

**TrendWorX32 SQL Query
Engine V9.2
Beta III
Documentation**
(Preliminary – November 2009)



OPC Automation at your fingertips

1. Introduction

TrendWorX32 Logger logs data to a database. You can use the TrendWorX32 Viewer to see and edit your logged data in a graphical format. TrendWorX32 SQL Query Engine allows you to use common SQL Data Manipulation Language (DML) queries to retrieve and edit data. These data can be displayed in any SQL clients (e.g. MS Excel, MS SQL Server or ICONICS ReportWorX).

2. General Requirements

GENESIS32 version 9.2 or later, VC Redistributable files for VS2008, some SQL client(s).

3. Provider capabilities and limitations

Testing and validating

The TrendWorX32 SQL Query Engine is developed, tested and validated against the Microsoft OLE DB RowsetViewer (version: 2.70.7531.0).

SQL capabilities

Due to the provider nature limited range of SQL features is supported. Catalogs, schemata and tables are predefined (by TrendWorX32 Logger). You can use only DML (Data Manipulation Language) queries, i.e. SELECT, INSERT, UPDATE and DELETE. Also the stored procedures are implemented in the TrendWorX32 SQL Query Engine.

Individual query types have specific limitations to the query structure too. These limitations are detailed in appropriate query section. In addition, advanced consumers (e.g. MS SQL Server) may perform query analysis (or preprocessing) and may change original query. This may lead to unexpected results. These rarities will be described later.

4. Architecture

TrendWorX32 SQL Query Engine was developed as in-proc COM server. Whole provider is divided to the three DLL libraries (modules). You can use the provider in the 32bit and also in the 64bit clients.

32bit binaries (needed every time)

- GenericOleDbProvider.dll,
- HistoryAggregates.dll,
- RawDataProviderHost.exe,
- RawDataProviderHostPS.dll,
- Twx32OleDbProvider.dll,
- Twx32RawDataProvider.dll.

64bit binaries (needed for the 64bit clients)

- GenericOleDbProvider_x64.dll,
- HistoryAggregates_x64.dll,

- RawDataProviderHostPS_x64.dll,
- Twx32OleDbProvider_x64.dll.

GenericOleDbProvider.dll, GenericOleDbProvider_x64.dll

This module communicates with a SQL client. Module processes data from the Twx32OleDbProvider.dll and provides them in the form of tables and stored procedures to the client.

HistoryAggregates.dll, HistoryAggregates_x64.dll

This module calculates history aggregates from raw data which are gained from the TrendWorX32 HDA Server.

Twx32OleDbProvider.dll, Twx32OleDbProvider_x64.dll

This module communicates with Twx32RawDataprovider.dll. The communication is done through the RawDataProviderHost.exe application.

RawDataProviderHost.exe

This is a host for the Twx32RawDataProvider.dll. If you want to use TrendWorX32 SQL Query Engine in the 64bit client you need the 64bit version of the provider. But the Twx32RawDataProvider.dll must be 32bit every time because this dll communicates with another one which is also 32bit. Because of this the RawDataProviderHost.exe application was developed.

RawDataProviderHostPS.dll, RawDataProviderHostPS_x64.dll

These files are only the proxy/stub for the RawDataProviderHost.exe application.

Twx32RawDataProvider.dll

This module can get data from the TrendWorX32 HDA Server. Twx32RawDataProvider.dll communicates with the GenClient.dll (part of the GENESIS32 installation). If you want to get HDA data from other than your computer the GenClient communicates with the GenBroker in this case.

5. Data architecture

QuerySettings table

This table contains information which is needed during the SQL query calculation. The table has following columns:

Column name	Data type	Description
TIMESTAMP_START	DBTIMESTAMP	Start of the timestamp range (default value – can be overloaded in the query).
TIMESTAMP_END	DBTIMESTAMP	End of the timestamp range (default value – can be overloaded in the query).
MAX_ROWS_TO_RETURN	UI4	Number of rows which can be returned to the client.
TREAT_UNCERTAIN_AS_GOOD	BOOL	For the HDA calculations.
PERCENT_DATA_BAD	BYTE	For the HDA calculations.
PERCENT_DATA_GOOD	BYTE	For the HDA calculations.

