



ICONICS, Inc.
100 Foxborough Blvd.
Foxborough, MA 02035
Phone: (508) 543-8600
Fax: (508) 543-1503
[Worldwide Support Centers
www.iconics.com](http://www.iconics.com)



© ICONICS, Inc. All rights reserved.
Specifications are subject to change without notice.
GENESIS, GENESIS32, Pocket GENESIS, BizViz and their
respective modules, OPC-To-The-Core, and Visualize Your
Enterprise are trademarks of ICONICS, Inc. Other product
and company names mentioned herein may be trademarks of
their respective owners.

Features in TrendWorX32 Reporting

TrendWorX32 Reporting includes the following key features:

- Microsoft Excel reporting password support
- Microsoft Excel reporting performance tuning
- Enhanced data retrieval support
- Microsoft ADO 2.7 support
- Updated Microsoft Excel date handling

- Microsoft Excel FDA password support. TrendWorX32 Reporting supports creating Microsoft Excel reports where a random password will be created to lock the worksheet if desired. The password is not stored anywhere, therefore providing added security for FDA reports. This support is on a per-report basis.
- Report tag reordering
- Report time span enabled for periodic reports
- TraceWorX32 debug tracing support
- Enhanced expression support
- Daylight savings time 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.x introduced 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:
 - o They check Windows for the PC Time Zone Settings and to see if automatic daylight savings is enabled.
 - o 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.)
 - o Depending on the previous information, the functions properly adjust the conversion times to compensate for daylight savings time.

TrendWorX32 also includes the following features:

- Data-retrieval filters
- Scheduling and load-balancing support
- Microsoft Excel integration and template support
- HTML and e-mail support for Microsoft Excel-based reporting
- Unicode version support
- Report integration to historical database, operator comments, and batch information
- Multiple database support

- Easy-to-use report configuration wizard
- Microsoft SQL 2000 support
- Tag column description support



Introduction

TrendWorX32 has the capability to log data to a relational database. In addition to the standard tools available for accessing these data, TrendWorX32 has built-in capability for historical data utilities and reporting. The tools available to perform historical data utilities and reporting are:

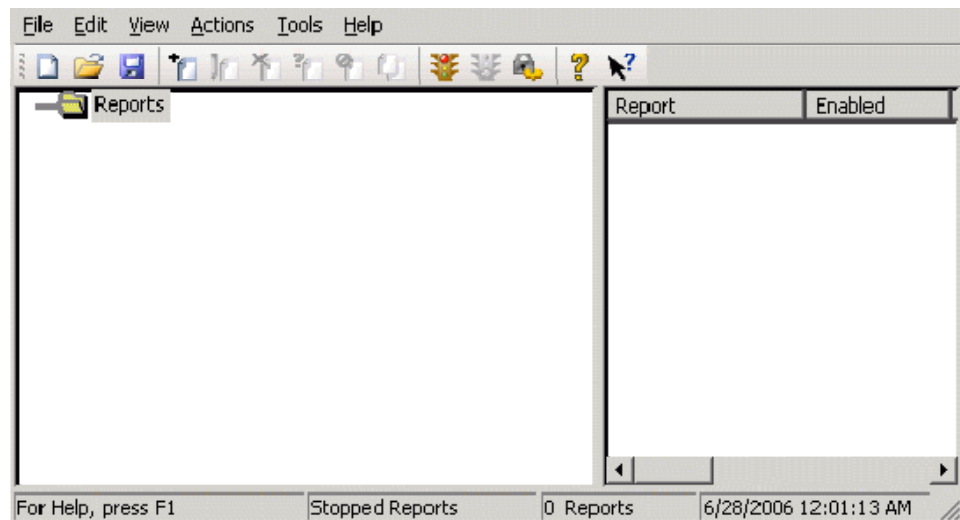
- [TrendWorX32 SQL Tool ActiveX Control](#)
- [TrendWorX32 SQL Reporting](#)



Starting TrendWorX32 Reporting

To launch TrendWorX32 Reporting from the Windows Start menu, select **Programs > ICONICS GENESIS32 > TrendWorX32 > TrendWorX32 Reporting**.

TrendWorX32 Reporting will be launched with the last configured file (.rtw). If this is the first time using TrendWorX32 Reporting, a blank configuration is displayed, as shown below.



TrendWorX32 Reporting Screen

TrendWorX32 Reporting displays a two-pane screen. The left-hand pane contains a "tree" of all the configured reports and the tags for each report.

The right-hand pane is a list view providing report configuration and status information. TrendWorX32 Reporting can execute reports containing multiple database groups and/or logging groups. The tree in the left pane sorts all signals by database type.

Note: Clicking a column heading in the right pane of the display, such as "Report," "Enabled," or "Report Type," sorts the list of reports by that category.

Reporting Modes

TrendWorX32 Reporting supports two modes of operation:

1. A **configuration** mode in which you can configure reports without any report processing.
2. A **runtime** mode in which all reports are processed and full configuration is available.

Auto-starting TrendWorX32 as a Service

When you run the AppSetupUtility as part of your installation (or at a later time) it creates an entry in the GENESIS INI file that registers TrendWorX32 Reporting as a service. The service is left in Manual state and in the Stopped condition. You can use the Services control panel to Start the service, and if desired set it to Automatic so that it starts up every time you reboot the system. The service is called IcoTrendReportHost.exe and appears as the ICONICS TrendWorX Reporting service in the control panel.

To add TraceWorX support add the following entry into the ICOSSETUP.INI file:

```
[TRACEWORX]
```

```
IcoTrendReportHost.exe=Software\ICONICS\IcoTrendReportHost
```

```
[ICOTRENDREPORTHOST]
```

```
Release TraceModuleDesc=TrendWorX32 Report Host service
```

Then add the IcoTrendReportHost.exe and register it as a service.

To add the GenEvent server:

Add support for NT Events using the TWXReportHost as GenEvent application name

If TWXReport is set to run as an NT service, GenTray should start the IcoTrendReportHost.exe service, which in turn will start TWXReport.exe.

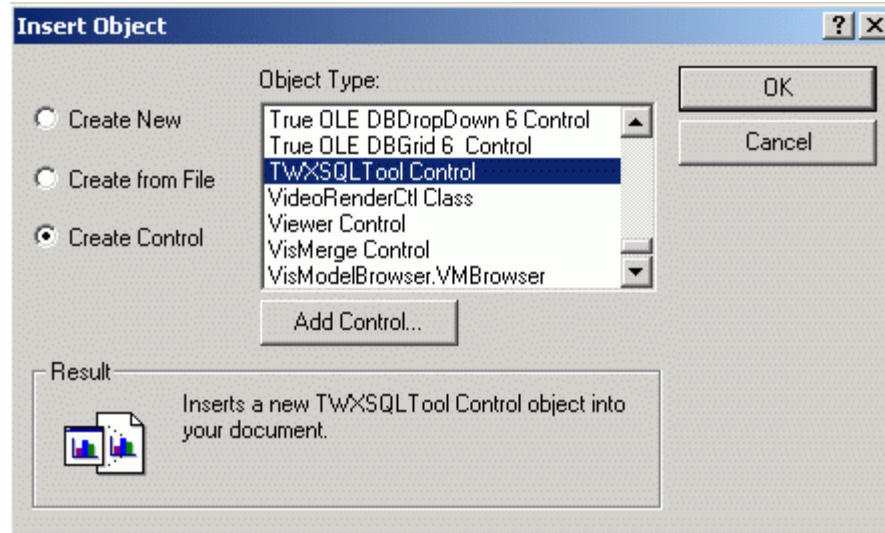
 [TrendWorX32 Reporting](#)

TrendWorX32 SQL Tool ActiveX Control

Version 7.x included an enhanced data-retrieval system, which utilizes an updated approach to creating historical reports:

- Data filter selection other than Raw will result in data time stamped at the beginning of each subinterval.
- When retrieving data using data filters other than Raw, subintervals for which there are no data because of no data-logging activity will be marked as "empty" slots at the corresponding time with an empty (VT_EMPTY) value. You can check the returned qualities for further processing.
- The historical qualities supported are as follows:
 - O OPCHDA_EXTRADATA 0x0001
 - More data may exist
 - O OPCHDA_INTERPOLATED 0x0002
 - Interpolated data value
 - O OPCHDA_RAW 0x0004
 - Raw data value
 - O OPCHDA_CALCULATED 0x0008
 - Calculated data value
 - O OPCHDA_NOBOUND 0x0010
 - No data found to provide upper or lower bound
 - O OPCHDA_NODATA 0x0020
 - No data collected. Archiving not active
 - O OPCHDA_DATALOST 0x0040
 - Calculation started/stopped/lost
 - O OPCHDA_CONVERSION 0x0080
 - Scaling / conversion error
 - O OPCHDA_PARTIAL 0x0100
 - Aggregate value is for an incomplete interval

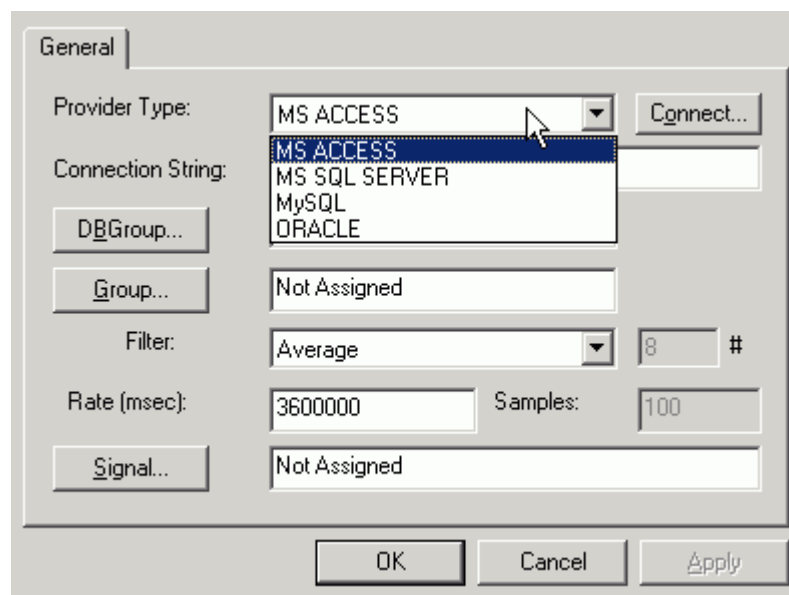
The TrendWorX32 SQL Tool, which is invisible during runtime mode, is an ActiveX control that you can use to browse the TrendWorX32 data-logging database and to retrieve time-based historical data using various data filter types. You can access this tool through the **Insert Object** dialog box, as shown below, or directly from the **ActiveX** toolbar of the TrendWorX32 Container.



Insert Object Dialog Box

During configuration, you can place this component inside TrendWorX32 displays, VBA forms, or any other container.

Use the **TWXSQL Tool Control Properties** dialog box, shown below, to configure the component. This dialog box has only one tab, the **General** tab shown below, which prompts you to complete the configuration.



TWXSQL Tool Control Properties Dialog Box

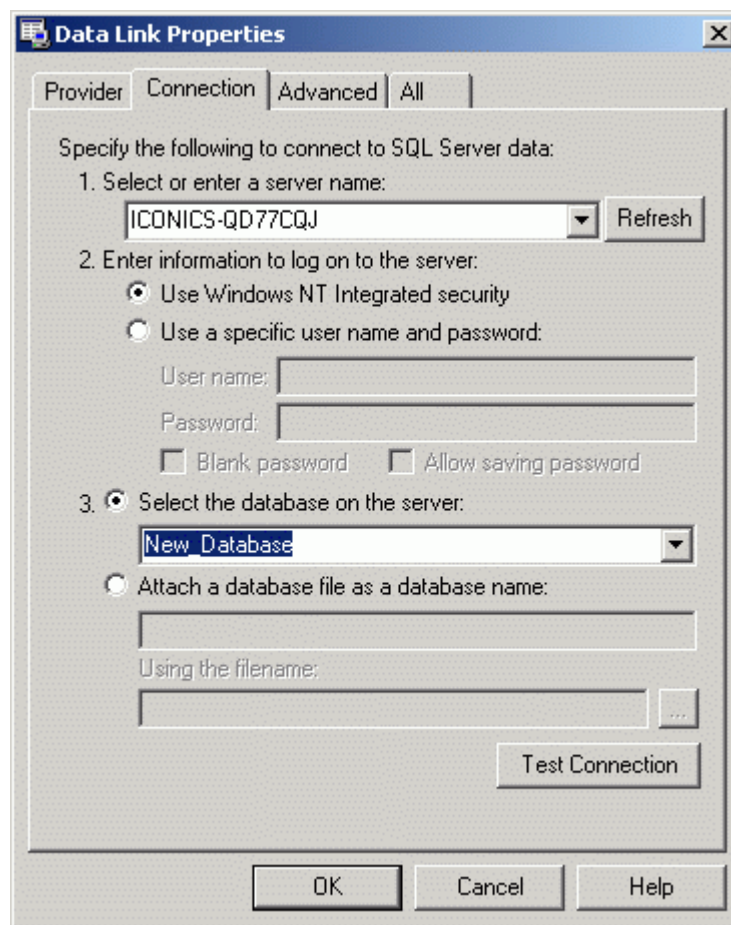
To configure the SQL Tool control, do the following:

1. From the **Provider Type** list box in the **TWXSQL Tool Control Properties** dialog box, select the OLE Database (DB) Provider type relating to the TrendWorX32 historical database. Currently, the following OLE DB Providers are supported:

- Microsoft Access
- Microsoft SQL Server
- Oracle
- MySQL

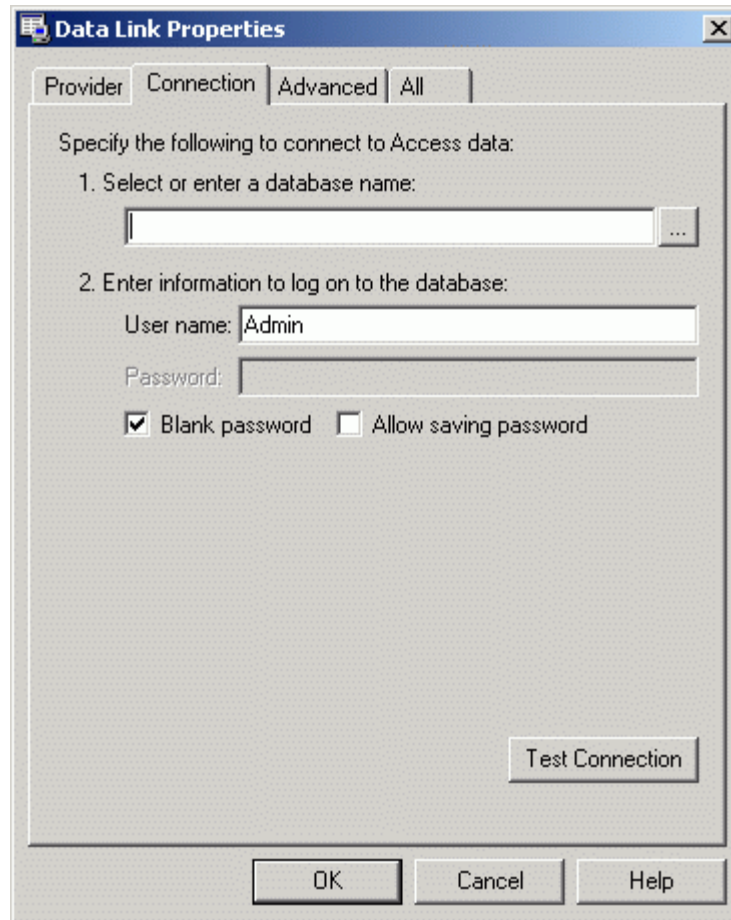
When you select a database type and then click **Connect**, the corresponding OLE DB Provider configuration dialog box opens, as shown below.

For Microsoft SQL Server, select a server name, an authentication type (Windows NT Integrated Security or user name and password), and the desired historical database, as shown in the figure below.



