

Description: An introduction to using Information Broker features to communicate between ICONICS products and an MQTT broker.

General Requirements:

- Basic knowledge of GENESIS64 and GraphWorX64.
- Basic knowledge of BridgeWorX64 and a license with at least one available transaction (for BridgeWorX64-related examples).
- Familiarity with MQTT brokers (specifically Mosquitto).

Introduction

Information Broker is a set of features for the IoTWorX Point Managers and BridgeWorX64, designed to connect GENESIS64 and ICONICS products to MQTT brokers, such as Mosquitto and ActiveMQ.

This document will walk you through setting up ICONICS communication to publish and subscribe to an MQTT broker.

Installing or Deploying the MQTT Broker

For a Windows device, if you do not already have an MQTT broker you can download the Mosquitto MQTT Broker from the Mosquitto website at: <https://mosquitto.org/download/>

If you have an edge device with ICONICS IoT Edge software installed, you can deploy a Mosquitto module during the deployment phase. When configuring modules for the IoT edge device, simply check the Mosquitto box in the Deployment Modules section.

The following steps will assume that Mosquitto MQTT Broker has been installed to the default location (C:\Program Files\mosquitto) on the same system as GENESIS64. This is for testing or demonstration purposes. In a real production system, the broker is likely to be installed on a remote system or edge device.

Basic Setup

These steps are required for both subscribing and publishing to an MQTT broker.

1. Open **Workbench**.
2. Expand **Internet of Things**.
3. In the **MQTT Brokers** folder, create a new **MQTT Broker**.
4. Set the name to **MyBroker**.
5. For **Protocol Type**, choose **Simple MQTT ("mqtt")**.

6. For **Server Address**, enter the IP address where the broker is located. For a locally installed broker, enter **localhost**.
7. Apply the changes.
8. In the **Subscriber Connections** folder, create a new subscriber connection.
9. Set the name to **MQTTSubscriber**.
10. Uncheck **Enable compatibility with ICONICS clients**.
11. Set the **Connection Type** to **MQTT**.
12. In the **MQTT Basic Settings** section, for **MQTT Broker** choose **MyBroker**.
13. Set the **Basic Topic** to **data/#**.
14. Apply the changes.
15. Use the stoplight icons in the **Home** menu to ensure that the **Subscriber** and **Publisher** services are running.

Note: If you do not see the **Publisher** and **Subscriber** service stoplights in the **Home** menu, make sure you have the **Internet of Things** or a child item selected in the **Project Explorer**.

16. If Mosquitto is not running, start it with this command:
`"C:\Program Files\mosquitto\mosquitto.exe" -v`

Subscribing to MQTT

The steps below describe a few different ways to get information from an MQTT broker into GENESIS64 or BridgeWorX64.

Subscribing to MQTT Events in Alarm Viewer

ICONICS event clients, such as AlarmWorX64 Viewer can subscribe to MQTT messages.

1. If you haven't already done so, follow the steps in "Basic Setup".
2. Open **GraphWorX64**.
3. Add a new **AlarmWorX64 Viewer**.
4. Configure the alarm viewer.
5. Select the **grid** object in the tree.
6. On the **Source** tab, add a new real time subscription.
7. Remove the default **Local Alarm Server** event point.
8. Add a new event point.
9. In the tag browser, expand **My Computer > Internet of Things** then select **MQTTSubscriber**.
10. Select **OK**.
11. The resulting point name should be:
iot:MQTTSubscriber/.
12. Close the configuration window.
13. Go into runtime.

14. Trigger a message from your MQTT broker. With Mosquitto, you can execute this command:

```
"C:\Program Files\mosquitto\mosquitto_pub.exe"
-t data/a -m message
```

You should see your message appear in the alarm viewer.

Subscribing to MQTT Datasets

Subscribed MQTT information can also be made available as a dataset and viewed with clients such as the GridWorX Viewer.

1. If you haven't already done so, follow the steps in "Basic Setup".
2. Open **Workbench**.
3. Expand **Internet of Things**.
4. In the **Custom Encoders/Decoders** folder, create a new encoder/decoder.
5. Set the name to **MQTTDecoder**.
6. Set **Plugin** to **CustomJson**.
7. Set **Message Type** to **The whole message is a dataset**.
8. Apply the changes.
9. Edit **Internet of Things > Subscriber Connections > MQTTSubscriber**.
10. In the **General Settings** section, set **Default Decoder** to **MQTTDecoder**.

Note: If you do not see **MQTTDecoder** in the list, use the refresh button next to the **Default Decoder** field.

