GraphWorX64 Scripting
Accessing a Database

APPLICATION NOTE

September 2013

Description: Guide to accessing a database using GraphWorX64 Scripting.
OS Requirement: Vista x64/ Windows 7 x64/ Windows 8 x64/ Windows Server 2008 x64/ Windows Server 2008 R2 x64/ Windows Server 2012
General Requirement: Installation of GENESIS64, SQL Server Management Studio or SQL Server Management Studio Express.

Introduction

Databases are being utilized more frequently to store a variety of information. It can be useful to be able to interface with such a database to extract the data that you need to use inside your GENESIS64 application.

In this Application Note, we are going to demonstrate how to access your data from a database using scripting and the .NET Framework DataGrid control.

> **NOTE:** GENESIS64 since version 10.6 includes a new GridWorX product that performs database operations on the server side and it is easy to configure for many projects. Moreover it does not require scripting. ICONICS highly recommends that you see the *GENESIS64 – Data Grid and GridWorX* application note first to determine if GridWorX is a better fit for your needs.

This Application Note assumes that you have read the GraphWorX64 Scripting - Quick Start application note.

Database Setup

If you want to use the *AdventureWorks* sample database that we use in this Application Note you can freely download it from CodePlex at one of these addresses, depending on your SQL Server version.

SQL 2008 R2:
http://msftdbprodsamples.codeplex.com/releases/view/55926

SQL 2008:
http://msftdbprodsamples.codeplex.com/releases/view/37109

SQL 2005:
http://msftdbprodsamples.codeplex.com/releases/view/4004

Feel free to use a different database if you have one already created or if you would like to create your own. If you are using a database other than *AdventureWorks2008R2* you will need to use your custom connection string and query in place of the *AdventureWorks2008R2* connection string and query throughout this example.

Setting up the GraphWorX64 Display

To start, create a new GraphWorX64 display, add a button, and set the pick action for the button to be "Run Script". Double-click the "RunScript" event to generate the script function associated to this button.

> **NOTE:** Please refer to the *GraphWorX64 Scripting - Quick Start* Application Note if you need more information on how to start with scripting in GraphWorX64.

To show the database content we will use the .NET Framework "DataGridView" control. You can access this user control from the Toolbox, selecting the category "Windows Forms" as shown in Figure 1.

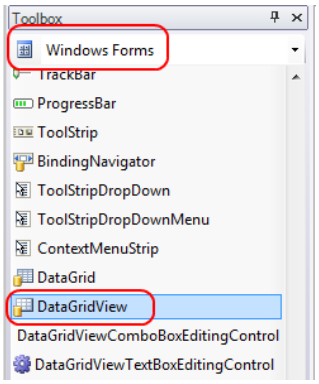

Figure 1 - The DataGridView Control in the Toolbox

> **NOTE:** If you do not see the "DataGridView", you can add it by clicking the "Add/Remove Components" button on the Windows Forms Toolbox. Browse for the *System.Windows.Forms.dll* usually located in C:\Windows\Microsoft.NET\Framework64\v2.0.50727. This will add a few more .NET controls into your Toolbox.

Drag this control on the screen, and size it as desired. Your current display should look similar to Figure 2, where the selected gray rectangle is the DataGridView control in configuration mode.

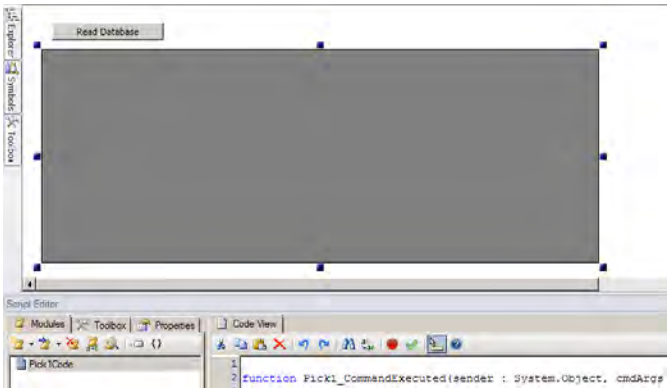

Figure 2 - Display with a DataGridView Control and Button

The display is ready. We just need to write some script to have it connect to our database.

Accessing the Database Content

The script code we are going to write consists of the following:

- The connection string used to connect to the database
- The button's script function used to trigger the reading from the database
- A utility function used to populate the grid

Let us start by defining the connection string. Yours might be slightly different depending on the location of your SQL Server, the database name, and security settings. In this example, we show the easiest possible configuration. In this case, the server is running locally and using integrated security.

Our connection string is defined in the script module as shown below. Place this line outside of your button function at the top of the script module.

```
var connectionString : String = "Data
Source=(local);Initial
Catalog=AdventureWorks2008R2;Integrated
Security=SSPI";
```

You can see that we have used "(local)" as our data source, because the server is local, and the database name is "AdventureWorks2008R2". If you installed a version of SQL Express then you may need to use "(local)\SQLEXPRESS" as your data source.

