GridWorX Viewer – Displaying HyperHistorian Data

APPLICATION NOTE

July 2012

Description: Displaying Hyper Historian Data in GridWorX Viewer
OS Requirement: Windows Server 2003 x64/Vista x64/ Server 2008 x64/Windows 7 x64/ Server 2008 R2 x64
General Requirement: GENESIS64 + Hyper Historian v10.7x (or higher) installation, basic knowledge of Hyper Historian and GENESIS64.

Introduction

GridWorX64 Viewer was developed in order to visualize database data or OPC DA data through a data grid type control. GridWorX64 can also read the plant data stored in Hyper Historian database via OLE DB provider.

This application note provides an example of how to read Hyper Historian data in GridWorX Viewer. This application note also assumes that you have followed though the example in the Application Note titled *GENESIS64 – GridWorX64 Server, GENESIS64 – GridWorX64 Viewer.*

Before you start

1. Before you start to read the data from HH database make sure that Hyper Historian is properly setup and you are logging the data into database. If you never configured Hyper Historian, you can start with app note: *Hyper Historian - Quick Start.*
2. Double check that Hyper Historian Logger Engine is started (via traffic light, service manager). Please notice that HH Engine has to be running to read the data from HH database via SQL Query Engine. For more details how to use SQL Query Engine see related app note *Hyper Historian - SQL Query Engine Quick Start*.
3. As you know how to get the data in SQL server via standard OPEN QUERY you can load the store procedure below to SQL server and make sure that is accurate to your configuration. You can test it in SQL Management Studio. For more details how to work with store procedures in SQL Server see SQL Server documentation or related app note(s).

Store procedure below needs to be saved to SQL server (in Hyper Historian DB) as we are going to call it from GridWorX64 Server later on:

```
/****** Object:  StoredProcedure
[dbo].[sp_GetRawData_perTag_perTimePeriod] Script
Date: 10/26/2010 ******/
/****** Copyright ICONICS Inc. *****/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE
[dbo].[sp_GetRawData_perTag_perTimePeriod]
        -- Add the parameters for the stored procedure
here
        @StartTime varchar(50),
        @EndTime varchar(50),
        @LoggingGroup varchar(100),
        @CollectingGroup varchar(100),
        @TagName varchar(1000)
AS
Declare @OpnQ varchar (1000)
Declare @SQL nvarchar(3000)

Select @OpnQ = 'SELECT [TAGNAME]
                                ,[TIMESTAMP]
                                ,[QUALITY]
                                ,[VALUE]
                                FROM
[HH2].['+@LoggingGroup+'].['+@CollectingGroup+'].[RAW
DATA]
                                WHERE
TAGNAME='''''+@TagName+''''' AND TimeStamp >=
'''''+@StartTime+''''' AND Timestamp <='''''
+@EndTime+''''''

Select @SQL = 'SELECT * FROM OPENQUERY (HH2,''' +
@OpnQ+''')'

exec sp_executesql @SQL

GO
```

Connecting to Hyper Historian database

1. Open up workbench by going to **Start → All Programs → ICONICS → Workbench**
2. In Project Explorer, choose **GridWorX64 Server** provider from the list at the bottom
3. Under GridWorX64 configuration database, right click on **SQL Server Connections** and click on **New Database Connection.**
4. Provide a name for the connection such as "Hyper Historian" and click on **Change Connection**.
5. Double click on SQL Server Connections and click on  button to open up Connection String dialog box. Enter the SQL server info and then select "Hyper Historian" database.
6. Click OK to save the changes.
7. Back on the Connection Strings, click on name of connection you created and click OK.
8. Click on "Test connection" button to check if you can connect to database.

NOTE: You can also use OLEDB Provider with this connection string:
`Provider=ICONICS.HHOleDbProvider.1;Password=""`

Configuring GridWorX64 Server

1. Right-click on the database connection you created and choose **New Data Source**.
2. Name this data source **GetRawData.**

3. In SELECT command tab, click on the pencil icon and choose **Store Procedure Call.**
4. Browse for store procedure which you have stored on SQL Server. If you skipped this step at the beginning go back to section "Before you start" (Step 3).

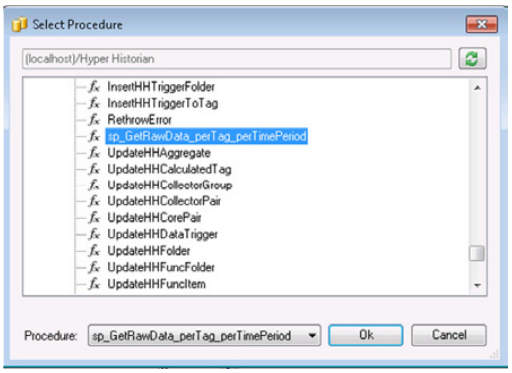

Figure 1 – Selecting Store Procedure

5. Ones you are done you can run test query with some standard parameters by clicking on "Test SELECT Command" icon. If you call store procedure for default Hyper Historian configuration you should see dialog similar to figure below. Please notice that input parameters vary on database structure, so, you need to insert it logically.