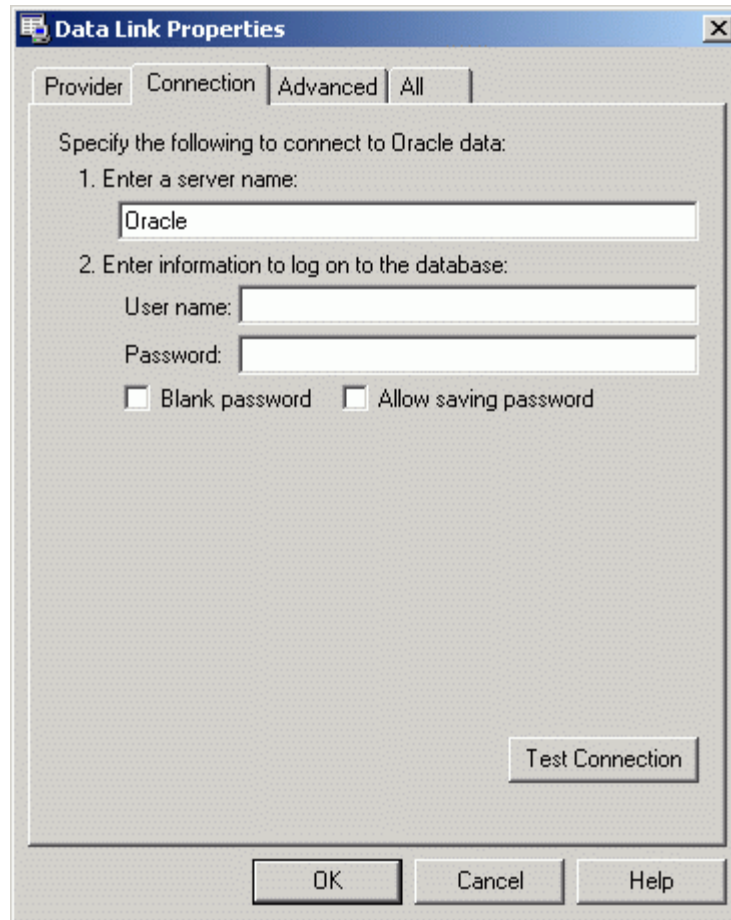
OLE DB Provider Configuration for SQL Server Databases

For Microsoft Access connections, simply navigate to the desired .mdb file by clicking on the browser button, as shown in the figure below.



OLE DB Provider Configuration for Microsoft Access Databases

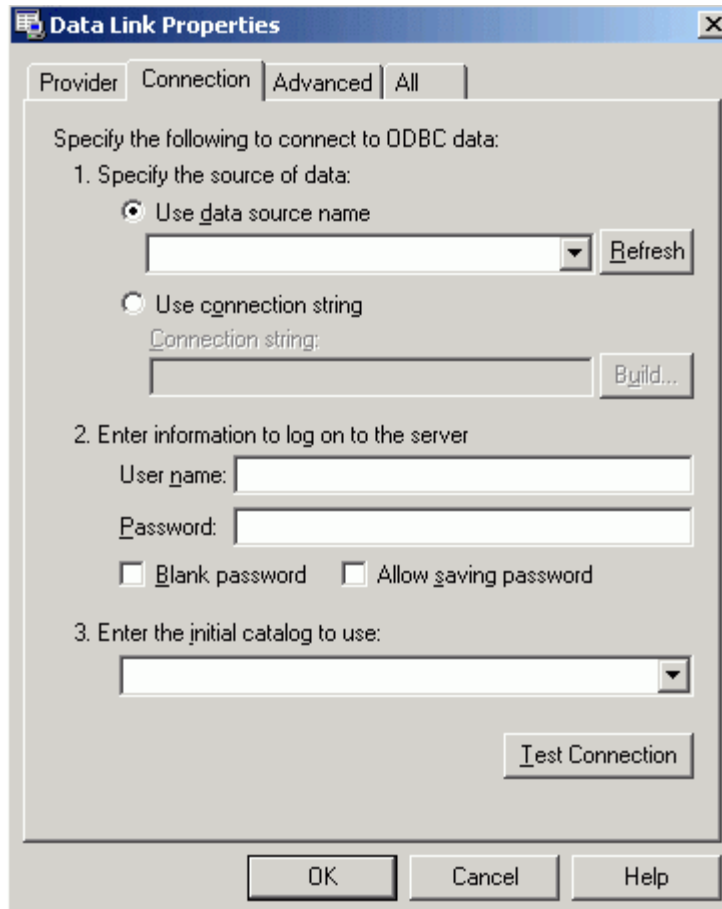
To connect to an Oracle database, fill in the appropriate information, as shown in the figure below.



The image shows a Windows-style dialog box titled "Data Link Properties". It has four tabs: "Provider", "Connection", "Advanced", and "All". The "Connection" tab is selected. Inside the dialog, it says "Specify the following to connect to Oracle data:". Below this, there are two numbered instructions: "1. Enter a server name:" followed by a text box containing "Oracle", and "2. Enter information to log on to the database:" followed by "User name:" and "Password:" labels, each with an empty text box. Below these are two checkboxes: "Blank password" and "Allow saving password", both of which are unchecked. At the bottom right of the main area is a "Test Connection" button. At the very bottom of the dialog are three buttons: "OK", "Cancel", and "Help".

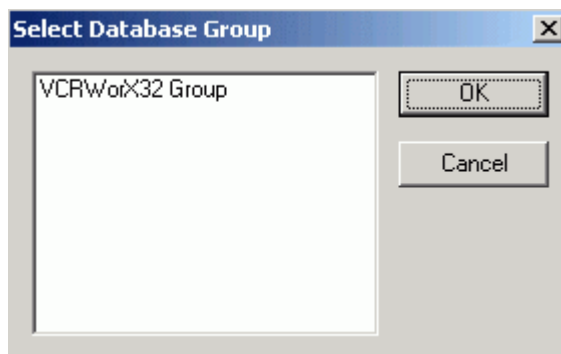
OLE DB Provider Configuration for Oracle Databases

To connect to a MySQL database, specify the desired **Data Source Name**, the **Database Name**, **User**, **Password**, etc., as shown in the figure below.



OLE DB Provider Configuration for MySQL Databases

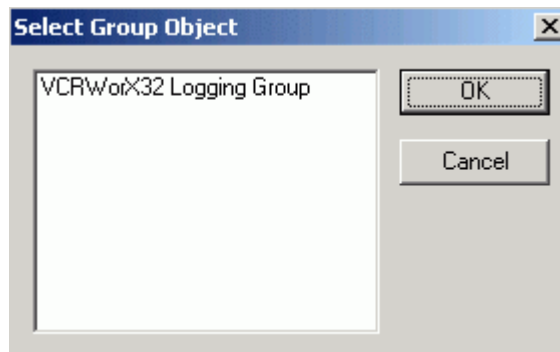
2. Select the database group object name by clicking **DBGroup** in the **TWXSQL Tool Control Properties** dialog box. This opens the **Select Database Object** list box, as shown in the figure below, which lists all of the database group objects configured for data logging in the selected database connection. To select a specific object, double-click the group object name.



Select Database Group List Box

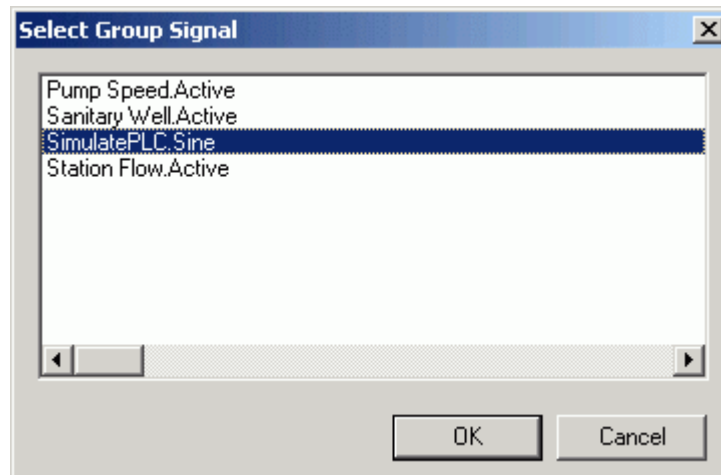
3. Select the logging group name by clicking **Group** in the **TWXSQL Tool Control Properties** dialog box. This opens the **Select Group Object**

list box, shown below, which displays all configured groups for data logging under this database. Again, you can select a desired group by double-clicking the group name in the list.



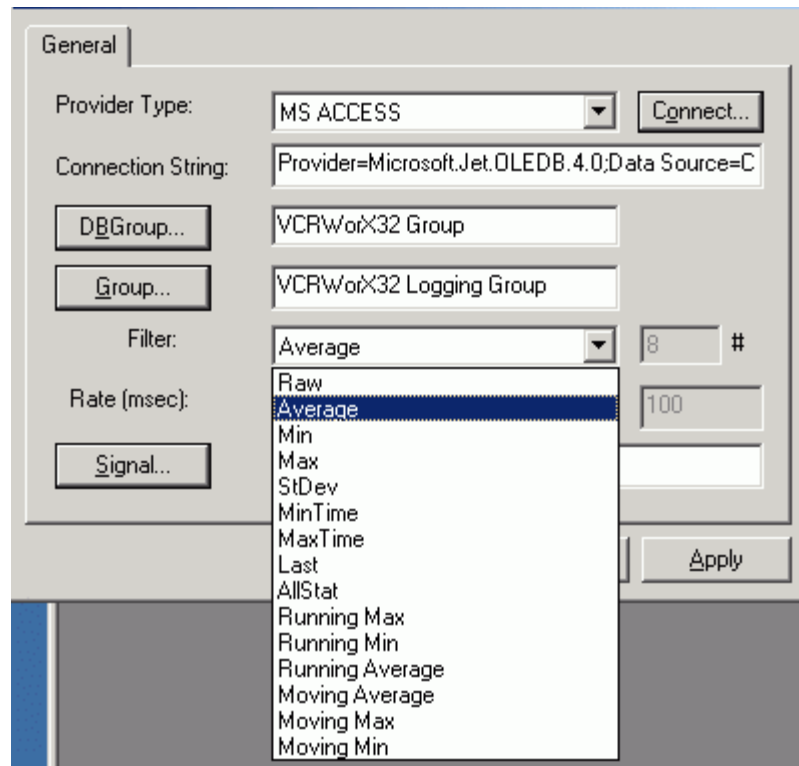
Select Group Object List Box

4. Select the signal name by clicking **Signal** in the **TWXSQL Tool Control Properties** dialog box. This opens the **Select Group Signal** list box. Select a signal name from the list of all the signals configured in the group, and then click **OK**.



Select Group Signal List Box

5. Choose the data filter type to be used in historical data retrieval from the corresponding list in the **Filter** field of the **TWXSQL Tool Control Properties** dialog box, as shown below.



TWXSQL Tool Control Filters

The following filters are currently supported:

- Raw (actual samples stored in the database)
- Average
- Min
- Max
- StDev (standard deviation)
- MinTime (returns actual time for Min value)
- MaxTime (returns actual time for Max value)
- Last (last sample in averaging interval)
- AllStat (all statistical values averaged through the entire period)
- Running Max
- Running Min
- Running Average (exponentially weighted moving average)
- Moving Average

- Moving Max
- Moving Min

If the data filter selected is **Moving Average**, **Moving Max**, or **Moving Min**, the data will be retrieved using a sliding statistics window, taking the last number of samples into consideration.

6. Select the maximum number of samples to retrieve from the **Samples** field of the **TWXSQ SQL Tool Control Properties** dialog box. This is used mostly in the case of a Raw data filter to avoid returning an excessive number of samples.
7. Select the data time resolution (in msec) to be used for data filtering (averaging) from the **Rate (msec)** field of the **TWXSQ SQL Tool Control Properties** dialog box. This interval is used in the case of selecting a data filter other than Raw. It divides the search interval into subintervals of the entered resolution (in msec) and averages all samples within that subinterval. If the value is set to 0, the TrendWorX32 SQL Tool will retrieve the desired average across the entire time-search interval.

During runtime mode, you can use the OLE Automation methods of the TrendWorX32 SQL Tool to perform the following tasks:

- Set a start time and date for historical data retrieval.
- Set an end time and date for historical data retrieval.
- Set the desired data filter (or use the configured one).
- Set the desired time resolution for the historical data (or use the configured one).
- Retrieve historical data for the signal selected during configuration or any other signal.

The TrendWorX32 SQL Tool has OLE Automation methods to set a new database connection by specifying the corresponding OLE DB Provider connection string. In addition, TrendWorX32 SQL Tool supports OLE Automation methods to return all database objects configured for a given database connection, all database object groups, and all tags for a given group. The OLE Automation methods are as follows:

Definition Type	Definitions	Functions
Constant definitions	enum { Raw = 0 Average = 1 Minimum = 2 Maximum = 3 StDev = 4 MinTime = 5 MaxTime = 6	Used in specifying a data filter

	Last = 7 AllStat = 8 Running Max = 9 Running Min = 10 Running Avg = 11 Moving Avg = 12 Moving Max = 13 Moving Min = 14 } TWXSQLTOOLFILTER	
Database type definitions	enum { None = 0 MSAccess = 1 MSSQLServer = 2 Oracle = 4 } } TWXSQLTOOLDBASE	Used in specifying database types
TrendWorX32 SQL Tool error codes	enum { ErrNone = 0 ErrInvalidFilter = 1 ErrDBaseType = 2 ErrConnFailed = 3 ErrQueryFailed = 4 ErrNoResults = 5 ErrNotEnabled = 6 ErrInvalidParam = 7 ErrInvalidRate = 8 ErrInvalidMaxSamples = 9 ErrProcessing = 10 ErrInvalidTDate = 11 ErrInvalidTableName = 12 ErrInvalidSignal = 13 ErrNoData = 14 ErrInvalidFilterOrder = 15 ErrUnableToLoadLangServer = 16 ErrTranslationFailed = 17 ErrOccuredInTranslation = 18 } } TWXSQLTOOLERROR	Identifies the various error codes

[TrendWorX32 SQL Tool ActiveX Control: Properties](#)

[TrendWorX32 SQL Tool ActiveX Control: Methods](#)

[TrendWorX32 SQL Tool ActiveX Control: Events](#)



[TrendWorX32 Reporting](#)

TrendWorX32 SQL Tool ActiveX Control: Properties

BSTR GroupName	Data logging-group name.
BSTR GroupSignal	Signal name within data-logging group.
TWXSQLTOOLFILTER DataFilter	Data-retrieval filter.

BSTR DatabaseObject	Database group name.
long Rate	Data-retrieval rate (in msec).
long MaxSamples	Maximum number of samples to retrieve.
BSTR OLEDBString	OLE DB connection string to historical database.
short DatabaseType	Database type (e.g. Microsoft Access, Microsoft SQL, and Oracle).
short FilterOrder	Filter order to use for "Moving Average," "Moving Max," and "Moving Min."

Notes

The "Rate" value is specified in milliseconds, and it can have a maximum value of 1814400000 (about 21 days). Note that this is the maximum sample-to-sample resolution in the returned historical data. If the rate is set to "0," all the historical data within the search interval will be averaged.

The "MaxSamples" value specifies the maximum number of samples to be retrieved when the data filter is set to "Raw." It can range from 3 to 3600.



[TrendWorX32 SQL Tool ActiveX Control](#)

TrendWorX32 SQL Tool ActiveX Control: Methods

boolean GetDBases(VARIANT* DBNames)	Returns all TrendWorX SQL Server database objects for a given database connection.
boolean GetGroups(VARIANT* GroupNames)	Returns all group objects for a given TrendWorX SQL Server S database object.
BSTR GetGroupTable()	Returns the name of the table that the current group is data-logging.
void SetStartTimeDate(DATE StartDate)	Sets the start time and date for historical data retrieval.
void SetEndTimeDate(DATE EndDate)	Sets the end time and date for historical data retrieval.
boolean GetGroupSignals(VARIANT* SignalNames)	Returns the signal names for the currently selected group.
boolean GetSignalValues(VARIANT* Values, VARIANT* TDates, VARIANT* MSecs, VARIANT* OPCQualities, VARIANT* HDAQualities)	Retrieves the historical data based on the selected ODBC data source name, TrendWorX SQL Server database object, group object, selected signal, desired time resolution, desired data filter, start time and date, and end time and date.
TWXSQLError GetLastError()	Returns the most current error code, if any

operation failed.

void SetLangID(long langID)

Sets the LCID for the language resources and support.

long GetLangID()

Returns the currently selected LCID.

Notes

The "VARIANT*" notation in the method parameters refers to a safe array of data. If the return code is "TRUE," TrendWorX32 SQL Tool will allocate a "SafeArray" of data of the appropriate data type and returned it with the values. The caller MUST free this array of data.

A typical VB type code segment for retrieving data from the returned arrays, and converting them to a string might look like the following:

```
Dim strEntry As String
For i = LBound(values) To UBound(values)
strEntry = "Sample#" + Str(i) + " "
strEntry = strEntry + "Val: " + Str(values(i)) + " "
strEntry = strEntry + "Time: " + Str(TDates(i)) + " "
strEntry = strEntry + "Msec: " + Str(MSecs(i))
```

The "MSecs" array holds the milliseconds time stamp of the sample.

If the data filter is set to "AllStat," the returned array will have the following ordered results:

- Maximum Value
- Maximum Actual Time
- Minimum Value
- Minimum Actual Time
- Average Value
- Standard Deviation
- Last Value

A total of seven values will be computed during the entire interval spanning start time and date up to end time and date.



[TrendWorX32 SQL Tool ActiveX Control](#)

TrendWorX32 SQL Tool ActiveX Control: Events

StartedTask

Fired to indicate that historical data retrieval has started.

FinishedTask

Fired to indicate that historical data retrieval has finished.

AbortedTask

Fired to indicate that historical data retrieval has been aborted due to error.