15. Apply the changes.
11. Create a new file under **C:\temp** named **message.json**.
12. Edit the file with **Notepad** or another text editor.
13. Type or paste this content into the text editor:

```
[
  {
    "first": "alice",
    "last": "brown",
    "age": 38
  },
  {
    "first": "bob",
    "last": "smith",
    "age": 22
  },
  {
    "first": "carol",
    "last": "polak",
    "age": 73
  },
  {
    "first": "dave",
```

```
    "last": "jones",
    "age": 49
  }
]
```

16. Save and close the file.
17. Run this command to publish the file we created:


```
"C:\Program Files\mosquitto\mosquitto_pub.exe"
-t data/a -f c:\temp\message.json
```
14. Open **GraphWorX64**.
15. Add a new **GridWorX Viewer**.
16. Configure the grid viewer.
17. Select the **grid** object in the tree.
18. On the **Source** tab, add a data set tag subscription.
19. In the tag browser, expand **My Computer > Internet of Things** then select **MQTTSubscriber**.
20. Select **OK**.
21. The resulting point name should be: **iot:MQTTSubscriber/**.

Note: If you get an error saying the data fields could not be retrieved, run the command above to publish the JSON file again, then hit the refresh button in the grid viewer configuration. The file needs to be published at least once for the subscriber to know what fields to expect.

18. Close the configuration window.
 19. Go into runtime.
 20. Run this command again to republish the JSON file:


```
"C:\Program Files\mosquitto\mosquitto_pub.exe"
-t data/a -f c:\temp\message.json
```
- You should see the data appear in your grid viewer. (If it does not appear after a few moments, refresh the viewer.)

Using BridgeWorX64 to Parse Datasets

BridgeWorX64 transactions can be used to subscribe to MQTT events and parse specific information out of the dataset.

1. If you haven't already done so, follow the steps in "Basic Setup" and "Subscribing to MQTT Datasets".
2. Open **Workbench**.
3. Expand **Bridging**.
4. In the **Templates** folder, create a new template.
5. Set the name to **MQTTTemplate**.
6. Drag a **JSON Content Reader** activity into the diagram.
7. Name the activity, **Read JSON**.
8. Set the **Activity Description** to **Reads the JSON from the alarm message field and converts it to a dataset that can be used by the next activity**.

9. In the **JSON Reader Settings** section, select **Edit Expression**.
10. In the sidebar, select **Trigger Attributes**.
11. In the dropdown at the top of the sidebar, select **Alarm Trigger**.
12. Double-click on **Message**.
13. Select **OK**. Your resulting expression should look like this:

```
x={{triggerVariable:Message}}
```
14. Set **JSON Parse Mode** to **Match Predefined Columns by Names**.
15. In the **Data Schema** section, select **Configure**.
16. Delete all but three of the items in this list.
17. Edit the remaining three items so they have these names in this order: **first**, **last**, **age**.
18. Select **OK**. Your configured schema should look like Figure 1.

Data Schema (Configure) (Reset)		
#	Name	Type
0	first	String
1	last	String
2	age	String

Figure 1 - Data schema

19. Drag a **Real Time Output** activity into the diagram.
20. Name the activity, **Write Output**.
21. Set the **Activity Description** to **Writes select pieces of the dataset from the previous block to output tags**.
22. Add two items to the **Data Sources** table, and configure them as follows:

Point Name	Output Expression
@sim64:String.Static("Static1").Value	x=findoutputcell("Read JSON", "last='polak'", "first")
@sim64:String.Static("Static2").Value	x=findoutputcell("Read JSON", "last='polak'", "age")

Note: The *findoutputcell()* function searches for a particular row in the given activity's dataset and returns a value in another column. In this case, it is looking for the row whose "last" column has a value of "polak". One expression returns the value in the "first" column and the other returns the value in the "age" column.