ALLOW_EMPTY_TAG	BOOL	If this value is set to true you can run queries against the RawData table without the defining the tagname in WHERE clause. Otherwise these queries fail.
-----------------	------	--

Table 1 - QuerySettings table columns

Tags table

This table contains information about tags. Table presents list of all tags which are available in the TrendWorX32 HDA Server. The table has following columns:

Column name	Data type	Description
TAGNAME	WSTR BYREF	Name of the tag.
DESCRIPTION	WSTR BYREF	Tag description.
DATATYPE	UI2	Information about tag's data type.
STEPPEDINTERPOLATION	BOOL	Stepped interpolation info.
ACCESSRIGHTS	UI2	Information about access rights.

Table 2 - Tags table columns

RawData table

This table contains information about all raw data which are in the TrendWorX32 HDA Server (you can specify catalog – DBGroup and schema – LoggingGroup). The table has following columns:

Column name	Data type	Description
TAGNAME	WSTR BYREF	Name of the tag.
TIMESTAMP	DBTIMESTAMP	Timestamp where the value was caught.
QUALITY	UI4	Quality of the value.
VALUE	VARIANT	Value it-self.

Table 3 - RawData table columns

6. Supported SQL constructs

SELECT query

Following SELECT syntax is supported:

SELECT [lead count restriction] expression FROM table | stored procedure [WHERE condition];

Following rules apply to individual select elements:

- *Expression* – defines column(s) to be retrieved by the SELECT query. These may include SQL aggregate column(s) too. But it is not possible to combine ordinal and aggregate columns together in one SQL query (we do not support GROUP BY).
- *Table | stored procedure* – defines table or stored procedure to select data from. It is not possible to combine table with stored procedure in one SQL query.
- *WHERE condition* – defines logical expression that needs to evaluate to true in order to return given table row. The expression only allows AND/OR chaining, usage of brackets, and logical comparison operators (<=, <, >, >=, <>, =).
- *Count restriction* – can be used to restrict number of retrieved rows. Lead restriction has form *TOP x* where x is number of rows to return. Note that count restrictions are applied after the query is evaluated thus they unfortunately won't affect performance in positive.

Additional information

Be very careful when you are running queries against the RawData table. If the ALLOW_EMPTY_TAG in the QuerySettings table is set to true you can see following. You can run query without the WHERE clause but if you do so the OLE DB Provider will get all data for all tags in the specified catalog and schema. Then the provider must sort data from all tags to one stream and then these data will send to the output. This can take very long time in dependence on the tag count. You can set ALLOW_EMPTY_TAG to false. In this case the queries without tagname in WHERE clause will fail (such queries are not permitted).

UPDATE query

Following UPDATE syntax is supported:

UPDATE table SET column_name_1 = column_value_1 [, column_name_2 = column_value_2, ...] [WHERE condition];

UPDATE works as standard SQL UPDATE – it expects table name and columns to be modified and values for these columns. It is also desired to narrow affected rows by WHERE clause.

Update works only against the QuerySettings table. TrendWorX32 API doesn't support inserts, updates and deletes so the functionality is not implemented in the OLE DB Provider.

SQL aggregates

Following standard SQL aggregates are supported in SELECT column definition expression:

- MIN,
- MAX,
- AVG,
- SUM,
- COUNT.

All of these functions require column name as parameter, e.g. *SELECT SUM(Value) FROM ...* .

Note: When you are using SQL aggregates it is not possible to get other common columns, GROUP BY clause is not supported.

SQL functions

Follows list of SQL functions:

QualityToStr

This function can be used instead of the Quality column. Function converts quality from DWORD representation (number) to the string, e.g. *SELECT TagName, QualityToStr(Quality) FROM RawData ...* .

Output can be one of the following values:

- Good,
- Bad,
- Uncertain.

DataTypeToStr

This function can be used instead of the DataType column. Function converts data type from the WORD representation (number) to the string, e.g. *SELECT TagName, DataTypeToStr(DataType) FROM Tags ...* .

Output can be one of the following values:

- VT_BOOL,
- VT_UI2,
- VT_UI4,
- VT_R8.

AccessRightsToStr

This function can be used instead of the AccessRights column. Function converts access rights from WORD representation (number) to the string, e.g. *SELECT TagName, AccessRightsToStr(AccessRights) FROM Tags ...* .

Output can be one of the following values:

- TAG_ACCESS_RIGHTS_NONE,
- TAG_ACCESS_RIGHTS_READ,
- TAG_ACCESS_RIGHTS_WRITE,
- TAG_ACCESS_RIGHTS_READ_WRITE.