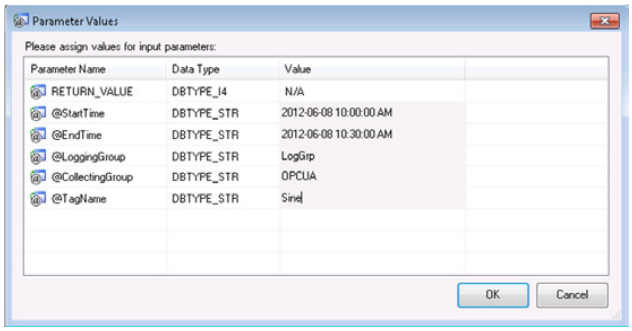

Figure 2 – Test Query with input parameters

6. After running the query you will see output similar to this:

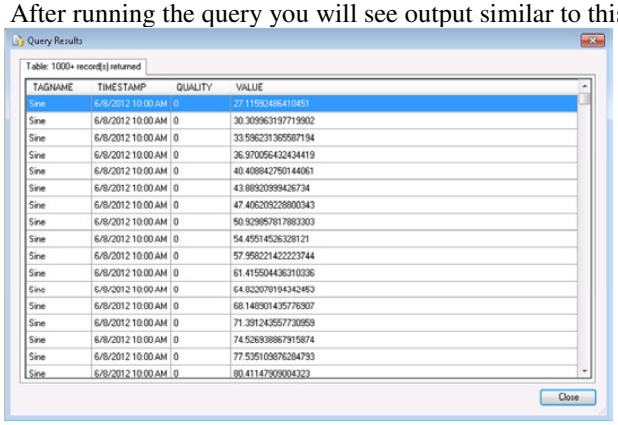

Figure 3 – Test Query with input parameters

NOTE: Please notice that you are working with raw data, so, you should be careful about entering long time period as it can overload the Server.

Reading data in GridWorX64 Viewer

1. Open up Workbench by going to **Start → All Programs → ICONICS GENESIS64 → Workbench.**
2. In Project Explorer, choose **GraphWorX64** Provider from the list of providers.
3. Select Controls tab in the ribbon bar and click on GridWorX64 Viewer.
4. Click anywhere on the empty display to drop the control onto it.
5. Double click on the GridWorX64 Viewer control to bring up the configuration dialog box.
6. Click on **Grid** from the tree on the left.
7. Under **Source** tab, click on  to create a new source. Choose **Data Set Tag Subscription**. Give it a name Hyper Historian.
8. Click on  under Select Data tag to open up OPC UA Browser.
9. Click on **Databases → SQL Server → Hyper Historian →Data Sources → GetRawData.** Then click OK.
10. Remember to fill in all the values for the parameters. You already entered them in GridWorX64 Server when you ran Test Query, so, it should be easy task.

NOTE: Make sure that you have string delimiters "" before and after each parameter value as you work with string parameters in this case. Your Data Tag should look similar to this:
db:Hyper Historian.GetRawData<@StartTime="2012-06-08 10:00:00 AM", @EndTime="2012-06-08 11:00:00 AM", @LoggingGroup="LogGrp", @CollectingGroup="OPCUA", @TagName="Sine">

11. Click  on to refresh the data tag fields retrieved. You should see configuration shown on figure below.

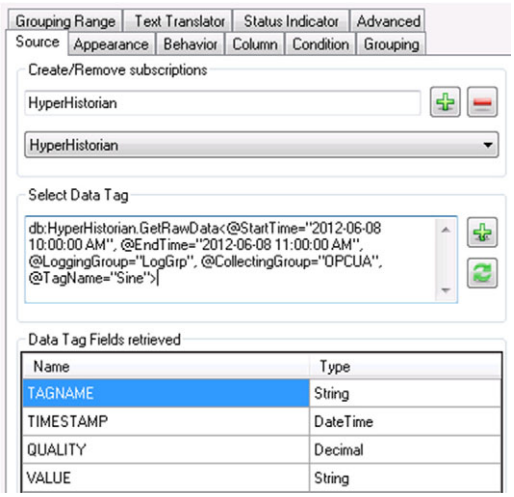

Figure 4 – Data Tag with parameters

12. Click Close to save.
13. Go to runtime and see the table with Hyper Historian data.

Example with dynamic values and GAS

This example shows how to read Hyper Historian data via GridWorX64 based on user inputs (StartTime, EndTime, LoggingGroup, CollectionGroup, TagName) and then sets these values into Global Aliases (GAS) which resolve the data dynamically in GridWorX64/TrendWorX64 Viewer.