Next, we will write the utility function that will populate the grid. This function is made to accept two parameters: the name of the grid to populate, and the query to use to get the data. This piece also goes outside of the button function. It does not matter if it's above or below the button function.

```
function PopulateGrid(gridName, commandString)
{
    // Declare variables
    var wfCtl : Ico.Gwx.GwxWindowsFormsControl;
    var grid : System.Windows.Forms.DataGridView;
    var conn : System.Data.SqlClient.SqlConnection;
    var command : System.Data.SqlClient.SqlCommand;
    var dataAdaptor:
System.Data.SqlClient.SqlDataAdapter;
    var dataTable: System.Data.DataTable;

    // Initialize variables
    conn = new
System.Data.SqlClient.SqlConnection(connectionStrin
g);
    command = new
System.Data.SqlClient.SqlCommand(commandString,
conn);
    dataAdapter = new
System.Data.SqlClient.SqlDataAdapter(command);
    dataTable = new System.Data.DataTable();
    dataAdapter.Fill(dataTable);

    // Get the control and give it properties
    wfCtl =
Ico.Gwx.GwxWindowsFormsControl(ThisConfiguration.Ge
tObjectByName(gridName));
    grid =
System.Windows.Forms.DataGridView(wfCtl.Control);
    grid.DataSource = dataTable;
}
```

In the function, you can identify three different blocks. In the first block, we have declared all the variables we are going to use. In the second block, we initialize them. We start creating a SqlConnection using the connection string we built before. Using this connection, we prepare a SqlCommand object, passing it the query we got as a parameter. We also create and prepare a SqlDataAdapter and a DataTable object.

In the last block, we get a reference to the data grid, and then we fill its DataSource property with the DataTable object created earlier.
The last thing we need to do to complete our example is to call our PopulateGrid function when the button is clicked. To do so we fill the button's pick function as follows:

```
function Pick1_CommandExecuted(sender :
System.Object, cmdArgs :
Ico.Gwx.CommandExecutionEventArgs)
{
    var commandString : String;
    commandString = "SELECT * FROM
Production.Product";
    PopulateGrid("DataGridView1", commandString);
}
```

The first two lines define the query to select all the records from the *Production.Product* table. Last line calls the function we defined earlier to populate the data grid.

> **NOTE:** We pass the name "DataGridView1" into the PopulateGrid function. This is the default name of your DataGridView object. To see or change the name of your DataGridView object, select it and go to the Properties panel. Scroll down to the Design section and look at the (Name) field.

The example is now complete, you may want to save your display, and then go into Runtime mode to try it out. When clicking the button you should see your data populated in the grid.

ICONICS Customer Connection Portal hosts a library of automation examples that are very handy when you start scripting.

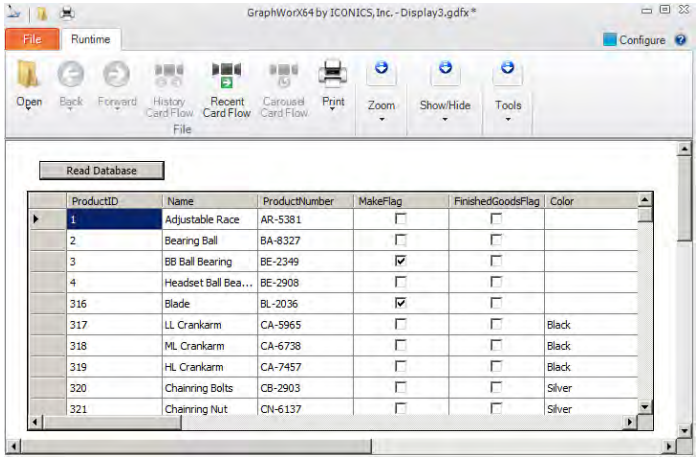

Figure 3 – Database Data Populated in the DataGridView Control

Final Code

Your final code should look something like this in the script editor:

```
var connectionString : String = "Data Source=(local);Initial Catalog=AdventureWorks2008R2;Integrated
Security=SSPI";

function Pick1_CommandExecuted(sender : System.Object, cmdArgs : Ico.Gwx.CommandExecutionEventArgs)
{
        var commandString : String;
        commandString = "SELECT * FROM Production.Product";
        PopulateGrid("DataGridView1", commandString);
}

function PopulateGrid(gridName, commandString)
{
        // Declare variables
        var wfCtl : Ico.Gwx.GwxWindowsFormsControl;
        var grid : System.Windows.Forms.DataGridView;
        var conn : System.Data.SqlClient.SqlConnection;
        var command : System.Data.SqlClient.SqlCommand;
        var dataAdaptor: System.Data.SqlClient.SqlDataAdapter;
        var dataTable: System.Data.DataTable;

        // Initialize variables
        conn = new System.Data.SqlClient.SqlConnection(connectionString);
        command = new System.Data.SqlClient.SqlCommand(commandString, conn);
        dataAdapter = new System.Data.SqlClient.SqlDataAdapter(command);
        dataTable = new System.Data.DataTable();
        dataAdapter.Fill(dataTable);

        // Get the control and give it properties
        wfCtl = Ico.Gwx.GwxWindowsFormsControl(ThisConfiguration.GetObjectByName(gridName));
        grid = System.Windows.Forms.DataGridView(wfCtl.Control);
        grid.DataSource = dataTable;
}
```