IsGood, IsBad and IsUncertain

These functions can be used in WHERE clause. IsGood function select only raw data where quality is good, likewise other functions, e.g. *SELECT * FROM RawData WHERE IsBad(Quality) = 1;*

Date / Time functions

In the WHERE clause you can use date / time functions. You can use *dateadd()* function and you can use also *now* and *nowutc* variables. Example SQL query: *SELECT * FROM RawData WHERE timestamp >= dateadd(hh, -2, nowutc) and timestamp <= nowutc;* This query returns all raw data which are maximally 2 hours old.

Function syntax is *dateadd(DATE_PART, VALUE, BASE_TIMESTAMP)*, parameters can have following values:

- DATE_PART
 - ms – milliseconds,
 - ss – seconds,
 - mi – minutes,
 - hh – hours,
 - dd – days,
 - wk – weeks,
 - mm – months,
 - yy – year.
- VALUE – value can be any integer.
- BASE_TIMESTAMP
 - now – local time,
 - nowutc – UTC time,

- or you can type your own timestamp in following format 'yyyy-mm-dd hh:mi:ss'.

SQL stored procedures

All historical data aggregates calculations are provided through the stored procedures. In addition more complex operations above raw data are also provided through the stored procedures too.

Follows list of all stored procedures (supported at this time):

- *HDA_BOOL* – calculates Duration in State 0, Duration in State 1, Number of Transitions and Count for one tag.
- *HDA_BOOL_5* – same functionality like *HDA_BOOL* but up to five tags.
- *HDA_ANALOG* – calculates Min, Max, Average, Time Average, Totalize Average, Interpolative, Last, Delta, Range, Total and Count for one tag.
- *HDA_ANALOG_5* – same functionality like *HDA_ANALOG* but up to five tags.
- *HDA_REDUCED_ANALOG* – calculates Min, Max, Time Average and Count for one tag.
- *HDA_REDUCED_ANALOG_5* – same functionality like *HDA_REDUCED_ANALOG* but up to five tags.
- *SP_RAW_DATA* – gets raw data for one tag.

HDA_BOOL, HDA_ANALOG, HDA_REDUCED_ANALOG

Follows list of input parameters for these functions:

- *Start timestamp* – start of the timestamp range,
- *End timestamp* – end of the timestamp range,
- *Resampling period* – time interval size,
- *Tag name* – when you specify only tag name (without catalog and schema) then the catalog and schema is taken for the tag is taken from the stored procedure name.

HDA_BOOL_5, HDA_ANALOG_5, HDA_REDUCED_ANALOG_5

Follows list of input parameters for these functions:

- *Start timestamp* – start of the timestamp range,
- *End timestamp* – end of the timestamp range,
- *Resampling period* – time interval size,
- *Tag name 1*,
- *Tag name 2*,
- *Tag name 3*,
- *Tag name 4*,
- *Tag name 5*.

Note: when you specify only tag name (without catalog and schema) then the catalog and schema is taken for the tag is taken from the stored procedure name.

SP_RAWDATA

Follows list of input parameters for this function:

- *Start timestamp* – start of the timestamp range,
- *End timestamp* – end of the timestamp range,

- *Tag name* – when you specify only tag name (without catalog and schema) then the catalog and schema is taken for the tag is taken from the stored procedure name.

7. Historical aggregates

Aggregate name	Behavior description
Min	Retrieve the minimum value in the resample interval.
Max	Retrieve the maximum value in the resample interval.
Average	Retrieve the average data over the resample interval.
Time average	Retrieve the time weighted average data over the resample interval.
Totalize average	Retrieve the totalized value (time integral) of the data over the resample interval.
Interpolative	
Last	Retrieve the value at the end of the resample interval. The timestamp is the timestamp of the end of the interval.
Duration in state 0	Retrieve the duration of time in the interval during which the data is bad.
Duration in state 1	Retrieve the duration of time in the interval during which the data is good.
Number of transitions	Retrieve the number of state changes a Boolean value experienced in the interval.
Delta	Retrieve the difference between the first and last value in the resample interval.
Range	Retrieve the difference between the minimum and maximum value over the resample interval.
Total	Retrieve the sum of the data over the resample interval.
Count	Retrieve the number of raw values over the resample interval.

Status code calculation

For aggregates values, the status code for each returned aggregate shall be good, if the status code for all values used in the aggregate was good.