TrendWorX32 includes VBA-based examples to demonstrate the usage of the Reporting tools. Under the "ICONICS\GENESIS32\Examples\TrendWorX32 Examples\TrendWorX Data Logging and Reporting" folder, there is a TrendWorX32 display called "TrendSQLTool.t32," shown in the figure below, which utilizes VBA forms to demonstrate the usage of TrendWorX32 SQL Tool ActiveX control. In the same folder, there is also a "Readme.txt" file, which contains instructions on how to use the example.



TrendWorX32 SQL Tool ActiveX Control Example

 [TrendWorX32 SQL Tool ActiveX Control](#)

TrendWorX32 SQL Reporting

This application is a stand-alone OLE Automation server capable of generating reports from the TrendWorX32 SQL Data Logging Server databases. It is integrated within the TrendWorX32 container, and it can be used as a stand-alone application.

 [TrendWorX32 Reporting](#)

Toolbar

All actions associated with TrendWorX32 Reporting can be accessed through the menu commands and the toolbar, shown in the figure below.



TrendWorX32 Reporting Toolbar

The TrendWorX32 Reporting toolbar contains the following command buttons. For more information about these commands, refer to the [Menus](#) section.

- New
- Open
- Save
- Add report
- Edit report
- Delete report
- Report status
- Cancel report
- Duplicate report
- Start reports
- Stop reports
- Security configuration
- About
- Help



Menus

The TrendWorX32 Reporting menu bar contains the following menus:

- [File menu](#)
- [Edit menu](#)
- [View menu](#)
- [Actions menu](#)

- [Tools menu](#)
- [Help menu](#)

Some of the commands available on these menus are standard Windows options. However, some options are specific to TrendWorX32 Reporting.



File Menu

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

Command	Shortcut Keys	Function
New	CTRL+N	Creates a new report.
Open	CTRL+O	Opens an existing report.
Save	CTRL+S	Saves the open report to file.
Save As		Saves the report under a different name.
Recent		Opens the most recently used file(s).
Exit		Closes the application.

For further explanations of these commands, please refer to your Microsoft Windows documentation.



Edit Menu

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

Command	Shortcut Keys	Function
Add report	CTRL+A	Adds a report to the configuration.
Edit report	CTRL+E	Edits an already-configured report.
Enable report	CTRL+SHIFT+E	Makes a report available for execution.
Duplicate	CTRL+D	Duplicates the selected report.
Delete report	DEL	Deletes the selected report.



View Menu

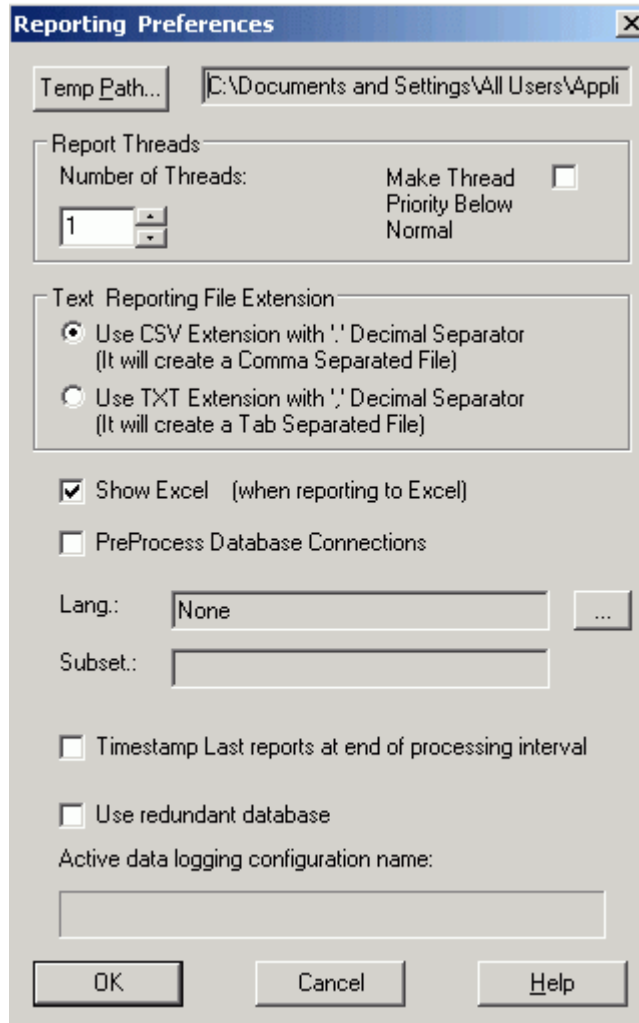
The **View** menu contains the following commands:

Command	Shortcut Keys	Function
---------	---------------	----------

Toolbar	CTRL+T	Toggles the toolbar .
Status bar	CTRL+B	Toggles the status bar.
Select language	CTRL+ALT+U	Prompts you to select a language . If a language resource .dll is present in the TrendWo rX32 "Bin" directory, TrendWo rX32 will automatically load it, convert its menus and dialog boxes to the chosen language , start up in that language .
Reporting preferences	SHIFT+R	Opens the Reportin g Preferen ces dialog box.

Reporting Preferences

The **Report Threads** field in the **Reporting Preferences** dialog box, shown below, specifies the number of threads on which to run reports. Because TrendWorX32 Reporting is a multithreaded application, you can choose to run up to 20 reports simultaneously.



Reporting Preferences Dialog Box

You can select to log to a .csv file or a .txt file. When you are using a .csv file, decimal places will be indicated by the '.' When you are using the .txt file, decimal places will be indicated by the ',' . This is useful if you are using the TrendWorX32 Reporting on systems set up for European standards, which use ',' as a decimal separator).

You can also indicate whether you want to show Microsoft Excel after a report has run. This option is selected by default but may be deselected if you do not wish to view your report after each instance.

A temporary path is used for intermediate table creation. When you click **Temp Path** in the **Reporting Preferences** dialog box, TrendWorX32 Reporting will use a temporary database table to create the report and then

populate Microsoft Excel, .csv files, or the target database. Prior to Version 6.00, this temporary table was created in the source historical database. Thus, it added an additional load to the database while the database was actively data logging and retrieving historical data for clients. Now this temporary table is created inside a temporary Microsoft Access (.mdb) file in the chosen temporary path, and it is deleted as soon as the report is completed.

If **Make Thread Priority Below Normal** is checked in the **Reporting Preferences** dialog box, TrendWorX32 Reporting will start its background report processing threads at a lower priority than normal, so that the system will be able to spend more time executing other more important tasks.

Note: By default, TrendWorX32 Reporting uses only a single thread. This allows for lower usage of CPU on single-processor systems. In general, it is suggested that you increase the number of threads to more than one only if the TrendWorX32 Reporting tool is running on a dedicated node.

The **Language** and **Subset** fields in the **Reporting Preferences** dialog box allow you to select a desired language set from the Language Server. To select a language, click the ... button to the right of the **Language** field.

You also have the option to add a timestamp to reports at the end of the processing interval.

Checking **Use Redundant Database** enables support for redundant HDA servers.

Note: For more information about redundancy, please refer to the DataWorX32 Professional Redundancy help documentation.



Actions Menu

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

Command	Shortcut Keys	Function
Report status	SHIFT+S	Opens the Report Status dialog box, which provides you with pertinent information about the report.

Cancel
report

SHIFT+A

Cancels
the
report
that is
currently
selected
and
running.Start
reports

CTRL+R

Starts
previousl
y
configure
d reports.Stop
reports

CTRL+SHIFT+R

Stops
the
reports
that are
currently
running.**Tools Menu**

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

Command	Shortcut Keys	Function
Security login	F6	Opens the Security Login dialog box. Enter your user name and passwor d to access the Security Server.

Security
configurati
on

F7

Opens the Security Configurator. If a proper login has not occurred, you will be prompted for one at this time.

Working
directory

F8

Opens the **Set Working Directory** dialog box, which enables you to configure a custom directory in which all application configuration files will be stored and retrieved. Click **Browse** to select the directory.



Help Menu

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

Command	Shortcut Keys	Function
Help Topics	F1	Launches the online Help for TrendWo rX32 Reporting .
About Applicatio n		Launches the About Box , which contains informati on about the product version number, copyright, and available disk space. It also contains contact informati on.



Report Configuration

In configuration mode, you can add, edit, or delete reports without report processing taking place. To add a report, select **Add Report** from the **Edit** menu or the toolbar. This opens the Report Configuration Wizard, which is a series of dialog boxes that guide you through the report configuration.

- [General](#)
- [Tags](#)
- [Report](#)
- [Target](#)
- [Schedule](#)
- [Range](#)

Note: For TrendWorX32 Reporting to run properly on your machine, the TrendWorX32 SQL Data Logger must be running.



General

The first step in configuring a report is to pick the database from which the report data will be retrieved. There are two ways that the report will be referenced: in the report window and in the actual report. In the **General** dialog box, shown below, the **Report Name** is what will appear in the window, and the **Target Name** is what will appear in the report. While both of these names are somewhat arbitrary, you must enter a name in both fields. If either field is left blank, you will not be able to continue with the configuration of the report.

A screenshot of the 'General' dialog box. It has a title bar with 'General' and a close button. Inside, there are two text input fields. The first is labeled 'Report Name:' and contains the text 'My Report'. To its right is a checkbox labeled 'Enabled' which is currently unchecked. The second is labeled 'Target Name:' and also contains the text 'My Report'. To its right is a button with a small icon and the text 'Expr...'. At the bottom of the dialog are four buttons: '< Back', 'Next >', 'Cancel', and 'Help'.

General Dialog Box

The **Enabled** check box controls whether the report will be scanned for processing within runtime mode. Any report that is enabled is removed from the active scan list of reports to be processed. All reports that are not enabled will show up with an "X" mark on the left-hand pane tree structure. If you do not select a report to be enabled upon creation, you can select it by using the right-click menu or by selecting **Edit Report** from the **Edit** menu.

Note: If an error is encountered during the adding or editing of an existing report, the report will automatically reset the "Enabled" flag if it is properly

configured. In addition, if a report copy is inserted its "Enabled" flag will be reset by default.

Note: If you choose to include qualities, the maximum number of tags to be included in the report is reduced to a total of 55. If qualities are not selected, the maximum number of tags to be included in the report is 110 tags.

To enter an expression for the **Target Name**, click the **Expr** button. This opens the **Expression Editor**. When you are finished, click **Next** to continue the Report Configuration Wizard. This opens the [Tags](#) dialog box.

Note: For information about the Expression Editor, please see the Expression Editor help documentation.

[Report Configuration](#)

Tags

The next step in the Report Configuration Wizard is the **Tags** dialog box, shown below, which prompts you to select the database group, group, and tags used for the report.

TrendWorX32 Reporting supports the use of a Description text on a per-report-tag basis. This option, if used, will label the resulting data columns of the target report using the defined Description entries in the TrendWorX32 Configurator. By default, when a signal is inserted in the TrendWorX32 Configurator, the Description entry of the signal will be its logging name. You can override the **Description** field, which will be used to label the resulting data column in the final report. If the **Description** field in the TrendWorX32 Configurator is left empty, the logging name will be used to label the resulting data column of the report. The TrendWorX32 Reporting application will still display in the tree view or edit dialogs the report tags using their logging names.

In addition, for the Unicode version of TrendWorX32 Reporting, the **Description** field can be language-aliased to be used with various languages.

Note: TrendWorX32 Reporting also supports report generation across multiple database groups. The **Tags** dialog box lists and sorts all report signals based on database group, type, etc. You can use following procedure for each database group:

1. Click the **DSN** button to establish a connection to the source historical database.
2. Click the **DBase Group** button to select a database group.
3. Click the **Group** button to select a logging group.
4. Click the **Tags** button to select report signals (tags) from the data-logging group for each individual database group and/or logging group. If you have selected multiple tags, you can change the order of the tags for reporting. Simply the desired tags to move up or down by clicking on the Up and Down arrow buttons to the right of the tags list, as shown below.

Tags

Datasource

DSN... Not_Assigned None

User ID: Not_Assigned Password: xxxxxxxx

DBase Group... Not_Assigned

Group... Not_Assigned

Tags...

Tag Name	DSN Name	Type
----------	----------	------

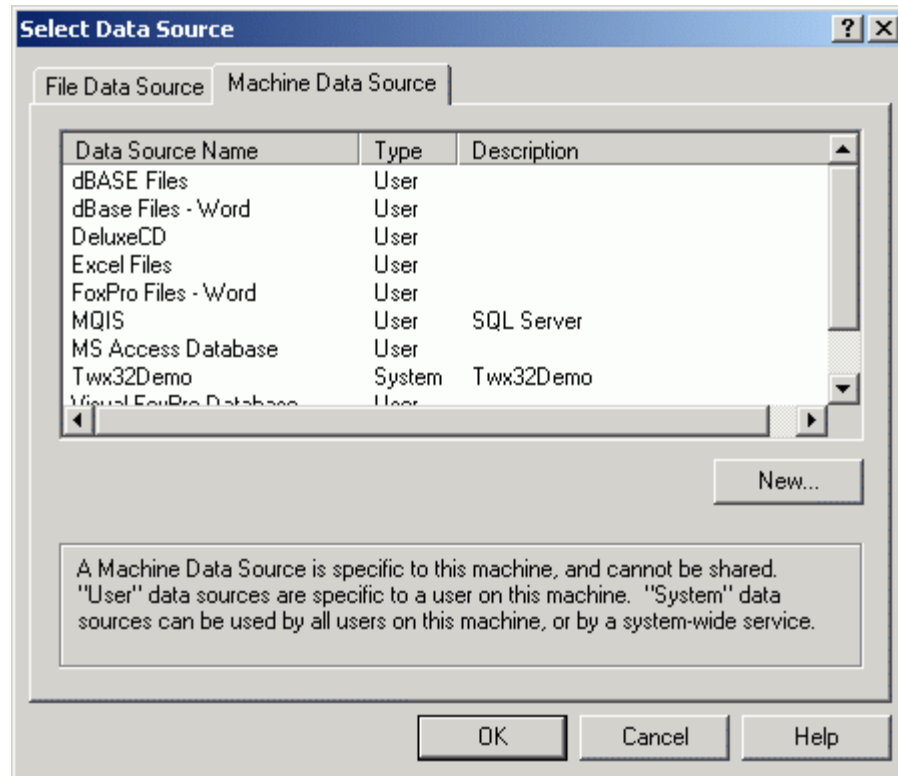
Delete

< Back Next > Cancel Help

Tags Dialog Box

Selecting a Data Source

In the **Data Source** field of the **Tags** dialog box, you can select the source from which the data for the report will come. Click the **DSN** (Data Source Name) button to open the **Select Data Source** dialog box, shown below.



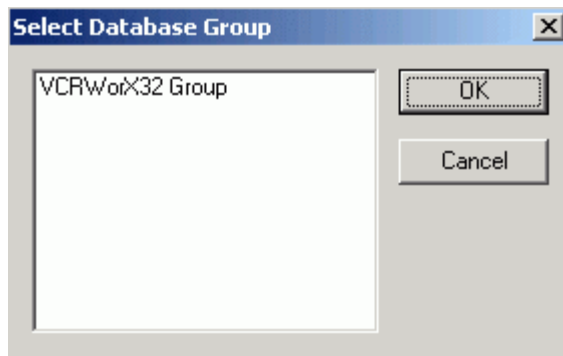
Select Data Source Dialog Box

Select the desired database. This should be a database where the TrendWorX32 SQL Data Logger is configured to log data. As soon as the selection is made, the TrendWorX32 Reporting will fill in the type of the source database, the user ID, and name. When you have selected the data source, click **OK**. To continue the Report Configuration Wizard, click **Next** in the **Tags** dialog box.

Note: If TrendWorX32 Reporting is running on a PC other than the one on which the TrendWorX32 SQL Data Logger is running, you will have to configure an ODBC data source by connecting to the databases on which TrendWorX32 SQL Data Logger is logging data.

Selecting a Database Group

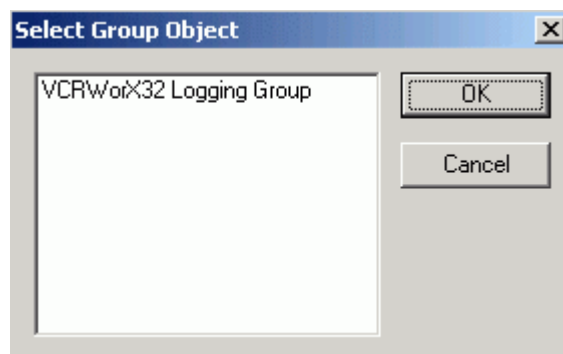
You must choose a database group from those configured for the database you selected in the **Select Data Source** dialog box. Click **DBase Group** in the **Tags** dialog box. Then select the desired database from the **Select Database Group** list box, as shown below.



