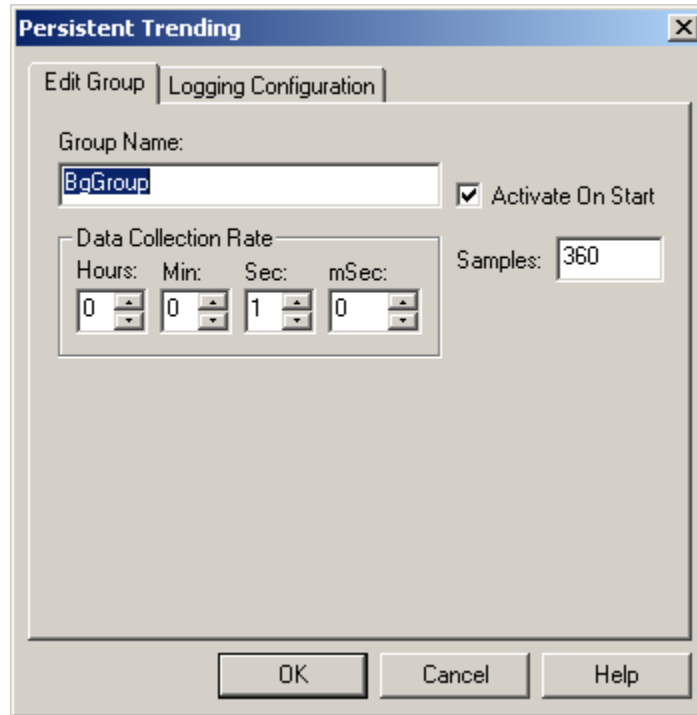
Adding and Configuring a Group Context Menu

Adding and Configuring a Group

To add a group, choose **Add Group** from the **Edit** menu, or press the shortcut keys **CTRL+G**. This opens the **Persistent Trending** dialog box, which enables you to configure a new Persistent Trending group and (optionally) data logging for that group to a .csv or .txt file. The **Persistent Trending** dialog box contains two tabs: the **Group Configuration** tab and the **Logging Configuration** tab.

Group Configuration Tab

The **Group Configuration** tab of the **Persistent Trending** dialog box, shown in the figure below, allows you to configure a name for the new Persistent Trending, the desired data-collection rate, and the number of samples to buffer in memory for each signal of the group. The **Activate On Start** option, if enabled, automatically starts data collection for the specific group, when the Persistent Trending is committed to runtime mode. Otherwise, you can start data collection by activating the group during runtime mode through the Persistent Trending User Interface or through OLE Automation.



Group Configuration Tab

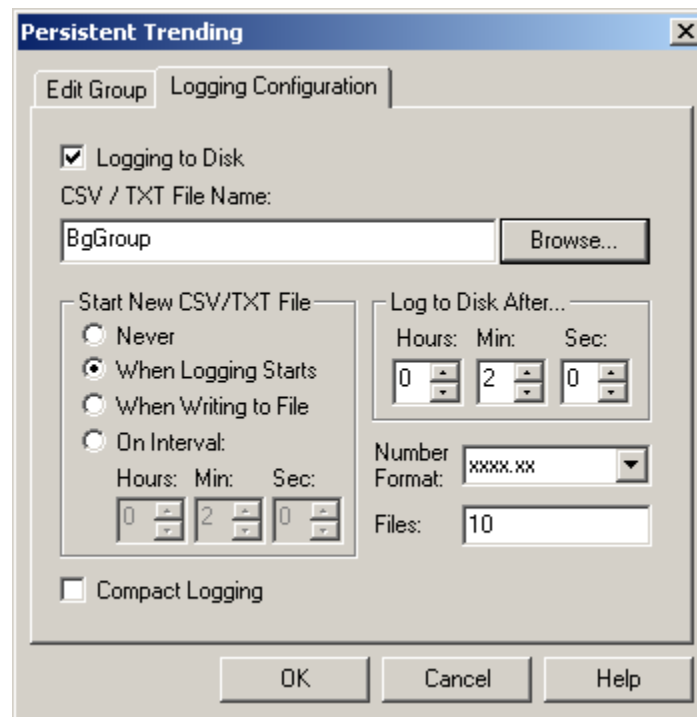
Logging Configuration Tab

The **Logging Configuration** tab of the **Persistent Trending** dialog box, shown in the figure below, allows you to configure the Persistent Trending server to log the buffered data to a .csv or .txt file. At this point, you can:

1. Enable data logging.
2. Choose a .csv or .txt file to log using the **Browse** command.
3. Choose a .csv or .txt file system maintenance scheme. You may choose to append all data to the same file (if **Never** is selected), create a new file every time logging starts (if **When Logging Starts** is selected), create a new file every time Persistent Trending writes to disk (if **When Writing to File** is selected), or create a new file after a certain interval of time has passed (if **On Interval** is selected and an interval is set).
4. Write more samples to each row using the **Compact Logging** feature.
5. Configure the frequency by which data will be written to disk.
6. Select the desired number format for the .csv or .txt file data.
7. Select the desired date format.
8. Select the maximum number of .csv or .txt files to maintain.

If you want to log to multiple files, the Persistent Trending server will create

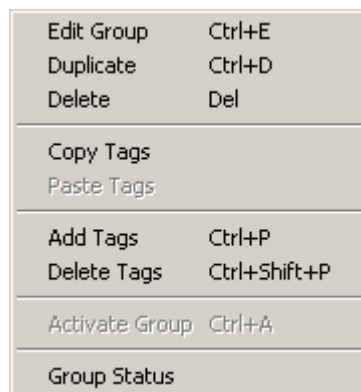
the files by appending to the file name the corresponding extension for the current file number. Once it reaches the maximum number of files, it will start overwriting from the first created file.



Logging Configuration Tab

Context Menu

Right-clicking on a group in the Persistent Trend interface will open a **Context** menu, which contains some of the more-common menu commands, including **Edit Group**, **Duplicate**, **Delete**, and **Add Tags**.

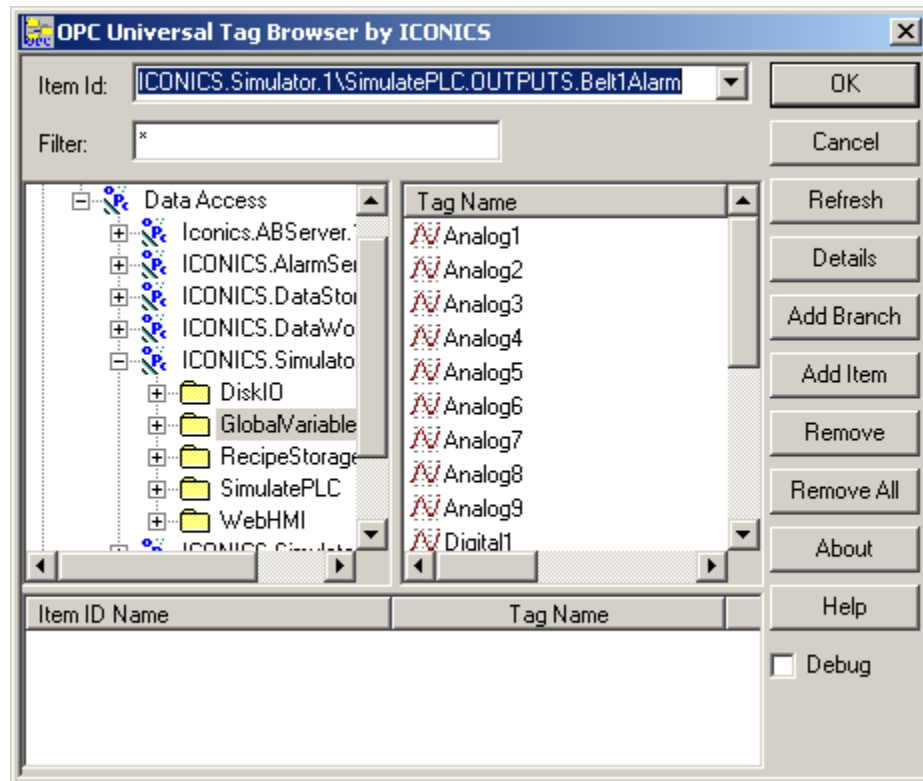


Context Menu

For further explanation of each of these commands, please see the **Menus** section.