If the status code of any value used in computing the aggregate was not good, then the server must use the *TreatUncertainAsBad*, *PercentDataBad* and *PercentDataGood* parameter settings to determine the status code of the resulting aggregate for the interval. Some aggregates may explicitly define their own method of determining quality.

If the percentage of good values in an interval is greater than or equal to the *PercentDataGood*, the aggregate is considered good, otherwise bad.

Since a value can be either good or bad only (uncertain is defined as good or bad as per treat uncertain as bad setting), $percentage\ good = 100 - percentage\ bad$. If a $percentage\ good(X)$ is in the following range $percentage\ bad < X < percentage\ good$ the quality of the aggregate is *Uncertain_DataSubNormal*.

8. MS SQL Server specifics

MS SQL Server performs specific modifications to queries passed to linked server (e.g. aliasing). It also performs many auxiliary operations such as data type conversions and local functions.

MS SQL Server preprocessing puts certain limitations on query parameters. Namely, the MS SQL Server won't propagate time specified as absolute date, e.g. '2009-01-01 12:00:00', to the provider. It is necessary to use date manipulation functions, i.e. *dateadd()*, *getutcdate()* or "hack" the conversion using nested SELECT statement, e.g. *select convert(datetime, '2009-01-01 12:00:00')*. Or it is possible to use OPENQUERY statement, e.g. *select * from openquery(Twx32, '<SQL query>')*. This last solution is recommended.