Select Database Group List Box

Selecting a Group Object

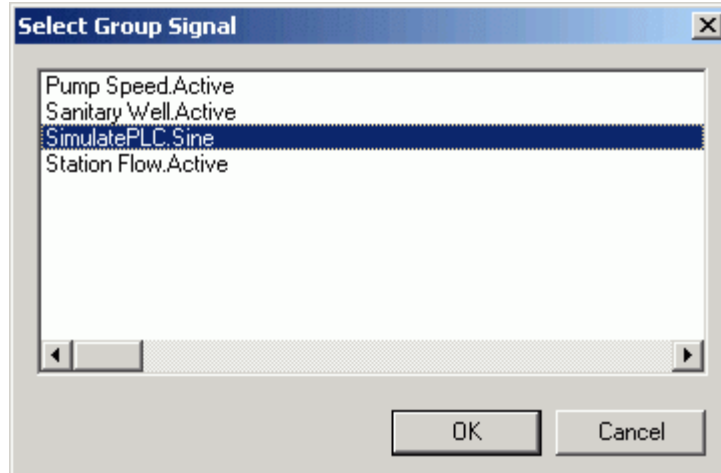
You must select a configured group within the selected database group. Click **Group** in the **Tags** dialog box. This opens the **Select Group Object** list box, shown below, which contains a list of all groups configured for the specific database group. Then select a desired group from the list.



List of Available Groups

Selecting a Group Signals

You must also select the tags that will be used for the basis of the report. Click **Tags** in the **Tags** dialog box. This opens the **Select Group Signals** list box, shown below. Then select the appropriate tags from the list.



List of Available Tags

Once you have made all your selections, click **Next** to continue the Report Configuration Wizard. This opens the [Report](#) dialog box.

[Report Configuration](#)

Report

The next step in the Report Configuration Wizard is the **Report** dialog box, shown below. This allows you to select which data filter to use when creating the report, as well as how often to process data.

Version 7.0 introduced an enhanced data-retrieval system, which utilizes an updated approach to creating historical reports:

- Data filter selection other than Raw will result in data time stamped at the beginning of each subinterval.
- When creating reports using data filters other than Raw, subintervals for which there are no data because of no data-logging activity will be marked as "empty" slots at the corresponding time stamp within the database, .csv or Microsoft Excel target report.
- The historical qualities supported are as follows:
 - ☐ OPCHDA_EXTRADATA 0x0001
 - More data may exist
 - ☐ OPCHDA_INTERPOLATED 0x0002
 - Interpolated data value
 - ☐ OPCHDA_RAW 0x0004
 - Raw data value
 - ☐ OPCHDA_CALCULATED 0x0008
 - Calculated data value
 - ☐ OPCHDA_NOBOUND 0x0010
 - No data found to provide upper or lower bound
 - ☐ OPCHDA_NODATA 0x0020
 - No data collected. Archiving not active
 - ☐ OPCHDA_DATALOST 0x0040
 - Calculation started/stopped/lost

- 0 OPCHDA_CONVERSION 0x0080
 - Scaling / conversion error
- 0 OPCHDA_PARTIAL 0x0100
 - Aggregate value is for an incomplete interval

Report Dialog Box

Note: When running an "AllStats" report for every month or year, the report should return the Min, Max, Avg (and StDev) for the last month or year for the specified tags. Also, only one value for each of the stats should be reported per tag. To do this properly set all values in the **Process Data Every** field of the **Report** dialog box to "0."

The following table describes the different Data Filter types listed in the Data Filter field of the **Report** dialog box.

Filter Type	Description
Raw	All of the actual (raw) samples stored in the database will be retrieved and time-aligned to produce the report. If no samples were stored for a specific time interval, a "NULL" value will be

	stored. You can override this by enabling the Backfill Empty Cells option. As a result, TrendWorX32 Reporting will copy over the previous sample value (if it is available). This is a time-intensive and CPU-consuming report type that may increase hardware requirements or prolong report execution times.
Average	Divides the overall time range of the report into subintervals, and provides the computed average sample.
Min	Divides the overall time range of the report into subintervals, and provides the computed minimum sample.
Max	Divides the overall time range of the report into subintervals, and provides the computed maximum sample.
StDev	Divides the overall time range of the report into subintervals, and provides the computed standard deviation sample.
Last	Divides the overall time range of the report into subintervals, and provides the computed the computed last value sample.
All Statistics	The columns of the report will be the statistical quantities

	"Average," "Min," "Max," "StDev," etc. computed for every subinterval.
RunningMax	Running maximum value.
RunningMin	Running minimum value.
RunningAvg	Exponentially weighted moving average.
Moving Average	Average computed over a sliding time window.
Moving Max	Maximum computed over a sliding time window.
Moving Min	Minimum computed over a sliding time window.
Totalizer	Performs totalizer function.

If **Include Qualities** is checked in the **Report** dialog box, values, OPC Data Access and historical data qualities will be retrieved.

The **Filter Order** field becomes available when the Moving Average, Moving Max, or Moving Min filter is selected under **Data Filter**. If **Filter Order** = X, then X samples in the past from the current time will be used to calculate the Moving Average, Moving Max, or Moving Min.

Other Options

The following table lists the functions of the **Input Signal** and **Totalizer Type** fields in the **Report** dialog box.

Function	Description
Amount per second	The signal represents data in seconds.
Amount per minute	The signal represents data in

	minutes.
Amount per hour	The signal represents data in hours.
Amount per day	The signal represents data in days.
Continuous	When Continuous is checked, the Totalizer will keep counting, starting from 0 until it comes to the end of the trend period, or until the integrated value exceeds the Auto Reset Level (a maximum limit).
Preload	When Preload is checked, the Totalizer commences counting from any value. This is especially useful when totalizing must be "held" (suspended) while a necessary operation is being performed. For example, the pipes are being cleaned and then started from a known value.
Preset	When Preset is checked, you simply specify a preset value. • If you are counting upwards, the value will obviously be positive. • If you are counting downwards, the preset value

Auto reset level

will be negative.

A negative value would be relevant in a situation where a reservoir is being drained, for example. Then the Totalizer function, if enabled for a circular chart, would help monitor the flow rate.

- When counting upwards, when the total exceeds the preset value, it is reset.
- When counting downwards, the preset value is reset at a point when it becomes negative.

Sets the level at which the system will automatically reset, returning to its initial value and disabling everything else.

Once you have made all your selections, click **Next** to continue the Report Configuration Wizard. This opens the [Target](#) dialog box.

[Report Configuration](#)

Target

The next step in the Report Configuration Wizard is the **Target** dialog box, shown below. The target name that you entered will be used as the basis of the generated target report database table, .csv file, .txt file, or Excel worksheet.

Target Dialog Box

The **Target** dialog box has several options for logging a report, including:

- [Logging to a database](#)
- [Logging to a .txt or .csv file](#)
- [Logging to an Excel file](#)

In addition, you can select to:

- Overwrite the report.
- Create a new report with a date.
- Append report data to a database

If you overwrite the report, the target report table or file will be named using the selected target name, and any previous data will be overwritten. If you create a new report, a new table or file will be generated every time the report is run, and its name will consist of the report name and a string representation of the time and date when the report was created.

TrendWorX32 Reporting can also generate a report table within the same database in which the historical data reside. The selection process is identical to the one described earlier.

Note: TrendWorX32 Reporting currently supports only Microsoft Access, Microsoft SQL Server, and Oracle databases as targets configured as machine data sources, *not* file data sources. Refer to the **Tags** section as needed.

Note: When you are creating reports to Excel or to a .txt or .csv file, remember that Excel 2000 can handle a maximum of about 65,000 rows. Make sure that your report will not exceed this amount so that the report will be completed without errors.

Note: When you are creating reports to Excel, the target report file name will always be appended by the time and date extension at the time the report was executed.

TrendWorX32 Reporting will create a table in the target database. The table name will be produced from the target name. If **Create Report with Date** is selected, the time and date of the report creation will be appended to the name. If **Overwrite Report** is selected, the table name will be the same as the report name, and all the existing data will be overwritten.

Now that you have selected the Target Type and location, click **Next** to continue the Report Configuration Wizard. This opens the [Schedule](#) dialog box.



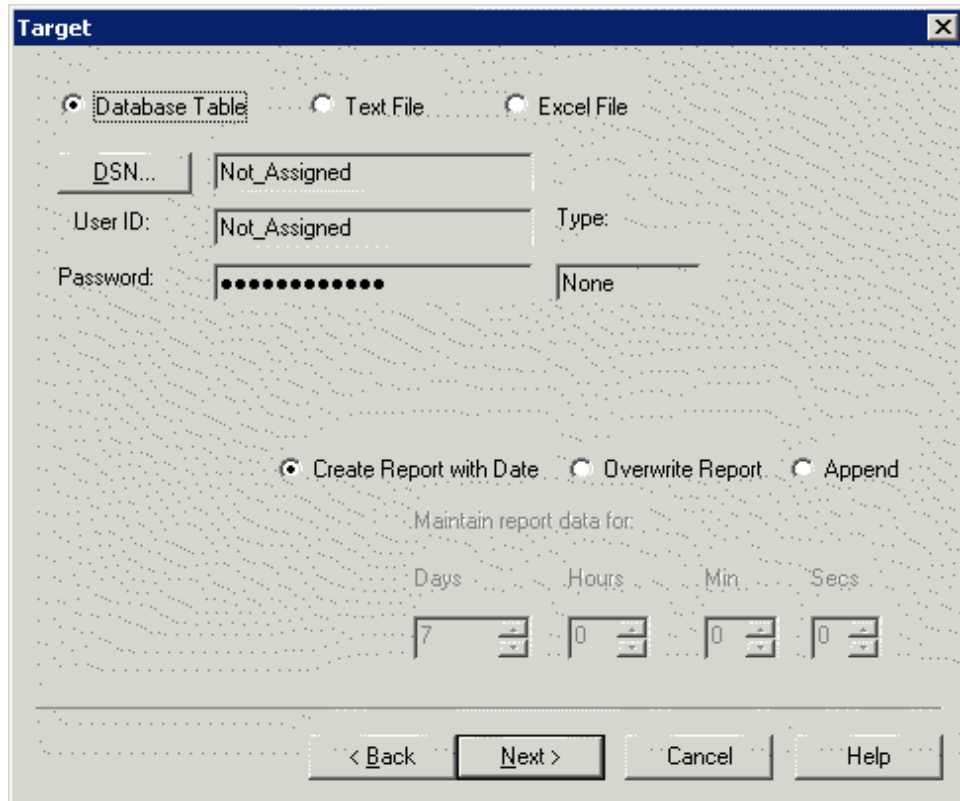
[Report Configuration](#)

Logging to a Database

If you choose to create a report that will write to a database, you must configure a database that will contain all the resulting data from the report. Click the **DSN** button in the [Target](#) dialog box to open the **Select Data Source** dialog box. You can select an existing database, or you create a new database by selecting **New** and following the Report Configuration Wizard. When you make your selection, TrendWorX32 Reporting will fill in the type of the source database, user ID, and name.

The Append button in the Target dialog box writes the data in your report to your database.

Note: If TrendWorX32 Reporting is running on a PC other than the one on which the TrendWorX32 SQL Data Logger is running, you will have to configure an ODBC data source by connecting to the databases on which the TrendWorX32 SQL Data Logger is logging data.

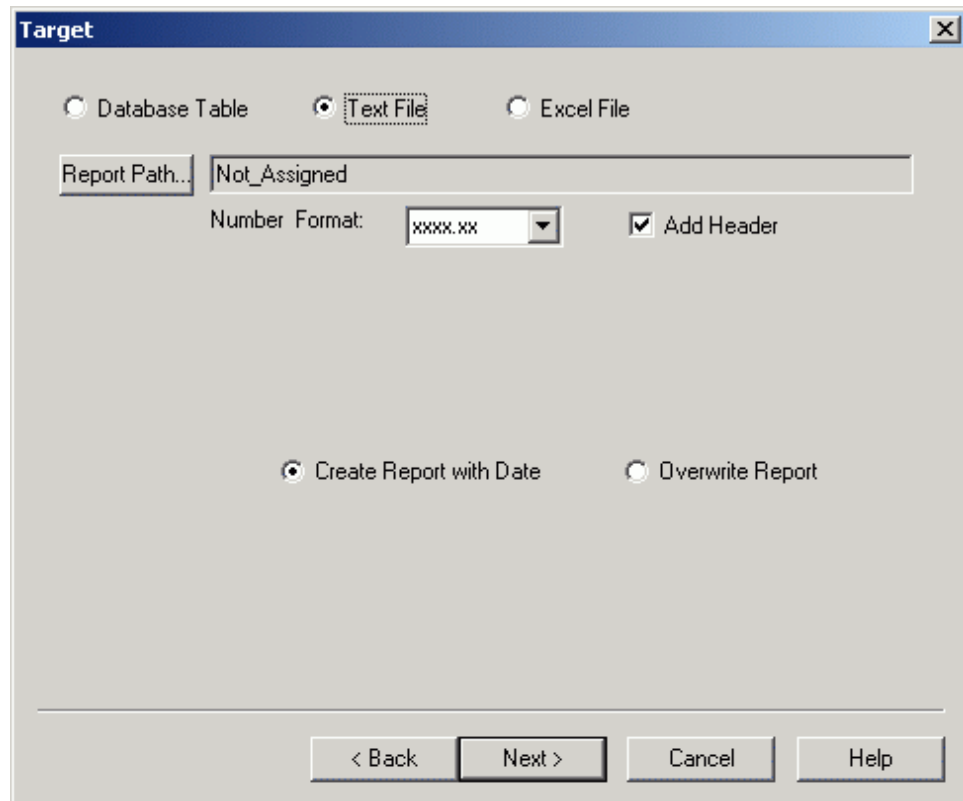


The Target Dialog Box is a window with a title bar labeled 'Target'. It contains three radio buttons at the top: 'Database Table' (selected), 'Text File', and 'Excel File'. Below these are three text input fields: 'DSN...' with the value 'Not_Assigned', 'User ID:' with the value 'Not_Assigned', and 'Password:' with a masked password '.....'. To the right of these fields is a 'Type:' label with a dropdown menu showing 'None'. Below the input fields are three radio buttons: 'Create Report with Date' (selected), 'Overwrite Report', and 'Append'. Underneath these is a section labeled 'Maintain report data for:' with four spin boxes for 'Days' (value 7), 'Hours' (value 0), 'Min' (value 0), and 'Secs' (value 0). At the bottom are four buttons: '< Back', 'Next >', 'Cancel', and 'Help'.

Target Dialog Box

**Logging to a Text File**

To configure the report to log to a .txt or .csv file, select **Text File** on the **Target** dialog box, as shown in the figure below. Then click **Report Path** to select the directory in which the file will be created. After you have selected the location for the file, select the **Number Format** for the data from the report. If **Add Header** is checked, TrendWorX32 Reporting will add a report header to the file.



Logging to a Text File in the Target Dialog Box



Logging to an Excel File

To configure the report to log to an Excel file, select **Excel File** from the **Target** dialog box, as shown in the figure below.

Target

☐ Database Table
 ☐ Text File
 ☒ Excel File

Template...: Not_Assigned
 Report Path...: Not_Assigned
 Publish: ☐ HTML Path...: Not_Assigned
 WSheet: ☐ Select...: Not_Assigned
 Row: 5 ☐ Auto Print ☒ Add Header ☒ Format ☐ Protect
 Column: 5 ☒ Create Report with Date ☐ Overwrite Report
 Email: ☐
 To: Not_Assigned
 Subject: Not_Assigned

< Back Next > Cancel Help

Logging to an Excel File in the Target Dialog Box

TrendWorX32 Reporting has the following integration with Microsoft Excel:

- Microsoft Excel template support.
- HTML output support.
- Report header support.
- E-mail support.
- Enhanced directory management.
- Enhanced performance integration with Microsoft Excel, which greatly reduces the time required to populate an Excel spreadsheet with the report data. This functionality may require the utilization of more memory resources.
- Date format support, which includes the month name for international support.
- TrendWorX32 Reporting now supports an internally created password for creating Microsoft Excel reports. Based on this internal password, the resulting Microsoft Excel worksheet will be locked and no data changes will be allowed.

When creating reports in Microsoft Excel, you need to create and customize (if desired) a source Excel workbook, which should be saved as a template.

TrendWorX32 treats each Excel workbook selected from the **Template** button as an Excel template, and it creates an Excel-based report using the user-configured format and layout in the source workbook. To configure an Excel report, you need to do the following:

1. Create an Excel workbook to be the source of the report formatting and layout.
2. Click **Template** in the **Target** dialog box to select the workbook.
3. Click **Report Path** in the **Target** dialog box to assign a path where the generated reports will be created.
4. Select **Publish** if HTML output of the Microsoft Excel-based report is desired.
5. If the **Publish** option is checked, click **HTML Path** to select a target directory for the HTML output of the report.
6. If you want to select a specific worksheet from the source workbook, you should enable the **WSheet** option and then click **Select** to display a list of all workbooks available. Then select the desired workbook.
7. You can specify the starting cell of the Excel workbook in which the report will be stored by supplying the desired values for the **Row** and **Column** fields.