Support for Multiple-Tag Connection

The **OPC Universal Tag Browser**, shown below, helps add the Multiple-Tag Connection feature to Persistent Trending. To open the **OPC Universal Tag Browser**, select **OPC Tag** from the **Add Persistent Tags** dialog box. Use the CTRL or SHIFT keys to select items from the **Selected Tags** list. Click **Add Item** to add all of the selected tags to a certain group.



OPC Universal Tag Browser

Using the Persistent Trending OLE Automation Server

The purpose of the Persistent Trending OLE Automation server is to function primarily as a memory-based data-logging engine, which buffers data for user-configured signals at desired data-collection rates in memory. In addition, the buffered data can be configured to be data-logged in a .csv or .txt file.

The Persistent Trending automation server is a stand-alone (EXE) server, which registers as the active running object with the tables maintained by OLE. This means that there is only one instance of Persistent Trending running per computer, and all interested clients should get an interface pointer to that running instance (rather than create a new Persistent Trending object, which will fail if Persistent Trending is already running).

The Persistent Trending server should be started and running at the time of committing a process control/HMI application to runtime mode, so it will have enough time to buffer data before any client requests are processed. There are several ways this can be done.

You can start the Persistent Trending server manually, from the Program Group, or place it in the StartUp Folder. In either case, the Persistent Trending server will load the most-recent configured file. Then it can be committed into runtime mode by choosing **Start Logging** from the **Actions** menu, or clicking the **Traffic Light** button on the toolbar.

You can evoke the Persistent Trending server programmatically. The following is a small Visual Basic code sample used to initiate the Persistent Trending server, to load a desired configuration file, and to enter runtime mode:

```
Dim myTrend As Object
```

```
' Create Persistent Trending Server Object
```

```
Set myTrend = CreateObject("TWXBgTrend. Document")
```

```
' load configuration file
```

```
myTrend.Open "D:\MyDirFE32\sim10.b32"
```

```
' Show Persistent Trending Window
```

```
myTrend.Visible = True
```

```
' Refresh Persistent Trending Window
```

myTrend.Refresh

' Enter Runtime

myTrend.EnterRuntime

Once this code is executed, Persistent Trending will initiate, load the desired configuration file, show its window, and enter runtime mode. At this point, any automation client of the Persistent Trending server should be able to get an instance to the running TWXBgTrend object, and it will be able to interact with the server completely.

Here is another Visual Basic code sample used to actually retrieve data from Persistent Trending's buffers:

Dim myTrendObject As Object

' Get Running Object

Set myTrendObject = GetObject(, "TWXBgTrend.Document")

' string sPoint is the desired Point Name for which data will be retrieved

' Set up variables to receive data from Background Trending Server

' A variant array for sample values

Dim Values As Variant

' A variant array for Time and Date stamps

Dim TDates As Variant

' A variant array for Msecs timestamps

Dim MSecs As Variant

' A variant array for OPC quality flags

Dim Qual As Variant

' Index used for iteration

Dim I As Long

' RateVal is a long representing the desired data resolution rate

**' SamplesVal is a long representing the desired number of
' samples to retrieve**

Dim sPointVal As String

Dim vEntry As Variant

Dim bRet As Boolean

Dim strValue As String

' Call Background Trending automation method

**bRet = myTrendObject.GetPointData(sPoint, RateVal, SamplesVal,
FALSE, Values, TDates, MSecs, Qual)**

' Fill In Values in a listbox control called PointValues

PointValues.Clear

'Fill In !!!