MS SQL Server query execution problems

When MS SQL Server decides not to send entire query to the provider, the query can be still executed and can succeed. But rather than being processed by the provider it will be processed at MS SQL Server.

The problems with this approach are:

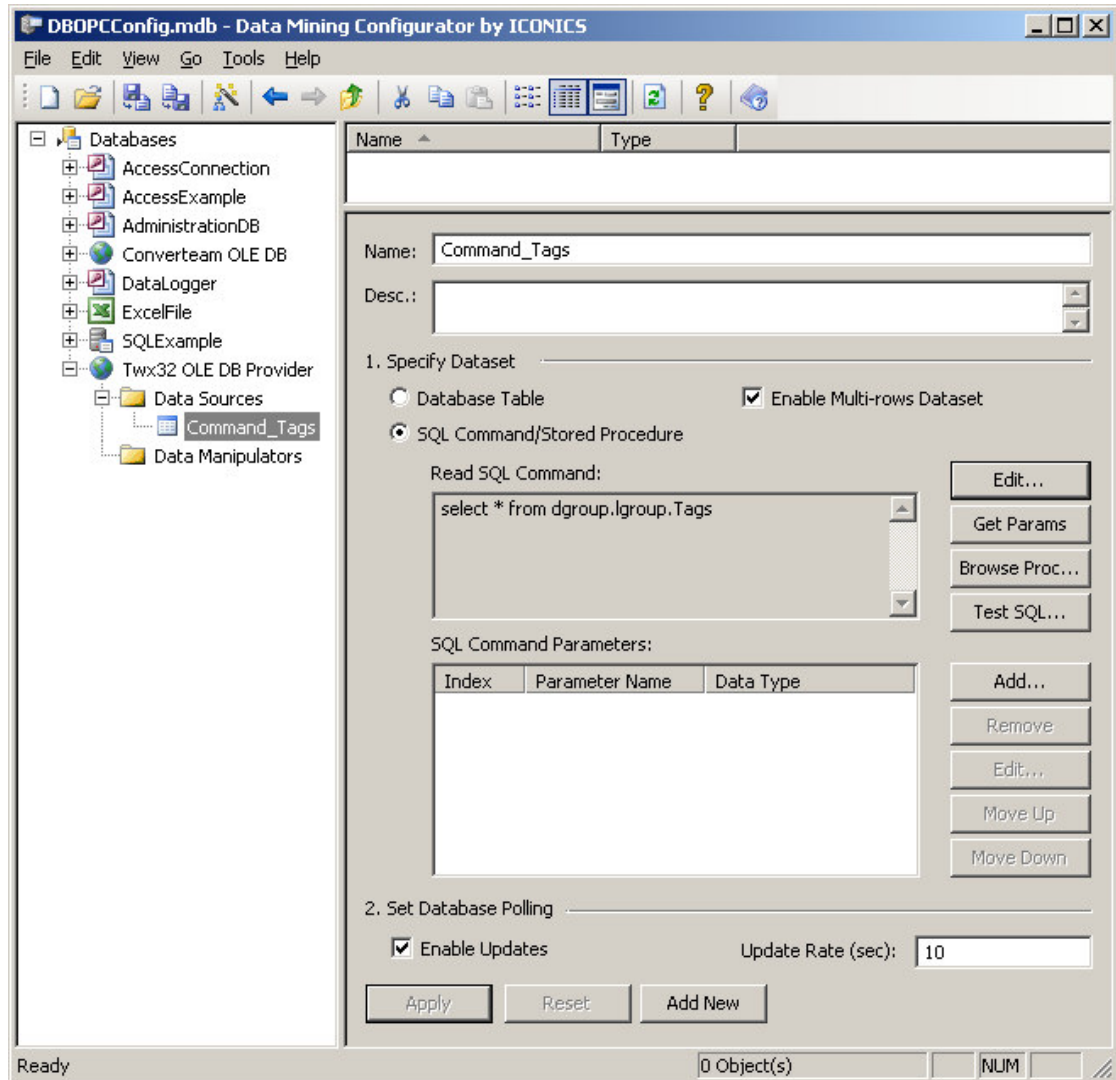
- *Performance* – if query, e.g. *SELECT * FROM Twx32.<catalog>.<schema>.RawData WHERE TIMESTAMP >= '2009-01-01 12:00:00' and TAGNAME = 'SimulatePLC.Sine';* is used, the MS SQL Server will execute *SELECT * FROM Twx32.<catalog>.<schema>.RawData;* against the provider. The provider won't get information about the tag for which you can read data. In this case the provider will get all data for all tags in the specified catalog and schema (this can take very long time) and after this the MS SQL Server will make the final filtering. If you want to forbid this set the ALLOW_EMPTY_TAG in the QuerySettings to false. Then these queries will not be started.
- *Runtime limitations* – certain queries are prohibited, e.g. queries without certain limit parameters. MS SQL Server might inadvertently use these queries when executing subset of user-supplied query which would lead to query execution failure.