In addition to the above, you have the following options:

- **Auto Print.** If enabled, the generated report will be sent automatically to the current Windows default printer.
- **Add Header.** If enabled, TrendWorX32 Reporting will add an informative header to the generated report.
- **Format.** If enabled, TrendWorX32 Reporting will automatically format the generated report and adjust the column width so that the values within the column cells will be readable. If you want to provide a custom format and layout, this option should not be checked. As an example, to hide a certain column in the generated report, calculate its location and make its width zero within Microsoft Excel.
- **Protect.** Enables password protection based on an internally created password for Microsoft Excel reports. The resulting Microsoft Excel worksheet will be locked and no data changes will be allowed.
- **Create Report with Date.** This is the only option for Microsoft Excel reporting. If enabled, the target worksheet will use the report name, and the time and date extension will be appended to the name.
- **Overwrite Report.** Not available for Excel reporting.

- **E-Mail.** If enabled, you can fill a subject and a recipient e-mail address list, such as "myname@mycompany.com; yourname@yourcomapny.com" (with the list items separated by a semicolon). TrendWorX32 Reporting, upon completing the report, will attach it to an e-mail message and send the message to all listed recipients.



Schedule

The next step in the Report Configuration Wizard is the **Schedule** dialog box, shown below.

Schedule Dialog Box

Reports can be configured to execute:

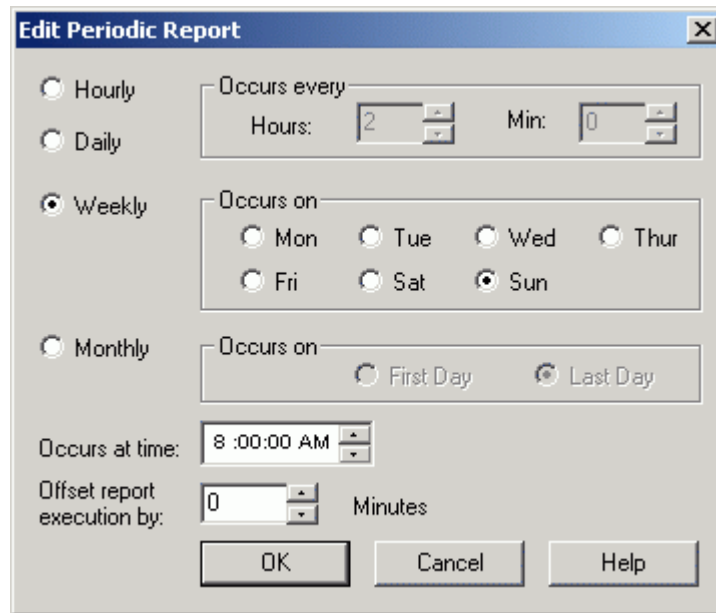
- Immediately
- One time
- Periodically
- On event

If the report is configured to run **Immediately**, it will execute as soon as possible within runtime, and after completion its enabled flag will be reset. All immediate reports have their priority of execution boosted internally by TrendWorX32 Reporting over periodic reports.

If the report is configured to run **One Time**, you must pick the desired time and date for the report to run by clicking on **Set** in the **Schedule** dialog box to enable time and date selection. To select the date, click the down arrow in the **Date** field, and then choose a date from the calendar. After completion, the report's enabled flag and time field will be reset.

Clicking **Set** opens the standard Microsoft Calendar control. Select a desired date by double-clicking on the date.

If the report is configured to be **Periodic**, you must select the reports periodic schedule by clicking **Change**. This opens the **Edit Periodic Report** dialog box shown below.



The **Edit Periodic Report** dialog box contains the following controls:

- Frequency Selection:** Radio buttons for **Hourly**, **Daily**, **Weekly** (selected), and **Monthly**.
- Occurs every (Hourly/Daily):** A group box containing **Hours:** (spinner set to 2) and **Min:** (spinner set to 0).
- Occurs on (Weekly):** A group box containing radio buttons for days of the week: **Mon**, **Tue**, **Wed**, **Thur**, **Fri**, **Sat**, and **Sun** (selected).
- Occurs on (Monthly):** A group box containing radio buttons for **First Day** and **Last Day** (selected).
- Occurs at time:** A time spinner set to 8 :00:00 AM.
- Offset report execution by:** A spinner set to 0, followed by the text **Minutes**.
- Buttons:** **OK**, **Cancel**, and **Help**.

Edit Periodic Report Dialog Box

Periodic reports can run:

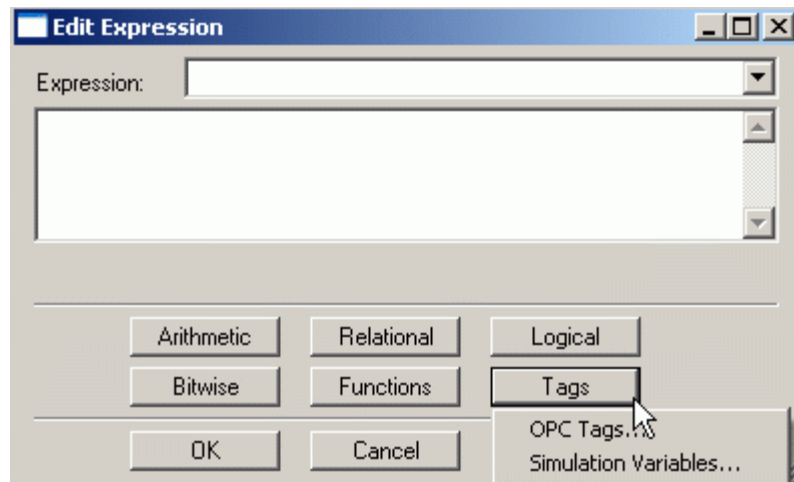
- Hourly
- Daily
- Weekly (on a desired day of the week)
- Monthly (on the first and/or last day of month)

You can select the desired time of the report for execution. In addition, you can select a report execution offset interval, which will add a delay between consecutive report executions. You can also run reports faster than one hour.

Note: Daily, weekly, and monthly reports will automatically create a report based on the last day's, week's, or month's data from the report execution time. Hourly reports will span data based on how many hours they are configured to run, while event-based reports can be configured to retrieve data spanning a selected report time span.

If the report is configured to run on an event, you must configure an event expression by clicking the **Expr**, which opens the **Edit Expression** dialog box, shown below.

Note: For information about the Expression Editor, please see the Expression Editor help documentation.



Expression Editor

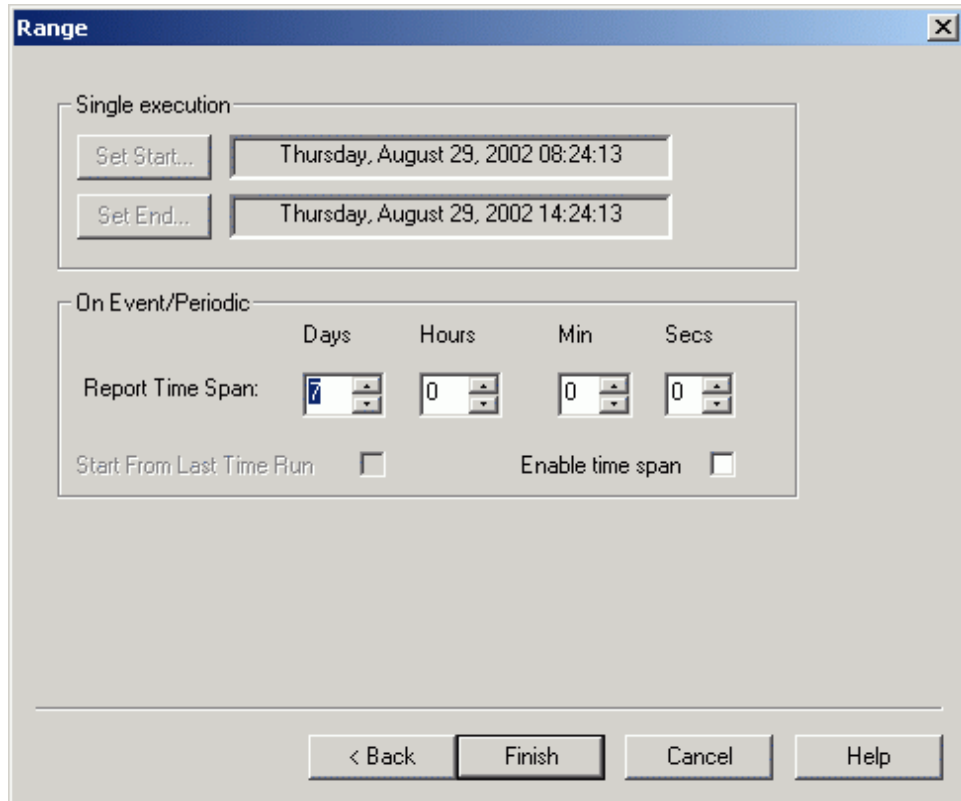
Event-based reports provide finer control over timing of report execution. The event scan time is currently 1 second.

Once you have configured the report schedule, click **Next** to continue the Report Configuration Wizard. This opens the **Range** dialog box, which enables you to pick the time range of the report. This is basically the time interval over which historical data will be retrieved and processed. It is available for immediate and one time report types.

[Report Configuration](#)

Range

The next step in the Report Configuration Wizard is the **Range** dialog box, shown below.



The Range dialog box is titled "Range" and contains two main sections: "Single execution" and "On Event/Periodic".

Single execution: This section contains two buttons, "Set Start..." and "Set End...", each followed by a text box showing a date and time. The "Set Start..." button is followed by "Thursday, August 29, 2002 08:24:13" and the "Set End..." button is followed by "Thursday, August 29, 2002 14:24:13".

On Event/Periodic: This section contains a "Report Time Span:" label followed by four spin boxes for "Days", "Hours", "Min", and "Secs". The "Days" spin box is set to 7, "Hours" to 0, "Min" to 0, and "Secs" to 0. Below these spin boxes are two checkboxes: "Start From Last Time Run" and "Enable time span", both of which are currently unchecked.

At the bottom of the dialog box are four buttons: "< Back", "Finish", "Cancel", and "Help".

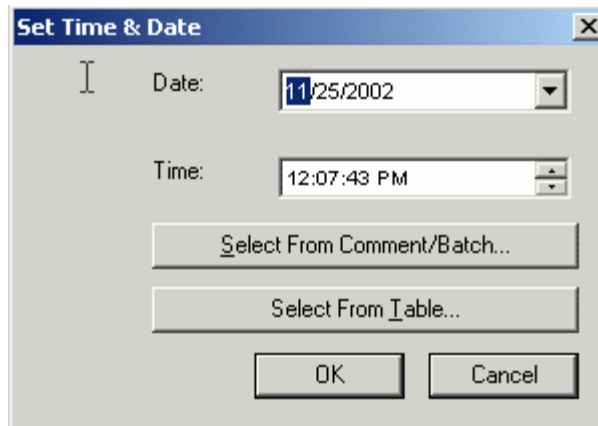
Range Dialog Box

If the report is configured to execute **Immediately** or **One Time**, you can configure the single execution start and end times by clicking **Set Start** and **Set End**.

If the report is configured to run **On Event**, you can configure the **Report Time Span** for which historical data should be processed. In addition, if **Start From Last Time Run** is checked, the starting time for data retrieval of the report will be the last time the report was executed.

You can also enable the **Report Time Span** for **Periodic** reports. This way you will be able to select a time span different from the one determined by the periodic frequency of the report alone.

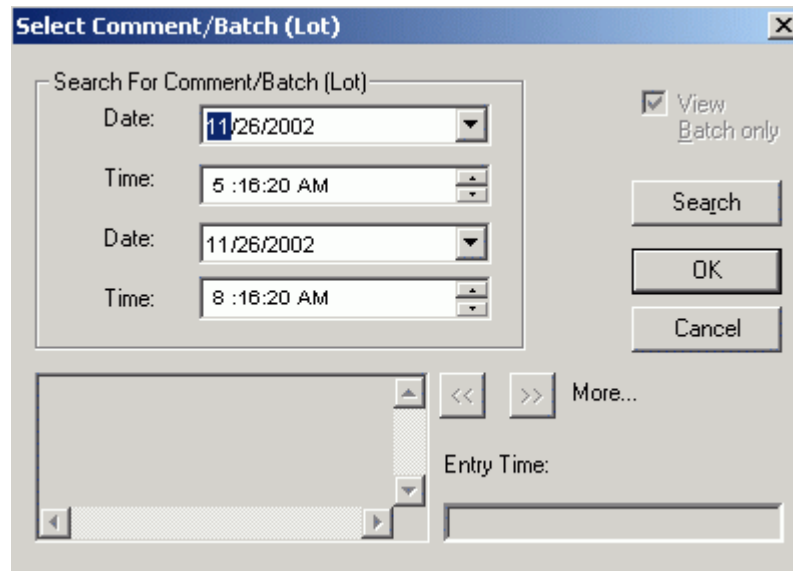
Clicking on **Set Start** or **Set End** in the **Range** dialog box opens the **Set Time and Date** dialog box, shown below. Within this dialog, the you can click the down arrow in the **Date** field to open the Calendar control for selecting a desired date, adjust the desired time in the time picker control or select the time based on a given historical table or operator comment and batch entry.



Set Time and Date Dialog Box

Clicking **Select From Comment/Batch** in the **Set Time and Date** dialog box opens the **Select Comment/Batch (Lot)** dialog box, shown below. This enables you to:

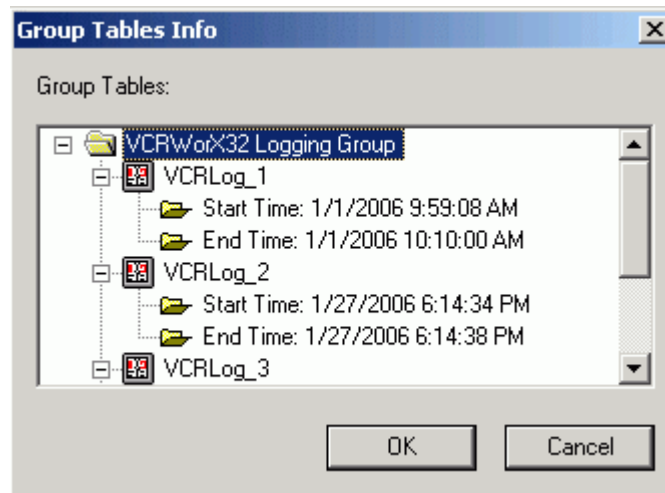
- Select a start time and date. To select the date, click the down arrow in the **Date** field, and then choose a date from the calendar.
- Select an end time and date.
- Click **Search** to initiate the search.
- Scroll through multiple operator comments and batch entries once the search is completed use the << and >> buttons. Once you locate the desired one, click **OK**.



Set Comment/Batch (Lot) Dialog Box

Note: Selecting **View Batch Only** filters out the historical operator comments to display only the related batch (lot) entries.

Clicking **Select From Table** in the **Set Time and Date** dialog box opens the **Group Tables Info** dialog box, shown below. This displays the table entries for the last-accessed database connection so that you can select a starting and ending time and date based on the available historical tables.



Group Tables Info

Note: The Unicode version has enhanced language-aliasing support, including automatic value scaling. You need to configure the Language Server as well as configure language aliases in the TrendWorX32 Configurator. Once all language aliases are configured, TrendWorX32 Reporting will automatically scale all historical values to those corresponding to the startup language. While TrendWorX32 Reporting is in runtime mode, all subsequent events generated by you selecting a different language will result only in updating the user interface to match the currently selected language, but not the scaling equations used in the report generation.

Finally, clicking **Finish** will complete the new report configuration and addition. If any errors are found, the new report's enabled flag will be reset and will be marked with an "X".

[Report Configuration](#)

Editing a Report

In configuration mode, you can also edit, delete, and copy a report using the **Edit** menu commands or the toolbar. Alternatively, you can select a report and right-click to display the configuration mode report inspector menu, as shown below.

The editing tabs in the **TWXReport** dialog box, shown below, are identical to the series of dialog boxes in the [Report Configuration Wizard](#).

General | Tags | Report | Target | Schedule | Range

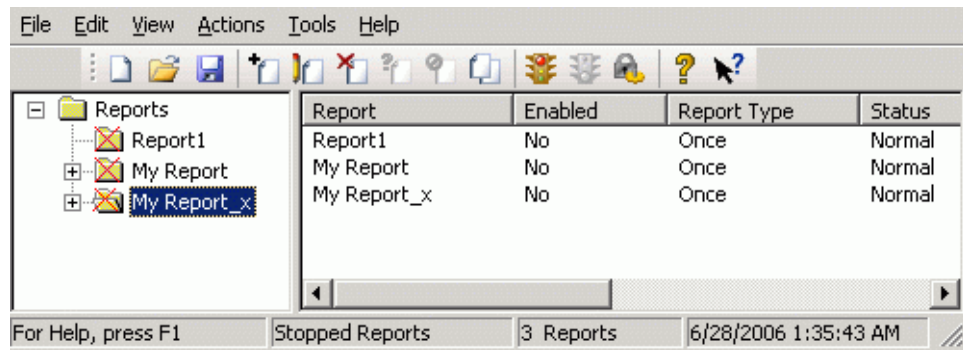
Report Name:
 ☐ Enabled

Target Name:

OK Cancel Help

Editing a Report

To copy a report, select an existing report and click **Duplicate** on the **Edit** menu or the main toolbar. This will copy all the configuration data of the original report, modify the original report name, and reset the enabled flag, as shown below.