For I = LBound(Values) To UBound(Values)

vEntry = Values(I)

strValue = TypeName(vEntry)

sPointVal = Str(vEntry)

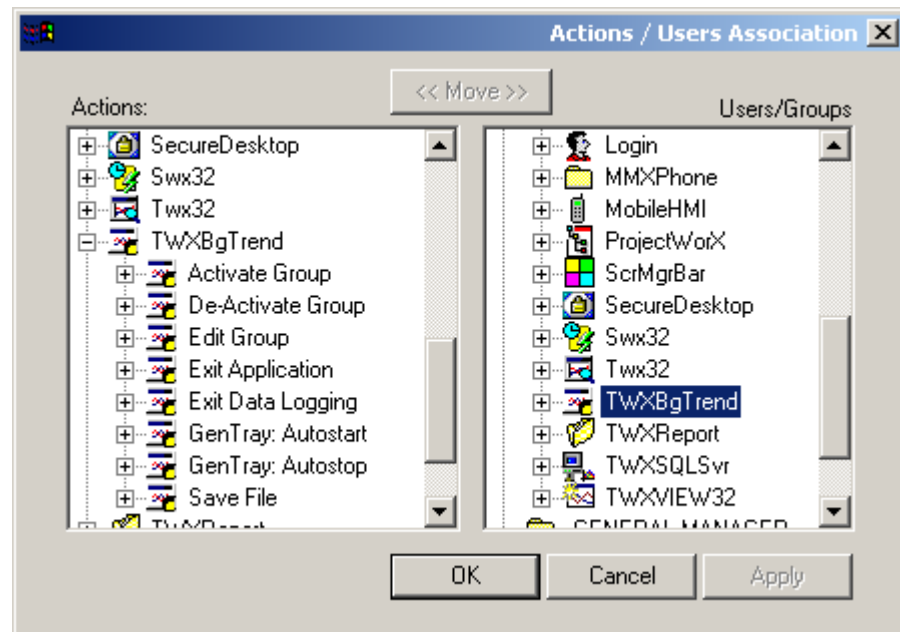
PointValues.AddItem sPointVal

Next I

Note that error-checking code is omitted for clarity. The caller should always check for empty arrays, in case there were no data to return.

Security

The Persistent Trending Server is interfaced to the ICONICS Security Server, which you can access by choosing **Security** from the **Tools** menu. It currently supports the security actions shown in the dialog box below.



Actions/Users Association Dialog Box

Persistent Trending is a very flexible OLE Automation server to configure and use. Its main advantage is that it allows you to define and configure custom client-side groups, with individual data-collection rates and number of samples to configure. Persistent Trending is interfaced to ICONICS OLEExpress thin-layer OPC wrapper .dll, which transfers requests for data to the underlying OPC server.

You can take some steps to optimize Persistent Trending performance and system resource usage. Defining tags (signals) from the same OPC server in Persistent Trending groups can improve system resource usage by Persistent Trending. Also, avoid requesting the same tags in different Persistent Trending groups to speed up performance. Although Persistent Trending does not require you to adhere to these optimizations, using them has many benefits, including enhanced performance and load balancing.

Persistent Trending also has support for calculating expressions. The time resolution of the expressions is not as granular as that of ordinary OPC server tags. *Under a heavy data-logging load, expressions may not get enough time to evaluate.* In any case, Persistent Trending cannot evaluate expressions *faster than 250 milliseconds*, assuming no data-logging load and a reasonable number of points overall (about 50 OPC tags).

Persistent Trending OLE Automation Examples

The ICONICS GENESIS32 installation directory on your hard drive contains a folder called "Examples," which contains a folder called "TrendWorX32 Examples." This folder contains a folder called "TrendWorX32 VBA Examples." This folder contains a sample TrendWorX32 file called "PersistentTrendVBADemo.t32," which demonstrates how to use Persistent Trending's OLE Automation properties. There is also a "readme.txt" file that explains how to use this sample Visual Basic application.