9. Configuring TrendWorX32 SQL Query Engine

ICONICS Data Mining Configurator

1. Open Data Mining Configurator by going to Start → Programs → ICONICS Tools → Data Mining Configurator.
2. On the left double-click Databases to expand it.
3. Right click on the Databases. Select New → Database → Connection.
4. Fill in Connection Name, e.g. TrendWorX32 SQL Query Engine.
5. Set Connection Type to Generic OLE DB data source and click Advanced... button.
6. Select first tab (Providers) and in the list select the TrendWorX32 SQL Query Engine. Click Next >> button.
7. Fill in Data Source. This is a name of the computer from which you can get HDA data.
8. Select initial catalog (this can take a while).
9. You can test a connection.
10. Click OK button. Then click Apply button.

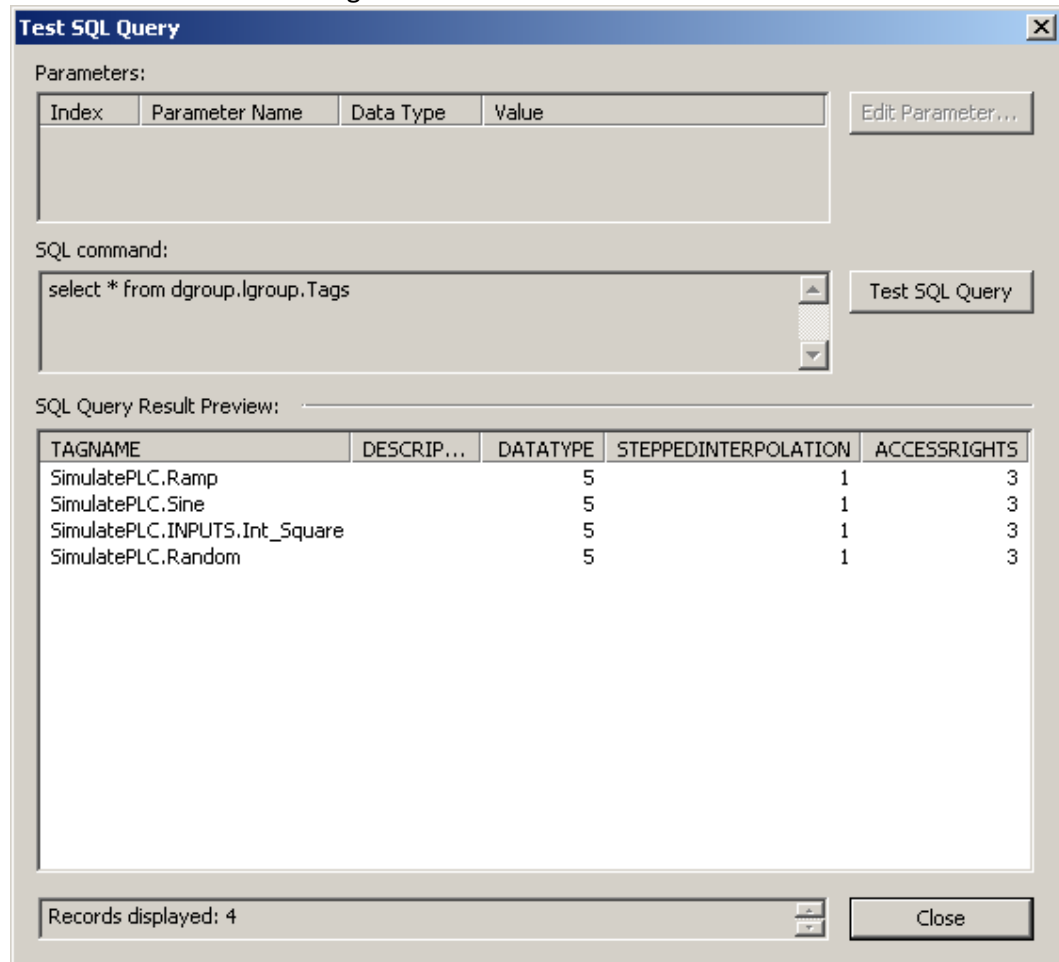
11. Double-click TrendWorX32 SQL Query Engine to expand it.
12. Right-click on the Data Sources.
13. Select New → Data Sources.
14. Fill in name, e.g. Command_Tags.
15. In the Specify Dataset set SQL Command/Stored procedure.
16. Next check Enable Multi-rows Dataset.



Picture 1 - Data Mining Configurator

17. Click Edit... button.
18. Write SQL query, e.g. select * from cat1.schem1.Tags.
19. Click Apply button.
20. You can test the query by clicking the Test SQL... button.
21. In new window click Test SQL Query button.

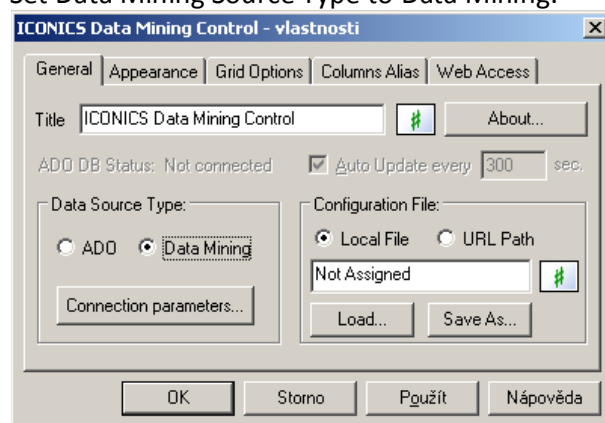
22. You should see the list of tags.