Copying a Report

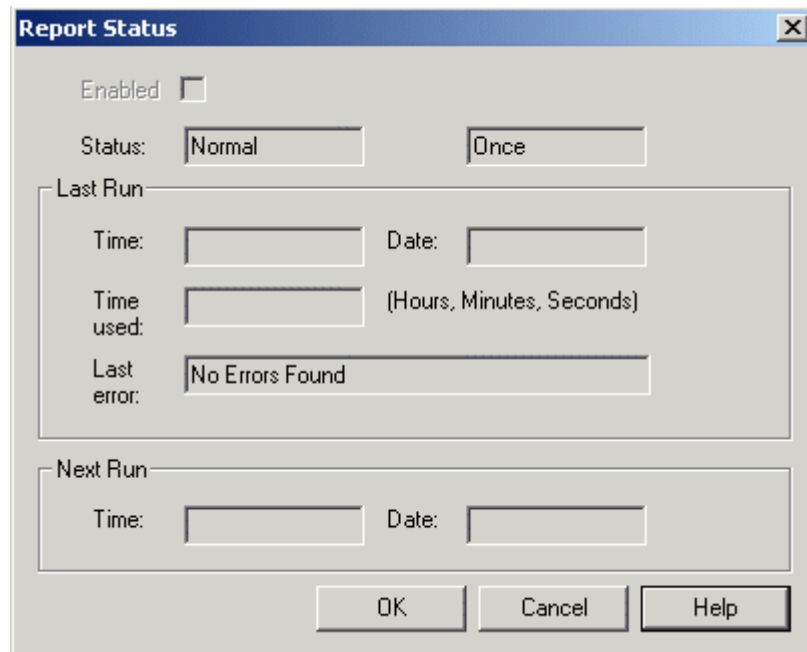


Runtime Mode

To enter runtime mode, select **Start Reports** from the **Actions** menu. TrendWorX32 Reporting will enable scanning of all reports and process them accordingly. In addition, both the right-hand and left-hand panes will be updated to reflect the latest status of reports. In runtime mode, you can:

- Add new reports.
- Edit existing reports.
- Duplicate reports.
- Delete reports.
- Cancel reports.
- Get report status information.

You can fully edit or delete reports in runtime mode, as long as they are not running. If the reports are currently running, or are in "stand by" mode, they need to be cancelled first and then edited (or deleted). In addition, you can get runtime status information by clicking **Report Status** on the **Actions** menu. This opens the **Report Status** dialog box, shown below.



The **Report Status** dialog box is a standard Windows-style window with a title bar and a close button. It contains the following elements:

- Enabled:** A checkbox that is currently unchecked.
- Status:** Two text boxes, the first containing "Normal" and the second containing "Once".
- Last Run:** A group box containing:
 - Time:** and **Date:** text boxes.
 - Time used:** a text box followed by the text "(Hours, Minutes, Seconds)".
 - Last error:** a text box containing "No Errors Found".
- Next Run:** A group box containing:
 - Time:** and **Date:** text boxes.
- Buttons:** "OK", "Cancel", and "Help" buttons at the bottom right.

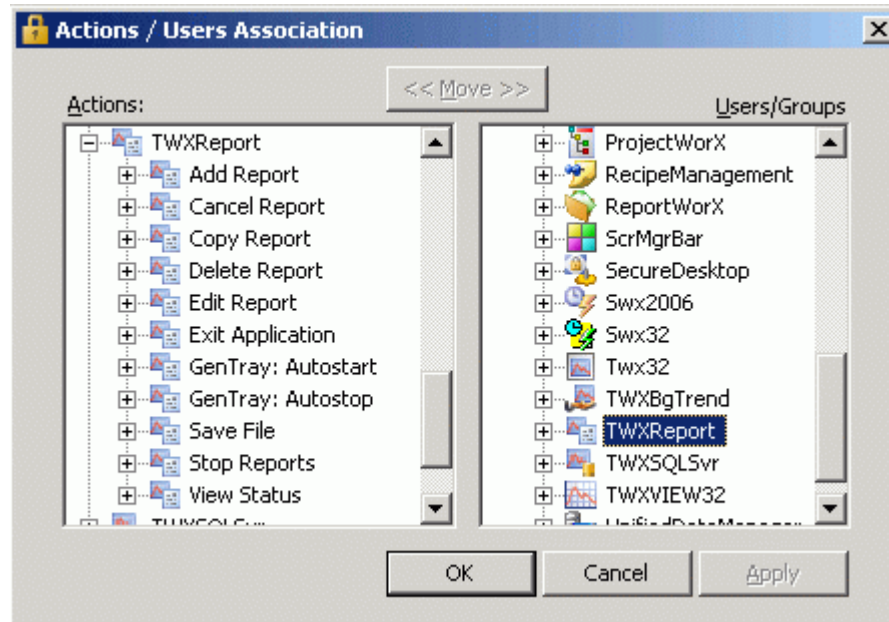
Report Status Dialog Box

TrendWorX32 Reporting is a multithreaded OLE Automation server, which allows for concurrent report processing. Within runtime, it allocates a pool of threads for simultaneous report scanning and processing. The number of threads can be adjusted via a registry setting. The default thread pool number is one worker thread. Note that if the target of all reports is an Excel worksheet, the reports will be serialized such that only one report accesses Excel at a time. In this case, you should *not* increase the thread number used by TrendWorX32 Reporting in the registry. In addition, TrendWorX32 Reporting implements its own scheduling mechanism, which boosts the priority of reports when they are configured to run immediately.

Since TrendWorX32 Reporting may use the source database for creating temporary tables and actual report generated tables, it is very important that you allow for the source database to have enough disk space for TrendWorX32 Reporting to perform its table creation.

 [TrendWorX32 Reporting](#)
TrendWorX32 Reporting Security

TrendWorX32 Reporting is interfaced to the GENESIS32 Security Server and currently supports the following security actions within runtime mode, as shown in the figure below.

**Security Server**
 [TrendWorX32 Reporting](#)
GenEvent Registrar Integration

TrendWorX32 Reporting is interfaced to the GenEvent Registrar Server. If it is enabled within the GenEvent Registrar configuration tools, it will post events to the GenEvent Registrar and subsequently to the NT Event Logger. The following events will be posted:

- Started Reports (Severity 500 - Normal Event)
- Stopped Reports (Severity 500 - Normal Event)
- Error in Starting Reports (Severity 800 - Abnormal Event)
- Report Deleted (Severity 650 - Warning Event)
- Report Cancelled (Severity 650 - Warning Event)
- Report Executed (Severity 500 - Normal Event) if no errors occurred
- Report Executed (Severity 650 - Warning Event) if errors occurred



TrendWorX32 Reporting OLE Automation Interfaces

OLE Automation

The TrendWorX32 Reporting Application registers with the OLE ROT (Running Object Table) as a single-instance OLE Automation server. If it has started, all clients can connect to it by using a "GetObject" call (i.e. connecting to the running instance of reporting). The TrendWorX32 Reporting OLE Automation server currently supports the following OLE Automation properties and methods.

[OLE Automation Properties](#)
[OLE Automation Methods](#)



OLE Automation Properties

boolean Visible	Shows/hides application window.
boolean Maximized	Maximizes/minimizes application window.



OLE Automation Methods

void Refresh()	Redraws the view.
void Open(BSTR FileName)	Opens a configuration file.
void StartReports()	Starts runtime mode.
void StopReports()	Exits runtime mode.
booleanAreReportsRunning()	Returns true if in runtime mode.
voidClose(boolean SaveChanges, BSTR file)	Exits the report.

A set of OLE automation methods has been added to allow for flexible report manipulation. For these OLE automation methods to be used effectively, you should call first the **GetReportNames** method to get a listing of all the currently available reports and then use automation to set the desired time span, etc. Depending on the status of each report or the periodic schedule, a VBA-enabled application can filter which reports will be available to the operator for manipulation. Note that the best way to make use of operator-based triggering of report execution within a VBA-enabled application, such as GraphWorX32, is to configure immediate report types, and do not enable the desired reports. Then, the operator through the newly added OLE Automation support will be able to start or stop existing reports.

These methods are:

VARIANT GetReportNames()	Returns a Safearray with all the report names.
short GetReportStatus(BSTR ReportName)	Returns an enumerated type TWXREPORTSTATUS with the report status for each report by name, which can be of the following enumerated values: RptStatusOk = 0 RptStatusEditing = 1 RptStatusDeleting = 2 RptStatusExecuting = 3 RptStatusExiting = 4 RptStatusPaused = 5 RptStatusCancelled = 6 RptStatusStandby = 7 RptStatusNotFound = 8 RptStatusConfigErrors = 9 RptStatusConfigure = 10
short GetReportScheduleType(BSTR ReportName)	Returns an enumerated type TWXREPORTSCHEDULE with the desired report schedule type: RptNotFound = 0 RptImmediate = 1 RptOnce = 2 RptPeriodic = 3 RptEvents = 4
boolean SetReportTimeSpan(BSTR ReportName, DATE StartDateTime, DATE EndDateTime)	Sets the report time range for immediate type of reports which currently are not executing.
boolean SetReportTarget(BSTR ReportName, BSTR ReportTarget)	Sets the report target name for the desired report. The report has to be disabled first using the DisableReport method if necessary.
short RunReport(BSTR ReportName)	Executes the desired report returning a TWXREPORTSTATUS status enumeration with the status of the report. Works only in runtime mode.
short StopReport(BSTR ReportName)	Stops execution of the desired report. Works only in runtime mode. It returns a TWXREPORTSTATUS status enumeration with the status of the report.
short AddNewReport(BSTR ReportName, BSTR ReportTargetName, long ReportDataFilter, long ReportFilterOrder, boolean Qualities, boolean CopyOver, long ProcessDataSecs, long ProcessDataMSecs)	Adds a new report with basic configuration settings. The report is initially disabled. It returns a TWXREPORTEDIT edit status code. The ReportDataFilter is of type GFWREPORT_DATA_FILTERS. The ReportFilterOrder parameter is used only in Moving filters, and the ProcessDataSecs, ProcessDataMSecs define the report processing interval for non-Raw reports
short SetReportCSVTarget(BSTR ReportName, BSTR ReportCSVPath, BSTR ReportCSVNumberFormat, boolean AddHeader, boolean CreateNewWithDate)	Sets the target of an existing report to be a .csv file. The report needs to be disabled first, and the ReportCSVPath refers only to the file path. It returns a TWXREPORTEDIT edit

	status code.
short SetReportEXCELTarget(BSTR ReportName, BSTR ReportXLSTemplate, BSTR ReportXLSPath, boolean PublishHTML, BSTR ReportHTMLPath, boolean UseXLSWKSHEET, BSTR ReportWKSHEETName, long ReportXLSRow, long ReportXLSColumn, boolean ReportAutoprint, boolean AddHeader, boolean ReportAutoFormat, boolean ReportXLSProtect, boolean ReportXLSEmail, BSTR ReportXLSEmailAddress, BSTR ReportXLSEmailSubject)	Sets the target of an existing report to be an Microsoft Excel workbook. The report needs to be disabled first, and the ReportXLSPath refers only to the file path. It returns a TWXREPORTEDIT edit status code.
short SetReportDBTarget(BSTR ReportName, BSTR DBConnString, long DBType, boolean CreateNewWithDate)	Sets the target of an existing report to be a database table. The report needs to be disabled first, and the DBConnString refers to an ADO ODBC connection string. It returns a TWXREPORTEDIT edit status code. The DBType is of TWXREPORTSDBTYPE enumeration.
short SetReportImmediate(BSTR ReportName, DATE StartTime, DATE EndTime)	Sets the schedule of an existing report to be of Immediate type. The report needs to be disabled. It returns a TWXREPORTEDIT edit status code.
short SetReportOnce(BSTR ReportName, DATE DateToRun, DATE StartTime, DATE EndTime)	Sets the schedule of an existing report to be of Single execution type (Once). The report needs to be disabled. It returns a TWXREPORTEDIT edit status code.
short SetReportEvent(BSTR ReportName, BSTR EventCondition, long ReportRangeSecs, boolean StartFromLastTime)	Sets the schedule of an existing report to be of Event execution type. The report needs to be disabled. It returns a TWXREPORTEDIT edit status code. The EventCondition represents an OPC tag-based expression.
short SetReportHourly(BSTR ReportName, DATE TimeToRun, long Hours, long Minutes, long OffSetMinutes, boolean UseReportTimeSpan, long ReportTimeSpanSecs)	Sets the schedule of an existing report to be of periodic (Hourly) execution type. The report needs to be disabled. It returns a TWXREPORTEDIT edit status code. The UseReportTimeSpan enables the time range of the report not to be defined by the periodic interval but to be set by the user. The TimeToRun is the desired time of the first report execution.
short SetReportDaily(BSTR ReportName, DATE TimeToRun, long OffSetMinutes, boolean UseReportTimeSpan, long ReportTimeSpanSecs)	Sets the schedule of an existing report to be of periodic (Daily) execution type. The report needs to be disabled. It returns a TWXREPORTEDIT edit status code. The UseReportTimeSpan enables the time range of the report not to be defined by the periodic interval but to be set by the user. The TimeToRun is the desired time of the first report execution.
short SetReportWeekly(BSTR ReportName, DATE TimeToRun, long DayOfTheWeek, long OffSetMinutes, boolean UseReportTimeSpan, long ReportTimeSpanSecs)	Sets the schedule of an existing report to be of periodic (Weekly) execution type. The report needs to be disabled. It returns a TWXREPORTEDIT edit status code. The UseReportTimeSpan enables the time range of the report not to be defined by the periodic interval but to be set by the user. The TimeToRun is the desired time of the first report execution. The DayOfTheWeek refers

	to the day of the week for the report to execute (with Sunday = 1 and Saturday = 7).
short SetReportMonthly(BSTR ReportName, DATE TimeToRun, boolean FirstDayOfWeek, long OffSetMinutes, boolean UseReportTimeSpan, long ReportTimeSpanSecs)	Sets the schedule of an existing report to be of periodic (Weekly) execution type. The report needs to be disabled. It returns a TWXREPORTEDIT edit status code. The UseReportTimeSpan enables the time range of the report not to be defined by the periodic interval but to be set by the user. The TimeToRun is the desired time of the first report execution. The DayOfTheWeek refers to the day of the week for the report to execute (with Sunday = 1 and Saturday = 7).
short EnableReport(BSTR ReportName)	Enables a given report for execution.
short DisableReport(BSTR ReportName)	Disables a given report for execution.
short AddReportTag(BSTR ReportName, BSTR TagLoggingName, BSTR TagDescrText, BSTR TagEngUnits, BSTR TagDBConnString, long DBType, BSTR TagDBBaseGroupName, BSTR TagLogGroupName, long TagDBHandle)	Adds a new report tag to an existing report. The report needs to be disabled. It returns a TWXREPORTEDIT edit status code. The TagLoggingName is the one defined in the TrendWorX32 Configurator, while the TagDBConnString parameter refers to the ADO ODBC connection string for the historical database. The TagDBHandle parameter is the index of the specific tag in the _Tags table of the historical database.
short DeleteReport(BSTR ReportName)	This method deletes a report. The report needs to be disabled. It returns a TWXREPORTEDIT edit status code.
short SetReportTotalizerParams(BSTR ReportName, long TotalizerUnits, long TotalizerType, long TotalizerDirection, double TotalizerValue, double TotalizerAutoResetLevel)	This method sets the Totalizer filter related parameters. The TotalizerUnits can be any of: - Seconds = 0 - Minutes = 1 - Hours = 2 - Days = 3 The TotalizerType can be of type: - Continuous = 0 - PreLoad = 1 - Preset = 2 The TotalizerDirection can be 0, for counting upwards, or 1 for counting downwards The Totalizer value is simply the Preload or Preset value depending on the Totalizer type.

For these OLE automation methods to be used effectively, you should call first the **GetReportNames** method to get a listing of all the currently available reports and then use automation to set the desired time span, etc. Depending on the status of each report or the periodic schedule, a VBA-enabled application can filter which reports will be available to the operator for manipulation. Note that the best way to make use of operator-based triggering of report execution within a VBA-enabled application, such as

GraphWorX32, is to configure immediate report types, and do not enable the desired reports. Then, the operator through the newly added OLE Automation support will be able to start or stop existing reports.