NOTE: Please notice this example is designed for Advanced users where it is expected that you followed the first part of this app note and user has already some experience with Global Aliasing, Scripting. If you are first time user please refer to appropriate app notes first.

Please follow the steps below to get this example working:

1. Prepare following Global Aliases in Global Aliasing Configurator:
   - **<#HHServer#>** = Hyper Historian\\Configuration/
   - **<#HHSignal#>** = Theme Items (Sine, Random, Ramp,…)
   - **<#HHStartTime#> =** mm/dd/yyyy hh:mm AM
   - **<#HHEndTime#> =** mm/dd/yyyy hh:mm AM
   - **<#HHLoggingGroup#>** = LogGrp
   - **<#HHCollectionGroup#>** = OPCUA
2. Add following objects on GraphWorX64 display and give them the name that you can reference to them in the script:
   - GridWorX64 Viewer (if you did not already add)
   - TrendWorX64 Viewer
   - 2x DataTime Pickers (you can load it via Toolbox) for selecting StartTime, EndTime
   - Local Variables: Logging Group, Collection Group"
   - Button with Pick Action "Popup menu" with collection of signals (SetGlobalAlias for each item in collection to resolve <#HHSignal#> for each item).
   - "Set" button which will contain script in attachment.
3. **Add Hyper Historian Tag** (e.g. Sine) into TrendWorX64 Viewer and switch to runtime to see the data. If you get the data switch back to configuration mode and replace the pen with Global Alias as shown on example below. Don't forget to select "Use HDA Connection" in Advanced configuration. Your Data Source should look similar to this:
   ```
   <#HHServer#>/<#HHLoggingGroup#>/<#HHCollectionGroup#>/<#HHSignal#>
   ```
4. Modify the existing Data Source in GridWorX64 Viewer and add your Global Aliases. Your Data Source should look similar to this:
   ```
   db:Hyper
   Historian.GetRawData<@StartTime="<#HHStartTime#>",
   @EndTime="<#HHEndTime#>",
   @LoggingGroup="<#HHLoggingGroup#>",
   @CollectingGroup="<#HHCollectionGroup#>",
   @TagName="<#HHSignal#>">
   ```
5. Click on Refresh button to refresh the data source. After this you should see output similar to Figure 5.

   NOTE: Please notice that each Global Alias can have default value and you can pre-define it in Global Aliasing Configurator. If you don't have default value configured you will not see any data after you are done with step 5. For the control you can add your Global Aliases on GraphWorX64 display to be sure that GAS resolves some values. Don't forget to click on "Reload Configuration" ones you are done with changes in Global Aliasing Configuration.

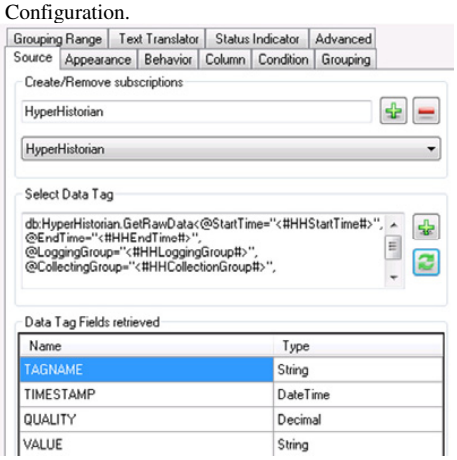

Figure 6 – Data Source with Global Alias GridWorX64 Viewer

6. After you verify that you can still resolve the data in TrendWorX64/GridWorX64 Viewer by using Global Alias you should see GraphWorX64 display which is similar to figure below.

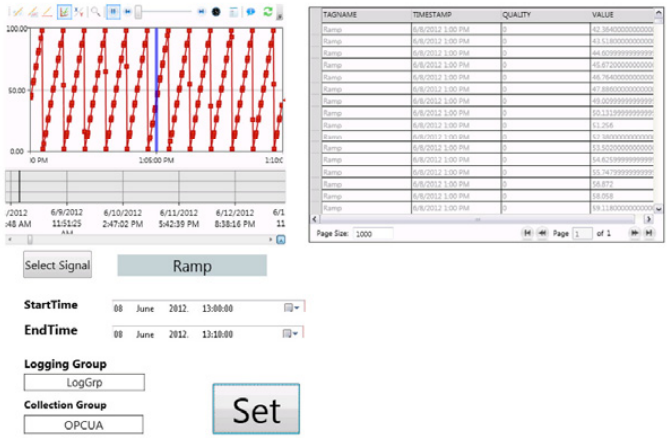

Figure 7 – GraphWorX64 display with all the Controls using GAS

7. Before you start with most important part (scripting), make sure that you gave the name to each object on GraphWorX64 display that you want to reference to. You can consult the names with the script in attachment or write the script from scratch.
8. Ones you are ready to write a script, add "Set" button on GraphWorX64 display (if you did not already add) and select **Pick Action "Run Script"**. Then switch to events and give your script the right name. Then follow the script in Attachment 1. For more details see GENESIS64 API.