Picture 2 - Data Mining Configurator - Test SQL Query

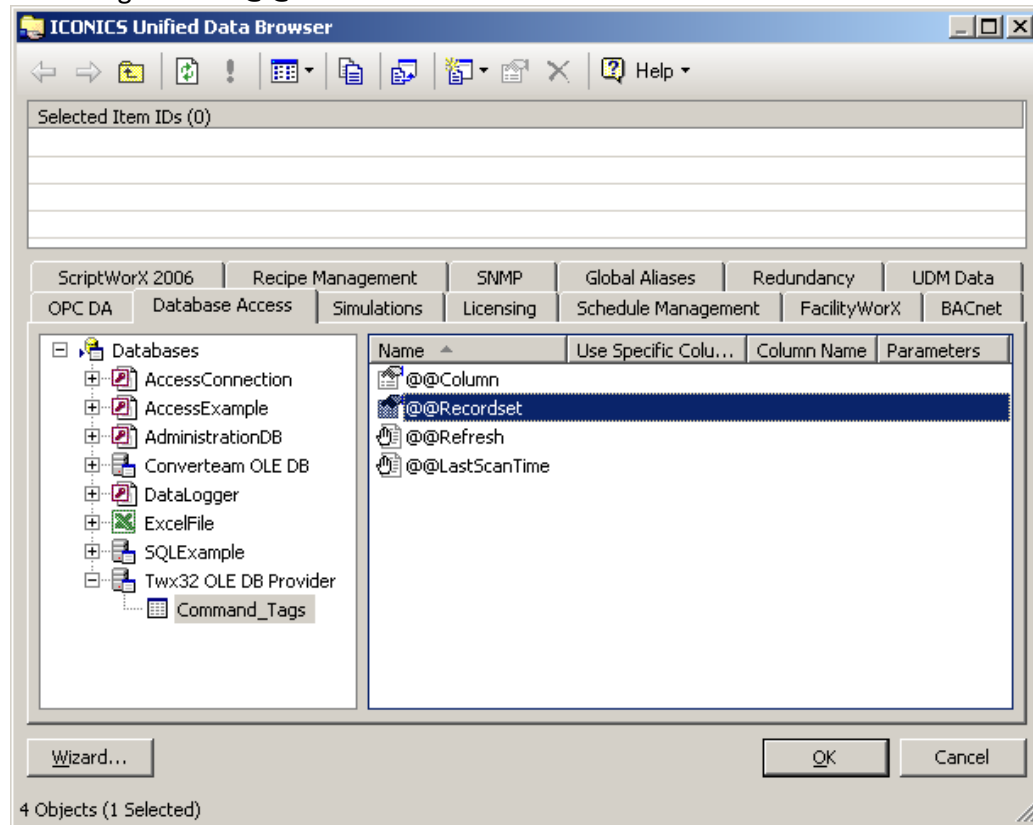
Getting data in the GraphWorX

1. Start GraphWorX by going Start → Programs → ICONICS GENESIS32 → GraphWorX32 → GraphWorX32.
2. In the ActiveX toolbar click on the ICONICS Data Mining Control.
3. Double-click on the ActiveX in the display.
4. Set Data Mining Source Type to Data Mining.



Picture 3 - Data Mining Control

5. Click Connection parameters... button.
6. Click Add Tags... button.
7. Select Database Access tab.
8. Double-click Databases to expand it.
9. Double-click TrendWorX32 SQL Query Engine to expand it.
10. Select Command_Tags.
11. On the right select @@Recordset and then click OK button.



Picture 4 - ICONICS Unified Data Browser

12. Confirm all changes.
13. Go to runtime.
14. You should see a list of tags (may take a while).

TrendWorX32 SQL Query Engine as a Linked Server

TrendWorX32 SQL Query Engine can be used in the MS SQL Server as a linked server.

1. Open MS SQL Server Management Studio by going to Start → Programs → Microsoft SQL Server 2008 → SQL Server Management Studio.
2. When asked, fill in credentials to connect to your local SQL Server.
3. Click New Query button.
4. Use script (Attachment 1) and copy it to the query window.
5. Fill in the data source in the script, part *Create Linked Server*, parameter *@datasrc=N''*. Data source is a name of the computer from which you can get HDA data.
6. Run the script.
7. In the Object Explorer on the left expand Server Objects.

8. Double-click Linked Servers to expand it. You should now see a linked server called Twx32.
9. You can now run SQL queries.

10. Troubleshooting

Checking if the TrendWorX32 SQL Query Engine is correctly installed in the SQL Server

- Start Microsoft SQL Server Management Studio.
- Open window for new query.
- Run following query: *exec sp_catalogs <name of the linked server>*.
- If the provider is installed you should get a list of all catalogs (TrendWorX32 database groups).

Cannot get any data in the SQL Server

- Run dcomcnfg application.
- Select Component Services → Computers → My Computer → DCOM Config.
- In the application list find the ICONCIS Raw Data Provider Host.
- Open its properties and go to the Security tab.
- Check if there is a user under which the SQL Server is running (usually NETWORK SERVICE). This user must exist in the Launch and Activation Permissions. Also this user must exist in the Access Permissions.
- If the user exists in both cases check if there is all allowed for this user.
- Confirm all changes.
- Open Windows Task Manager and go to the Processes tab.

Find the RawDataProviderHost.exe process and end the process (or you can restart the computer).