Example

The sample code below adds a simple Microsoft Excel report:

```
Private Sub cmdReports_Click()

Dim reports As TWXReport.Document

Set reports = GetObject(, "TWXReport.Document")
```

'Add a report

```
reports.AddNewReport TextBox1.Text, TextBox1.Text, TWXReport.Raw, 8,
False, False, 60, 0

reports.SetReportImmediate TextBox1.Text, Now, Now - 1

reports.SetReportEXCELTarget TextBox1.Text, "D:\Logging Tests\TWX
EXCEL Templates\Production -- Daily Yield.xlt", "D:\Logging Tests\Test
Reports", False, "", False, "", 3, 3, False, True, True, False, False, "", ""
```

'Add a tag

```
reports.AddReportTag TextBox1.Text, "SimulatePLC.Random",
"SimulatePLC.Random", "", "Provider=MSDASQL.1;Extended
Properties=""DSN=TWXREPORT_MSJET;DBQ=D:\Logging Tests\Logging
Data\TWXREPORT_MSJET.mdb;DriverId=25;FIL=MS
Access;MaxBufferSize=2048;PageTimeout=5;UID=admin;""", 1, "MSJET",
"JetGroup", 3
```

'Execute Report

```
reports.StartReports

reports.EnableReport TextBox1.Text

reports.RunReport TextBox1.Text
```

ProjectWorX32 Support

Several 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.

The following is a simple example of using the OLE automation methods of TrendWorX32 Reporting added for Version 7.x. It assumes that the TrendWorX32 Reporting application is already running. If not, a Shell command can be used to start it.

```
Private Sub cmdReports_Click()

Dim reports As TWXReport.Document

Set reports = GetObject( "TWXReport.Document")

'Add a report

reports.AddNewReport "MyReport", "MyReport", TWXReport.Raw, 8, False,
False, 60, 0

reports.SetReportImmediate "MyReport", Now, Now - 1
```

```
'reports.SetReportEXCELTarget "MyReport", "D:\Logging Tests\TWX
EXCEL Templates\Production -- Daily Yield.xlt", "D:\Logging Tests\Test
Reports", False, "", False, "", 3, 3, False, True, True, False, False, "", ""
```

```
reports.SetReportCSVTTarget "MyReport", "D:\Logging Tests", "x.xxxxx",
False, False
```

```
'Add tags
```

```
reports.AddReportTag "MyReport", "SimulatePLC.Random",
"SimulatePLC.Random", "", "Provider=MSDASQL.1;Extended
Properties=""DSN=TWXREPORT_MSJET;DBQ=D:\Logging Tests\Logging
Data\TWXREPORT_MSJET.mdb;DriverId=25;FIL=MS
Access;MaxBufferSize=2048;PageTimeout=5;UID=admin;""", 1, "MSJET",
"JetGroup", 3
```

```
' Get Report Names
```

```
Dim ReportNames As Variant
```

```
ListBox1.Clear
```

```
ReportNames = reports.GetReportNames
```

```
For i = LBound(ReportNames) To UBound(ReportNames)
```

```
ListBox1.AddItem ReportNames(i)
```

```
Next i
```

```
'Execute Report
```

```
reports.StartReports
```

```
reports.EnableReport "MyReport"
```

```
reports.RunReport "MyReport"
```

```
'Delete Report
```

```
'reports.DeleteReport MyReport
```

```
End Sub
```



[OLE Automation](#)

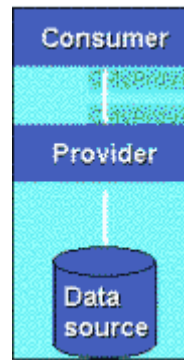
TrendWorX32 OLE DB Provider

TrendWorX32 OLE DB Provider

OLE DB is the dominant data access methodology from Microsoft that allows high-performance data access from any data source. Any tabular data is viewable through OLE DB regardless of whether it came from a database. This flexibility gives developers a tremendous amount of power.

In version 7.x, the TrendWorX32 OLE DB Provider included the following new features:

- New database connection dialog
- Enhanced data retrieval support



OLE DB Architecture

The design of OLE DB includes the concept of consumer and provider. The figure above shows a graphical representation of the OLE DB system. The consumer represents the traditional client. The provider places data into a tabular format and returns it to the client. A provider is a set of COM components that contains a series of interfaces. Because these are standard interfaces, any OLE DB consumer can access data from any provider. Since providers are COM objects, consumers can access them in any language (C++, Basic, Java, and so on).

ADO is a lightweight object set for accessing data from any OLE DB provider, using a consistent API within any programming language and or environment.

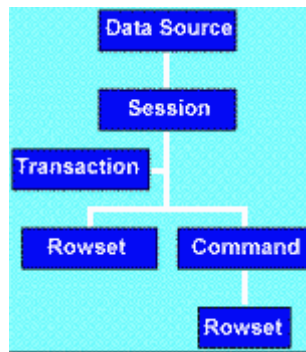
The TrendWorX32 SQL Server application, which is ICONICS' next-generation enterprise data-logging and historical data server, uses relational databases as storage media to log process historical data. The historical data can be accessed through a variety of methods, such as:

- Using the OPC Historical Data Access interface which is supported in TrendWorX32 SQL Server
- Using the TrendWorX32 SQL Tool ActiveX Control to retrieve process data in a VBA-scripting environment
- Using the TrendWorX32 Reporting, which creates a variety of historical data reports in databases, .csv and .txt files, and Excel

- Using ADO within any programming environment to parse TrendWorX32 SQL Server historical tables

The TrendWorX32 OLE DB Provider, which encapsulates all the work needed to retrieve historical data spanning multiple historical data tables, interpolate and or extrapolate for missing values, and perform custom averaging of data over desired time periods.

OLE DB providers are a set of COM objects that transfer data from a durable source to a consumer. The OLE DB provider places those data in a tabular format in response to calls from a consumer. Providers can be simple or complex. A provider may return a table, allows the client to determine the format of that table, or perform operations on those data. Each provider implements a standard set of COM objects to handle requests from the client. A provider may implement optional COM objects to provide additional functionality. The figure below shows the different COM objects a provider may implement.



COM Components in an OLE DB Provider

COM Components and Their Uses

Component	Interfaces	Comments
Data Source	[mandatory] IDBCreateSession [mandatory] IDBInitialize [mandatory] IDBProperties [mandatory] IPersist [optional] IDBDataSourceAdmin [optional] IDBInfo [optional] IPersistFile [optional] ISupportErrorInfo	Connection from the consumer to the provider. The object is used to specify properties on the connection, such as user ID, password, and data source name. The object can also be used to administer a data source (create, update, delete, tables, and so on).
Session	[mandatory] IGetDataSource [mandatory] IOpenRowset [mandatory] ISessionProperties [optional] IDBCreateCommand [optional] IDBSchemaRowset [optional] IIndexDefinition [optional] ISupportErrorInfo [optional] ITableDefinition [optional] ITransaction	The session object represents a single conversation between a consumer and provider. It is somewhat similar to the ODBC HSTMT in that many sessions can be active simultaneously. The session object is the primary link to get to OLE DB functionality. In order to get to a command, transaction, or rowset object, you go through the session object.

Command	[optional] ITransactionJoin	The command object handles operations on data such as queries. It can handle statements with or without parameters. The command object is also responsible for handling bindings for parameters and output columns. A binding is a structure that contains information about how a column, in a rowset, should be retrieved. It contains information such as ordinal, data type, length, status, and so on.
	[optional] ITransactionLocal	
	[optional] ITransactionObject	
	[mandatory] IAccessor	
	[mandatory] IColumnsInfo	
	[mandatory] ICommand	
	[mandatory] ICommandProperties	
	[mandatory] ICommandText	
	[mandatory] IConvertType	
	[optional] IColumnsRowset	
	[optional] ICommandPrepare	
	[optional] ICommandWithParameters	
	[optional] ISupportErrorInfo	
Rowset	[mandatory] IAccessor	The rowset object represents the data from the data source. The object is responsible for the bindings of that data and any basic operations (update, fetch, movement, and so on) on the data. You will always have a rowset object to contain and manipulate data.
	[mandatory] IColumnsInfo	
	[mandatory] IConvertType	
	[mandatory] IRowset	
	[mandatory] IRowsetInfo	
	[mandatory] IRowsetIdentity	
	[optional] IColumnsRowset	
	[optional] IConnectionPointContainer	
	[optional] IRowsetChange	
	[optional] IRowsetLocate	
	[optional] IRowsetResynch	
	[optional] IRowsetScroll	
	[optional] IRowsetUpdate	
Transaction	[optional] ISupportErrorInfo	The transaction object defines an atomic unit of work on a data source and determines how those units of work relate to each other. This object is not directly supported by the OLE DB provider templates. That is, you create your own object.
	[mandatory] IConnectionPointContainer;	
	[mandatory] ITransaction	
	[optional] ISupportErrorInfo	

Each COM component represents a series of COM interfaces. Some COM interfaces are mandatory, while others are optional. By implementing the mandatory interfaces, a provider guarantees a minimum level of functionality that any client should be able to use. By implementing the optional interfaces, a provider can have more functionality and a richer feel to the client. The client should always call "QueryInterface" to determine whether a provider supports a given interface.

The TrendWorX32 OLE DB Provider is a simple read-only provider, which implements all of the mandatory interfaces listed in the table above, with the exception of the transaction-based component. TrendWorX32 OLE DB Provider supports, among others, the following:

- Similar historical replay functionality to the OPC Historical Data Access specification
- Raw or time-processed data-retrieval over a user-defined time interval
- Multiple historical tag data retrieval
- Dedicated "trend SQL" query parsing and execution capability
- Historical database browsing/prompting capability
- Schema information support for the "Recordset" structure it returns
- Support for a "MaxRecords" property to limit the total number of returned samples
- Multiple historical data filter support

The TrendWorX32 OLE DB Provider internally uses a high-speed data-retrieval mechanism based on the native OLE DB providers for the historical databases configured in the TrendWorX32 SQL data-logging server.

[Using the TrendWorX32 OLE DB Provider](#)



[TrendWorX32 Reporting](#)

Using the TrendWorX32 OLE DB Provider

In this section, we will discuss how ICONICS TrendWorX32 OLE DB Provider can be used to retrieve data from the TrendWorX32 SQL Server historical database capability.

[Connecting to TrendWorX32 OLE DB Provider](#)

[Retrieving Historical Data](#)

[Returned Recordset Structure](#)

[Example Application Using ICONICS TrendWorX32 OLE DB Provider](#)



[TrendWorX32 OLE DB Provider](#)

Connecting to TrendWorX32 OLE DB Provider

To connect to the TrendWorX32 OLE DB Provider, you must set the corresponding properties or ask the provider to prompt and provide database navigation and browsing tools.

In order to have the provider prompt, a typical VBA type of code would be:

Dim twx As New ADODB.Connection

' Set Provider

twx.Provider = " ICONICS.TWXOLEDB "

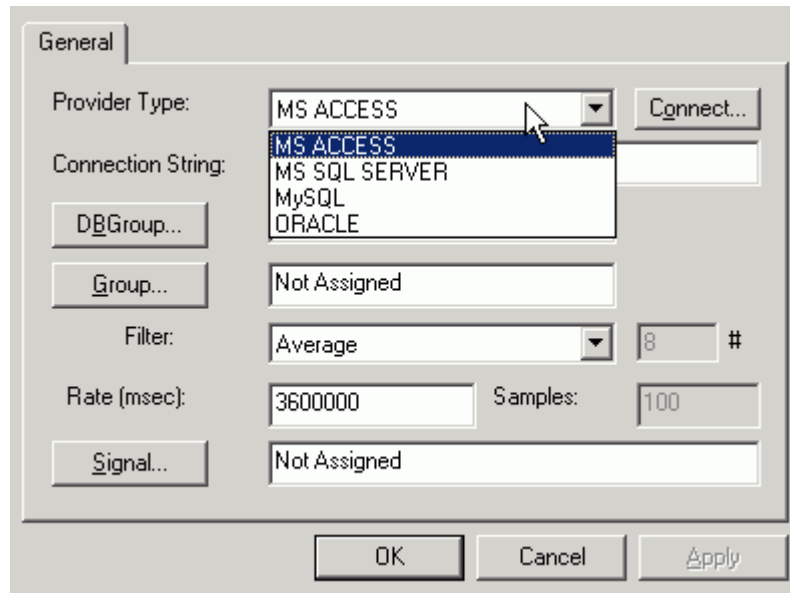
' Set up for Prompting

twx.Properties("Prompt") = 1

'Open Connection

twx.Open

Use the **TWXSQ Tool Control Properties** dialog box, shown below, to connect to a database. This dialog box has only one tab, the **General** tab shown below, which prompts you to complete the configuration.



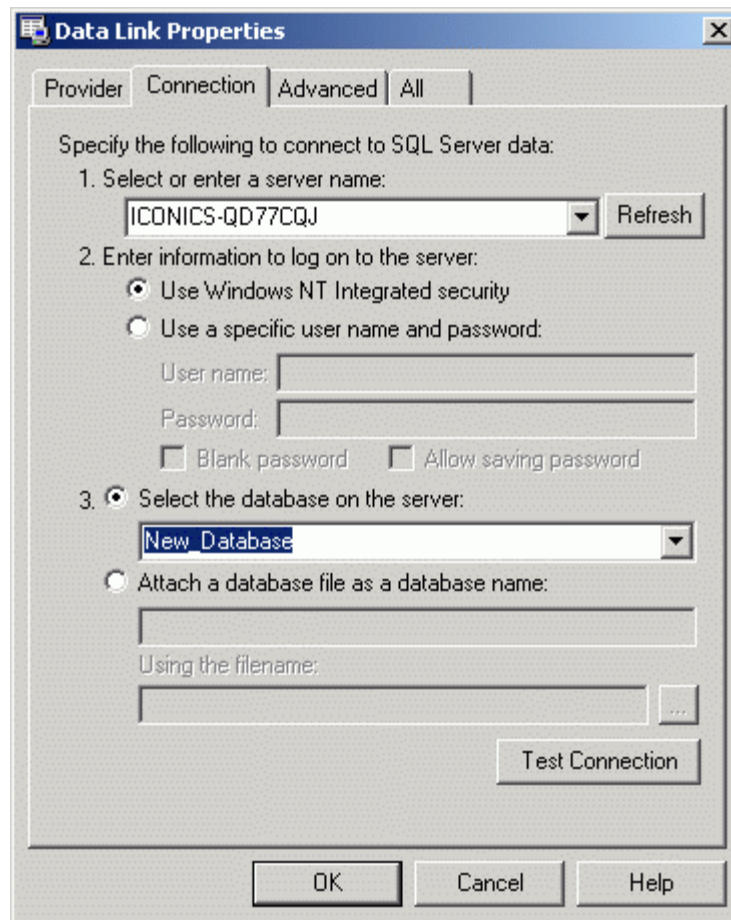
TWXSQ Tool Control Properties Dialog Box

To configure the SQL Tool control, from the **Provider Type** list box in the **TWXSQ Tool Control Properties** dialog box, select the OLE Database (DB) Provider type relating to the TrendWorX32 historical database. Currently, the following OLE DB Providers are supported:

- Microsoft Access
- Microsoft SQL Server
- Oracle
- MySQL

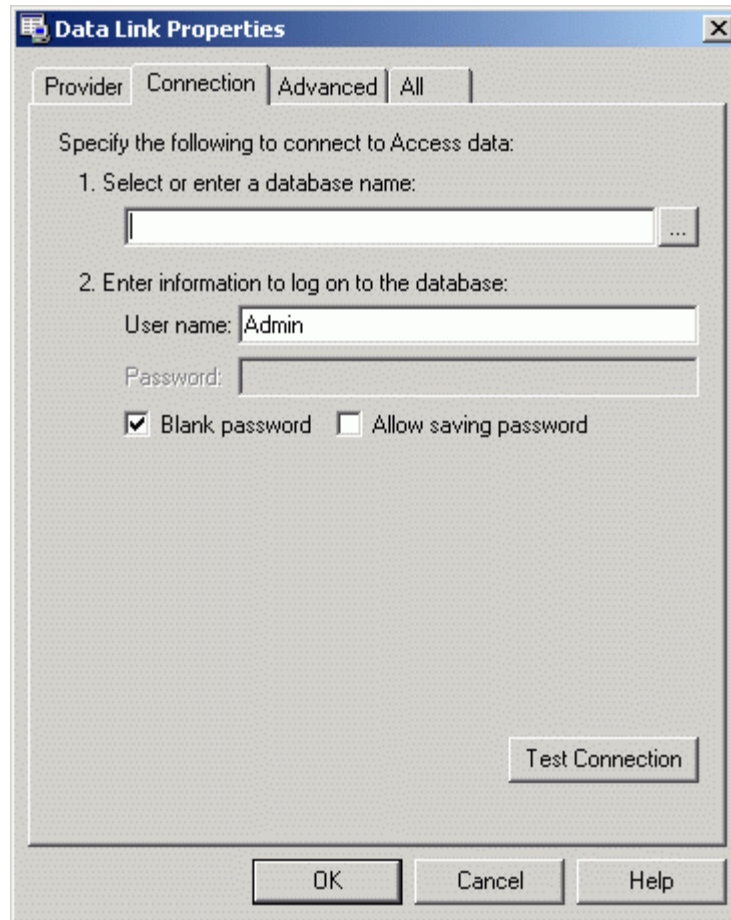
When you select a database type and then click **Connect**, the corresponding OLE DB Provider configuration dialog box opens, as shown below.

