GENESIS64 Security – Retrieving Advanced Security Information

APPLICATION NOTE

September 2013

Description: This Application Note explains how to access detailed security information from a GraphWorx64 display.
OS Requirement: Vista x64/ Windows 7 x64/ Windows 8 x64/ Windows Server 2008 x64/ Windows Server 2008 R2 x64/ Windows Server 2012
General Requirement: GENESIS64 installed on your machine.

Introduction:

GENESIS64 Security allows you to set individual rights and accesses based on users and groups, and create named policy settings that can be applied to users and groups. Additionally, it also allows you to control access to individual applications, features, systems, and more, based on time settings, if you wish.

One feature that is integrated into GENESIS64 Security and GraphWorx64 is the ability to query the level of access the logged in user has on a Data Point, Alarm, File, etc.. With this functionality, one can then secure GraphWorX64 further and even incorporate these values into expressions or dynamics.

This Application Note will go over several advanced features of GENESIS64 security, including showing the logged-in user through a GraphWorX64 Local Simulation variable, retrieving the level of access the current logged in user has on a Data Point, Alarm, and File, as well as show you how to verify the custom string against the logged-in user.

Setting up Security:

We will create one user in GENESIS64 Security that will be used in the examples for the rest of this Application note.

1. Open the Workbench from Start → All Programs → ICONICS → Tools → Security.
2. Expand database to the Users level. Right click on Users to create a new user called **TestUser**. Include a password.

Figure 1 – Setting up a new User

3. For the following steps, assumed that the phrase "Allow these operations" has an '*' following/below it. If it does not, please add the '*' wherever this text is seen.
4. On the Points tab enter ***Setpoint*** in the "Deny these operations" section and remove the checkmark from the Write column as shown in Figure .

Figure 2 - Security Points Configuration

5. On the Alarms tab enter ***Pump*** in the "Deny acknowledgement of these alarms" section.
6. On the Files tab enter ***Supervisor*** in the "Deny these" section and remove the checkmark from the Create and Modify columns as shown in Figure .

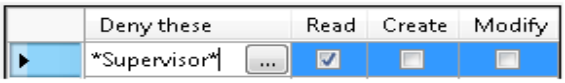

Figure 3 - Security Files configuration

7. On the Custom tab enter **customString** in the "Deny these operations" section.
8. Click Apply to save your changes.
9. If necessary, Login (on Home ribbon) with your new credentials to continue on with the next section.

NOTE: The * (asterisk) wildcard is used to represent zero or more characters.

For more information on setting up security please refer to the Help documentation or the *GENESIS64 Security – Security Quick Start* Application Note.

Displaying the Current Logged-In User:

There is a local simulation variable available in GraphWorX64 that retrieves the current logged-in user.

1. Open a GraphWorX64 display.
2. Add a Process Point to the display.
3. For the Datasource enter **localsim::currentUser**
4. Enter runtime and the Process Point should show something similar to Figure 4.

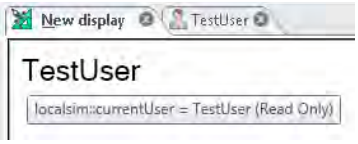

Figure 4 - localsim::currentUser in a PPT

You could then retrieve the logged-in User from a script by getting the value of this Process Point. Please refer to the Help documentation or *GraphWorX64 Scripting – Quick Start* for more information on scripting.

Retrieving Read, Write, and Modify Permissions:

Once you have security configured, you can view what level of access the current user has by using a special syntax available to GENESIS64. The syntax is: ?Category:comparision string

The Category can be any of the following

- **ApplicationAction**
- **DataPoint**
- **Alarm**
- **File**
- **Station**
- **CriticalPoint**
- **CriticalAlarm**
- **Report**
- **Transaction**
- **Custom**

The "comparison string" is basically any text that follows such as an OPC tag or a simple string. For example, the following are valid ways to retrieve the level of security for an alarm called "LowPressure" and an OPC tag from the ICONICS Simulator OPC Server:

?Alarm:LowPressure
?DataPoint:@ICONICS.Simulator.1\SimulatePLC.OUTPUTS.FLOAT1.Value

With the proper syntax in place, the following truth table shows the possible values this special security syntax can return. The resulting values are the decimal representation of the binary bits being true.

Table 1 - Permission Results

Modify 4	Write(Create) 2	Read 1	Returned Value
0	0	0	**0**
0	0	1	**1**
0	1	0	**2**
0	1	1	**3**
1	0	0	**4**
1	0	1	**5**
1	1	0	**6**
1	1	1	**7**

NOTE: Depending on the category you are trying to access (DataPoint, Alarm, File, etc.) the Modify and Write/Create may not apply.

Let us now bring the above topics together with an example we started to configure in the previous section:

1. Open a GraphWorX64 Display.
2. Add a new Process Point using each of the following Datasources:

   ?DataPoint:localsim:TankSetPointLow
   ?Alarm:PumpStatus
   ?File:MaintenenceSupervisorControl

3. Put the display in runtime and you should see values similar to the ones in Figure 5.

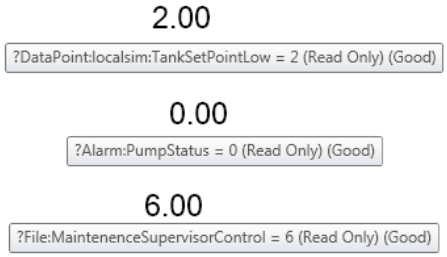

Figure 5 - Results in GraphWorX64 Runtime

Keep in mind that the Deny permissions take precedence over the Allow when manually calculating these values. If you have been following this Application Note, then there should be an ***** in the allow fields of each tab to allow everything by default. Feel free to experiment further with different combinations of Read, Write, and Modify permissions.

Checking Custom String from a GraphWorX64 Script:

If you are familiar with GraphWorX64 scripting you should be able to easily see if the current logged in user by trying the steps below. Please refer to the GraphWorX64 Scripting Application Notes and Help documentation for more information on scripting.

1. Open a GraphWorX64 Display
2. Add a button that will use the **Run Script** command and give the script a name (check the Scripts box in the View tab of the Workbench ribbon if necessary).
3. Click on Events ( ), and under the Dynamic Behavior section, double click on the empty space next to RunScript to add a new form in the Script Editor.
4. In the Script Editor, you will want to use the **ThisWindow.IsCustomSecurityAllowed("customString")** function. Similar to the example below.

```
var result : Boolean = ThisWindow.IsCustomSecurityAllowed(
"customString");

MessageBox.Show(result);
```

5. This should display a message box with a value of "false" when the button is pressed during runtime.