11. Examples

Selecting all tags:

*SELECT * FROM cat1.schem1.Tags;* - SQL client

*SELECT * OPENQUERY(Twx32, 'SELECT * FROM cat1.schem1.Tags;');* - MS SQL Server

Selecting raw data:

SELECT TagName, Timestamp, QualityToStr(Quality), Value FROM cat1.schem1.RawData WHERE TagName = 'SimulatePLC.Sine' and TIMESTAMP >= '2009-01-01 12:00:00' and TIMESTAMP <= '2009-01-01 13:00:00';

Selecting query settings:

*SELECT * FROM cat1.schem1.QuerySettings;*

Running HDA_BOOL stored procedure:

EXEC cat1.schem1.HDA_BOOL '2009-01-01 12:00:00', '2009-01-01 13:00:00', 300000, 'SimulatePLC.Sine';

Running HDA_ANALOG stored procedure in SELECT query:

```
SELECT * FROM cat1.schem1.HDA_ANALOG('2009-01-01 12:00:00', '2009-01-01 13:00:00',  
300000, 'SimulatePLC.Sine');
```

Selecting MAX, MIN and AVG value:

```
SELECT MIN(Value) as min_value, MAX(Value) as max_value, AVG(Value) as avg_value FROM  
cat1.schem1.RawData WHERE TagName = 'SimulatePLC.Sine' and TIMESTAMP >= '2009-01-  
01 12:00:00' and TIMESTAMP <= '2009-01-01 13:00:00';
```

12. Attachments

Attachment 1 – Script for Linked Server

```
/*  
*****  
***** Twx32 - Create Linked Server *****  
*****  
*/  
  
/*!!!  
Update @datasrc property in the script - data source is computer name from which you  
can get HDA data.  
!!!*/  
  
/*  
Set OLE DB provider parameters  
*/  
EXEC master.dbo.sp_MSset_oledb_prop N'ICONICS.Twx32OleDbProvider', N'DynamicParameters', 1  
EXEC master.dbo.sp_MSset_oledb_prop N'ICONICS.Twx32OleDbProvider', N'NestedQueries', 1  
EXEC master.dbo.sp_MSset_oledb_prop N'ICONICS.Twx32OleDbProvider', N'AllowInProcess', 1  
EXEC master.dbo.sp_MSset_oledb_prop N'ICONICS.Twx32OleDbProvider', N'NonTransactedUpdates', 1  
GO  
/*  
Crate Linked Server  
*/  
EXEC master.dbo.sp_addlinkedserver @server = N'Twx32', @srvproduct=N'Twx32 OLE DB provider',  
@provider=N'ICONICS.Twx32OleDbProvider', @datasrc=N''  
EXEC master.dbo.sp_addlinkedsrvlogin  
@rmtsrvname=N'Twx32', @useself=N'False', @locallogin=NULL, @rmtuser=NULL, @rmtpassword=NULL  
  
/*  
Set Linked server parameters  
*/  
GO  
EXEC master.dbo.sp_serveroption @server=N'Twx32', @optname=N'collation compatible',  
@optvalue=N'true'  
GO  
EXEC master.dbo.sp_serveroption @server=N'Twx32', @optname=N'data access', @optvalue=N'true'  
GO  
EXEC master.dbo.sp_serveroption @server=N'Twx32', @optname=N'dist', @optvalue=N'false'  
GO  
EXEC master.dbo.sp_serveroption @server=N'Twx32', @optname=N'pub', @optvalue=N'false'  
GO  
EXEC master.dbo.sp_serveroption @server=N'Twx32', @optname=N'rpc', @optvalue=N'true'  
GO  
EXEC master.dbo.sp_serveroption @server=N'Twx32', @optname=N'rpc out', @optvalue=N'true'  
GO  
EXEC master.dbo.sp_serveroption @server=N'Twx32', @optname=N'sub', @optvalue=N'false'  
GO  
EXEC master.dbo.sp_serveroption @server=N'Twx32', @optname=N'connect timeout', @optvalue=N'0'  
GO  
EXEC master.dbo.sp_serveroption @server=N'Twx32', @optname=N'collation name', @optvalue=null  
GO  
EXEC master.dbo.sp_serveroption @server=N'Twx32', @optname=N'lazy schema validation',  
@optvalue=N'false'  
GO  
EXEC master.dbo.sp_serveroption @server=N'Twx32', @optname=N'query timeout', @optvalue=N'0'  
GO  
EXEC master.dbo.sp_serveroption @server=N'Twx32', @optname=N'use remote collation',  
@optvalue=N'false'
```

13. References

[1] – OPC Unified Architecture Specification; Part 13: Aggregates; Release Candidate; version 1.00; July 15, 2008