For Microsoft SQL Server, select a server name, an authentication type (Windows NT Integrated Security or user name and password), and the desired historical database, as shown in the figure below.



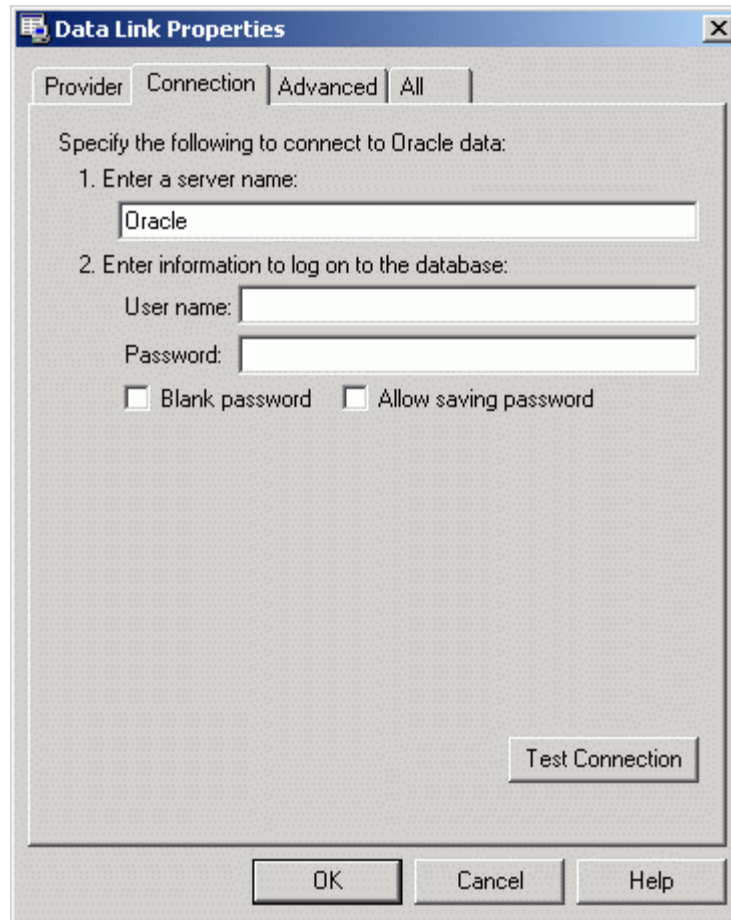
OLE DB Provider Configuration for SQL Server Databases

For Microsoft Access connections, simply navigate to the desired .mdb file by clicking on the browser button, as shown in the figure below.



OLE DB Provider Configuration for Microsoft Access Databases

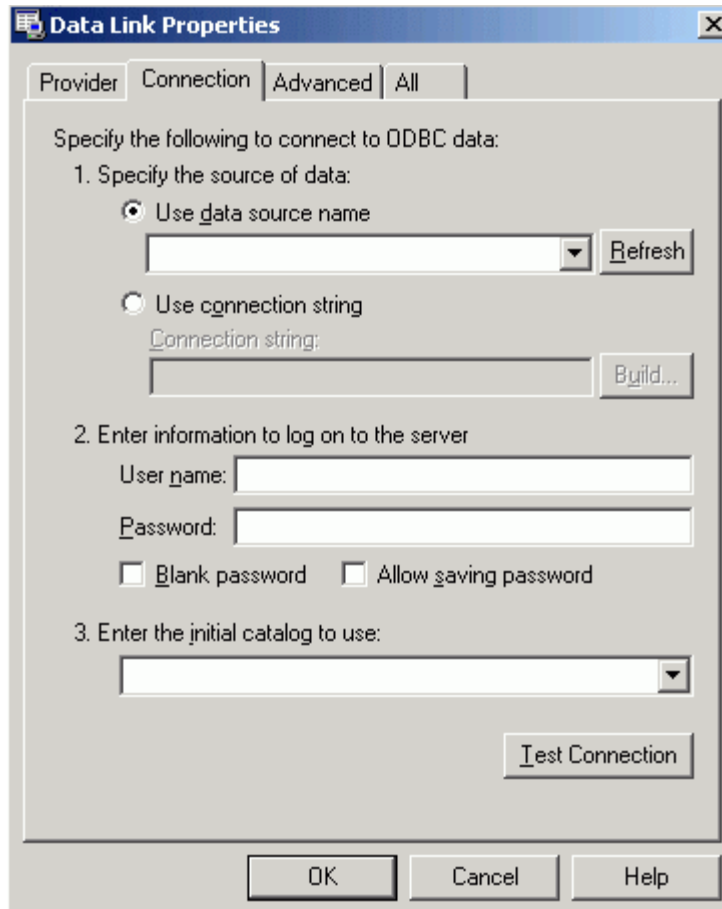
To connect to an Oracle database, fill in the appropriate information, as shown in the figure below.



The image shows a Windows-style dialog box titled "Data Link Properties". It has four tabs: "Provider", "Connection", "Advanced", and "All". The "Connection" tab is selected. Inside the dialog, it says "Specify the following to connect to Oracle data:". Below this, there are two numbered instructions: "1. Enter a server name:" followed by a text box containing "Oracle", and "2. Enter information to log on to the database:" followed by "User name:" and "Password:" labels, each with an empty text box. Below the password box are two checkboxes: "Blank password" and "Allow saving password", both of which are unchecked. At the bottom right of the main area is a "Test Connection" button. At the very bottom of the dialog are three buttons: "OK", "Cancel", and "Help".

OLE DB Provider Configuration for Oracle Databases

To connect to a MySQL database, specify the desired **Data Source Name**, the **Database Name**, **User**, **Password**, etc., as shown in the figure below.



OLE DB Provider Configuration for MySQL Databases

To increase speed and efficiency, the TrendWorX32 OLE DB Provider establishes a connection on a per-data-logging group basis and retrieves data for the historical tags in that group. You can always close an existing connection, set different properties and reopen the connection to retrieve data for a different group of historical tags.

Alternatively, you can set the required properties for opening a connection of the TrendWorX32 OLE DB Provider individually. These properties are:

Data source. For a Microsoft Access historical database, the data source is the complete path (including the file name) of the Microsoft Access .mdb file. For a Microsoft SQL Server historical database, the data source is the server name used for local SQL server installation. If the Microsoft SQL Server is installed locally, the server name will be "(local)."

Password. The required password. If the underlying historical database is a Microsoft SQL Server database, and no password is defined, the ICONICS TrendWorX32 OLE DB Provider will attempt to use NT integrated security.

User ID. The user ID name (if required).

Location. If the underlying historical database is a Microsoft SQL Server database, this property will be used for storing the initial catalog (i.e. the default SQL server database) for the connection to the historical database.

Extended properties. This is a dedicated ICONICS TrendWorX32 OLE DB Provider string that is formulated as follows:

```
TWXPROVIDER=provider;TWXDATABASE=database group;TWXGROUP=group;
```

"Provider" is the internal OLE DB Provider name (i.e. SQL OLE DB for Microsoft SQL Server, or Microsoft.Jet.OLEDB.4.0 for Microsoft Access historical database)

"Database group" is the historical database group.

"Group" is the desired group of historical tags under the database group for which the provider will retrieve data.

Using the properties listed above, you can also connect to the ICONICS TrendWorX32 OLE DB Provider through ADO in the following ways:

```
Dim twx As New ADODB.Connection
```

```
' Set Provider
```

```
twx.Provider = " ICONICS.TWXOLEDB "
```

```
' Set up Properties
```

```
twx.Properties("Prompt") = 0
```

```
twx.Properties("Data Source") = "(local)"
```

```
twx.Properties("Password")=""
```

```
twx.Properties("User ID")=""
```

```
twx.Properties("Extended Properties")=
```

```
"TWXPROVIDER=sqloledb;TWXDATABASE=MyFirstDBase;TWXGROUP=LoopX;"
```

```
twx.Properties("Location")="TWXSQL_TEST_1"
```

```
'Open Connection
```

```
twx.Open
```

This code assumes the following:

- The historical database is a Microsoft SQL Server database.
- A trusted connection is used.
- The SQL Server database where the historical data are logged is called "TWXSQL_TEST_1."
- The Microsoft SQL Server is located on the local workstation.

In addition to the above properties, the ICONICS TrendWorX32 OLE DB Provider supports language aliasing by making its resources available in a

resource .dll. When translating the resources, do *not* translate the strings related to the default properties of the provider.



[Using the TrendWorX32 OLE DB Provider](#)

Retrieving Historical Data

TrendWorX32 Reporting has been updated to be compliant with the latest OPC HDA specification 1.2. Although TrendWorX32 Reporting is not a direct OPC HDA client or server, it creates reports with data outputs as specified by the OPC HDA specification.

Version 7.x introduced an enhanced data-retrieval system, which utilizes an updated approach to creating historical reports:

- Data filter selection other than Raw will result in data time stamped at the beginning of each subinterval.
- When retrieving data using data filters other than Raw, subintervals for which there are no data because of no data-logging activity will be marked as "empty" slots at the corresponding time with a zero value. You can check the returned qualities for further processing.
- The historical qualities supported are as follows:
 - OPCHDA_EXTRADATA 0x0001
 - More data may exist
 - OPCHDA_INTERPOLATED 0x0002
 - Interpolated data value
 - OPCHDA_RAW 0x0004
 - Raw data value
 - OPCHDA_CALCULATED 0x0008
 - Calculated data value
 - OPCHDA_NOBOUND 0x0010
 - No data found to provide upper or lower bound
 - OPCHDA_NODATA 0x0020
 - No data collected. Archiving not active
 - OPCHDA_DATA_LOST 0x0040
 - Calculation started/stopped/lost
 - OPCHDA_CONVERSION 0x0080
 - Scaling / conversion error
 - OPCHDA_PARTIAL 0x0100
 - Aggregate value is for an incomplete interval

Critical Note: Because of the new compliance changes in historical replay and reporting, a version 7.x and subsequent versions historical replay or reporting may not produce identical results to existing 6.1x version reports and or historical replay screen captures for the same signals and time periods.

The TrendWorX32 OLE DB Provider supports a minimal set of "trend SQL" keywords that can be used to formulate "trend" queries in order to retrieve historical data. A typical "trend" SQL statement may look like the following:

**Select=Device1.Group.Tag1,Device1.Group.Tag10;Filter=MaxTime;
SubInterval=10000; Start=1999/6/16, 18:00:00; End=1999/6/16, 18:10:00;**

Select=tag1,tag2,...,tagn

This specifies the desired historical tags to participate in the query. If the '*' is used instead (e.g. Select=*,), without other keywords, the ICONICS TrendWorX32 OLE DB Provider will return information relating to the properties of the historical tags, such as device point name, logging name, and units. Note that in this query the logging names of the historical points should be used.

Filter=filter type;

This specifies the desired historical data-processing filter to be used when retrieving data historical data. The supported filter types are:

- Raw (all values are retrieved)
- Average
- Min
- Max
- StDev
- MinTime (minimum values with actual time stamp)
- MaxTime (maximum values with actual time stamp)
- Last (decimated value)
- All Statistics (maximum, minimum, standard deviation, and average over entire time span)
- Total (summary of values)
- Running Min
- Running Max
- Running Average (exponentially weighted moving average)
- Moving Average
- Moving Min
- Moving Max

If no "Filter" value is specified, the TrendWorX32 OLE DB Provider will retrieve all raw data-logged values.

SubInterval=interval;

This specifies the interval (in msec) between processed samples. The TrendWorX32 OLE DB Provider will process all samples, which have time stamps within a time interval to compute the corresponding sample with the desired filter. It is ignored if the data-retrieval filter is set to "Raw" or "AllStat." If it is set to 0, all historical samples within the search time interval will be included in creating a "processed" sample, which will be returned to the user.

Note that when the data filter is set to "Raw," the ICONICS TrendWorX32 OLE DB Provider will return all historical samples, which fall within the search time interval using the original time stamp that was data-logged. On the other hand, if it is not set to "Raw," the ICONICS TrendWorX32 OLE DB Provider will divide the search time interval into subintervals of "SubInterval" duration and "process" all samples that fall within each subinterval to calculate a "processed" sample to return.

Start=yyyy/mm/dd, hh:mm:ss;

This specifies the starting time for retrieving historical data, and it has to match the indicated format. If it is not specified, the time search will include all historical values starting at the beginning of the historical data

End=yyyy/mm/dd, hh:mm:ss;

This specifies the ending time for retrieving historical data, and it has to match the indicated format. If it is not specified, the time search will include all historical values up to the most recent ones

Important Notes

If any or both of the start and end dates are not specified, the TrendWorX32 OLE DB Provider will perform an exhaustive database search to retrieve all samples in the database, or those samples starting from the start date or ending prior to the end date. Because this can be an extremely time-consuming operation, it is suggested that you use the "MaxRows" property of the provider to establish a limit to the total number of retrieved samples.

Note: The Unicode version has enhanced language aliasing support, including automatic value scaling. You need to configure the ICONICS Language Server, as well as configure language aliases in the TrendWorX32 Configurator. Once all language aliases are configured, you can use the LCID property of the TrendWorX32 OLE DB Provider connection to set the desired language settings.



[Using the TrendWorX32 OLE DB Provider](#)

Returned Recordset Structure

The TrendWorX32 OLE DB Provider returns the following recordset structure (column names):

TagLoggingName

A variable length text column displaying the logging name of the tag.

TagPointName

A variable length text column displaying the actual point name of the tag.

TimeDate	A "date" type showing the time stamp.
Msecs	A "long" type indicating the msec accuracy of the time stamp.
Value	A "double" type indicating the value of the sample.
OPCDataQuality	A "long" type indicating the OPC data access quality of the sample.
OPCHDAQuality	A "long" type indicating the OPC data access quality of the sample.
QualityText	A variable-length "text" type indicating the quality string related to the combined data and historical quality values.
DataFilter	A variable-length "text" type indicating the data filter type, such as "Raw," used in retrieving the data.
EngUnits	A variable-length "text" type indicating the "EngUnits" of the tag.

Important Note

In the case that only the tags under the desired logging group are returned (that is, when Select=*) the "DataFilter" column will contain the data-logging filter for the tag. See the TrendWorX32 SQL Server documentation for more information.

A code sample for opening a recordset to return all tags data-logged in a particular group and to display their logging name might be as follows:

```
Dim rstwx As New ADODB.Recordset
    Dim strLoggingName As String

    ' Set Recordset properties
    rstwx.CursorType = adOpenForwardOnly
    rstwx.LockType = adLockReadOnly
    ' Set cache size
    rstwx.CacheSize = 100
    ' Open Recordset
    rstwx.Open "Select=*;", twx, , , adCmdText

    ' Process Records
    Do While Not rstwx.EOF
        strLoggingName = rstwx!TagLoggingName
        rstwx.MoveNext
    Loop
```

```

' Close Recordset
If rstwx.State = adStateOpen Then
rstwx.Close
End If

End Sub

```

In this case, "twx" is the ICONICS TrendWorX32 OLE DB Provider ADO connection object we just opened.

 [Using the TrendWorX32 OLE DB Provider](#)

Example Application Using TrendWorX32 OLE DB Provider

In order to provide a sample application using the TrendWorX32 OLE DB Provider, has created a sample VBA application to demonstrate its use. It is located under the GENESIS32 "Examples" directory in the "TrendWorX32 Examples\TrendWorX Data Logging and Reporting" folder, and it is called "TrendOleDb.t32." This file consists of a main form, which is shown below. Click **Connect** to connect to a TrendWorX32 SQL Server historical database. Click **Signal Info** to retrieve the data-logged signal for a particular group. You can also build a trend query and execute it to retrieve historical data.

To prepare for the example, you must:

- Set the GENESIS32 working directory to be "TrendWorX32 Examples\TrendWorX Data Logging and Reporting," where the examples are installed. (For example you can use the **Set Working Directory** command in the **Tools** menu of the TrendWorX32 Container.
- Make sure that none of the files in this folder has a read-only attribute.

Once you have completed these steps, do the following:

1. Start the TrendWorX32 Container.
2. Load the TrendOleDb.t32 display into TrendWorX32.
3. Enter animation mode.
4. Click the various buttons to retrieve TrendWorX32 historical data using the TrendWorX32 OLE DB Provider.
5. When prompted, you can select Microsoft Access as the historical database type and enter "Twx32Demo.mdb."
6. You can now start data logging, stop data logging, and retrieve historical data.

TrendOleDb.t32

TrendWorX32 OLEDB Provider Example

Connect with Prompt

Connect to TWX32Demo mdb

TWX Connection String:

Signal Info

Build Query

Samples

Get Data

Sample Visual Basic Application for the TrendWorX32
OLE DB Provider

 [Using the TrendWorX32 OLE DB Provider](#)