Figure 8 – Set button with "Run Script" .

Attachment 1 (JScript):

```
function SetUserData(sender : System.Object, cmdArgs : Ico.Gwx.CommandExecutionEventArgs)
{
	// GridWorX64 Viewer
	var myGrid : Ico.Gdx.GdxViewControl = ThisDocument.GetElementByName("Grid1");
	var tabGDX : System.Windows.Controls.ItemsControl = myGrid.Items[0];
	var grid : Ico.Gdx.GdxGridView = tabGDX.Items[0];

	// TrendWorX64 Viewer variables
	viewer = Ico.Twx.TwxViewControl(ThisConfiguration.GetObjectByName("TWXViewer1").ToDependencyObject());
	tab = Ico.Windows.Controls.MultipleTabItem(viewer.Items[0]);
	chart = Ico.Twx.TwxChartComponent(tab.Items[0]);
	var localStartTime : System.DateTime;
	var localEndTime : System.DateTime;
	var startTime : System.DateTime;
	var endTime : System.DateTime;


	//DateTime Picker
	var wfCtl5 : Ico.Gwx.GwxWindowsFormsControl;
	var wfCtl6 : Ico.Gwx.GwxWindowsFormsControl;
	var myDateTimePicker1  : System.Windows.Forms.DateTimePicker;
	var myDateTimePicker2  : System.Windows.Forms.DateTimePicker;
	var myDateTimePickerStartTime : System.DateTime;
	var myDateTimePickerEndTime : System.DateTime;

	//Read Start/End Time from DateTime Picker
	wfCtl5 = Ico.Gwx.GwxWindowsFormsControl(ThisConfiguration.GetObjectByName("DateTimePicker1"));
	wfCtl6 = Ico.Gwx.GwxWindowsFormsControl(ThisConfiguration.GetObjectByName("DateTimePicker2"));
	myDateTimePicker1 = System.Windows.Forms.DateTimePicker(wfCtl5.Control);
	myDateTimePicker2 = System.Windows.Forms.DateTimePicker(wfCtl6.Control);
	myDateTimePickerStartTime = myDateTimePicker1.Value;
	myDateTimePickerEndTime = myDateTimePicker2.Value;

	// Read all other user inputs
	mystarttime = myDateTimePickerStartTime.Year + "-" + myDateTimePickerStartTime.Month + "-" +
myDateTimePickerStartTime.Day + " " + myDateTimePickerStartTime.ToLongTimeString();
	myendtime = myDateTimePickerEndTime.Year + "-" + myDateTimePickerEndTime.Month + "-" +
myDateTimePickerEndTime.Day + " " + myDateTimePickerEndTime.ToLongTimeString();
	myloggroup = ThisConfiguration.GetDynamicObjectByName("LoggingGroupBox").GetValueOfPrimaryDataSource();
	mycollectgroup =
ThisConfiguration.GetDynamicObjectByName("CollectionGroupBox").GetValueOfPrimaryDataSource();

	//Set all user inputs to GAS
	ThisWindow.SetGlobalAliases("#HHStartTime=" + mystarttime + "/0;");
	ThisWindow.SetGlobalAliases("#HHEndTime=" + myendtime + "/0;");
	ThisWindow.SetGlobalAliases("#HHLoggingGroup=" + myloggroup + "/0;");
	ThisWindow.SetGlobalAliases("#HHCollectionGroup=" + mycollectgroup + "/0;");

	// Set Time Range in TrendWorX64 Viewer
	if (DateTime.TryParse(mystarttime, startTime)
		&& DateTime.TryParse(myendtime, endTime))
	{
		localStartTime = DateTime.Parse(myDateTimePickerStartTime);
		startTime = localStartTime.ToUniversalTime();
		localEndTime = DateTime.Parse(myDateTimePickerEndTime);
		endTime = localEndTime.ToUniversalTime();
		chart.SetTrendPeriod(endTime, endTime.Subtract(startTime));
	}
	else
	{
		MessageBox.Show("Cannot parse the start/end date and time.");
	}

	//Refresh Grid Control
    grid.Refresh();

}
```