23. Connect the activities as shown in Figure 2 below.

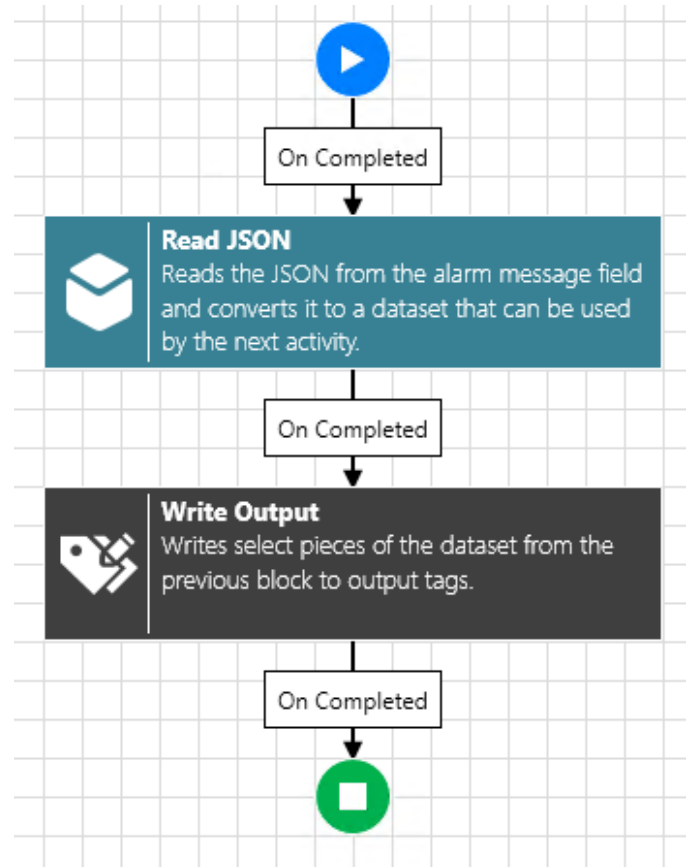


Figure 2 - Transaction diagram

24. Apply the changes.
25. Expand **Transactions**.
26. In the **Sample Configuration**, create a new transaction.
27. Set the name to **MQTTTransaction**.
28. Check **The transaction is enabled**.
29. Go to the **Alarm Subscriptions** tab.
30. Check **Trigger transaction using alarm source**.
31. Add a new item to the **Alarm Subscription** list.
32. Select the **browse** icon to bring up the tag browser.
33. Browse to **My Computer > Internet of Things** then select **MQTTSubscriber**.
34. Select **OK**.
35. Select the **@ICONICS.AlarmSvr.1.Server** tag and delete it.
36. Apply the changes.
37. In the **Home** menu, use the stoplight to make sure the **Point Manager and Scheduler** service is running.

Note: If you do not see the **Point Manager and Scheduler** service stoplight in the **Home** menu, make sure you have **Bridging** or a child item selected in the **Project Explorer**.

38. Open **GraphWorX64**.

39. Add two process points with these data sources:

```
@sim64:String.Static("Static1").Value
```

```
@sim64:String.Static("Static2").Value
```

40. Put the display into runtime.

41. Run this command to republish the JSON file:

```
"C:\Program Files\mosquitto\mosquitto_pub.exe"  
-t data/a -f c:\temp\message.json
```

Running the command should create an event, which triggers the BridgeWorX64 transaction. The transaction parses the JSON in the message field and writes out the first name and age into the two simulator points. The two points on the display should now show "carol" and "73".

Publishing to MQTT

The steps below describe a few different ways to send information from GENESIS64 or BridgeWorX64 to an MQTT broker.

Commanding

1. If you haven't already done so, follow the steps in "Basic Setup".
2. Open **GraphWorX64**.
3. Add a button.
4. Set the **Command** to **Call Method**.
5. In the **Name** field, select **Browse/Edit**.
6. Browse to **My Computer > Internet of Things > MQTTSubscriber** then select **Publish**.
7. Select **OK**.
8. Select the **ellipsis** button for **Parameters**.
9. Add three members and configure them as follows:

Value	ValueType
data/a	String
Hello World	String
0	Int32
false	Boolean

10. Select **OK**.

11. Go into runtime.

12. Run this command to see the messages being published to the MQTT broker:

```
"C:\Program Files\mosquitto\mosquitto_sub.exe"  
-t #
```

13. In **GraphWorX64**, hit the button you created. You should see a "Hello World" message appear in the subscriber window.

Scripting

1. If you haven't already done so, follow the steps in "Basic Setup".
2. Open **GraphWorX64**.
3. Add a button.
4. Set the **Command** to **Run Script**.
5. Select the **events**  icon.
6. Double-click in the **RunScript** field.
7. Select **OK** to the **Script Options** dialog.
8. The **Script Editor** should appear. Replace the contents of the Code View tab with the following code.

```
function Pick1_CommandExecuted(sender :  
System.Object, cmdArgs :  
Ico.Gwx.CommandExecutionEventArgs)  
{  
    var pub = new Object[4];  
    pub[0] = "data/a";  
    pub[1] = "Hello from scripting";  
    pub[2] = 0;  
    pub[3] = false;  
    var callback :  
Ico.Fwx.ClientWrapper.MethodCallDoneDelegate =  
ThisDocument.CreateDelegate(Ico.Fwx.ClientWrap  
per.MethodCallDoneDelegate, this, "CallDone");  
    ThisWindow.FwxClientWrapper.MethodCallA  
sync("iot:MQTTSubscriber/", "Publish", pub,  
callback, null);  
}  
  
function CallDone(result :  
Ico.Fwx.ClientWrapper.MethodCallDoneResult)  
{  
    message : String;  
    message = String.Format("Result: {0}",  
result.Status.ToString());  
  
    MessageBox.Show(message, "Publish  
Result", MessageBoxButton.OK);  
}
```

14. Go into runtime.
15. Run this command to see the messages being published to the MQTT broker:


```
"C:\Program Files\mosquitto\mosquitto_sub.exe"
-t #
```
16. In **GraphWorX64**, hit the button you created. You should see a "Hello from scripting" message appear in the subscriber window.

BridgeWorX64

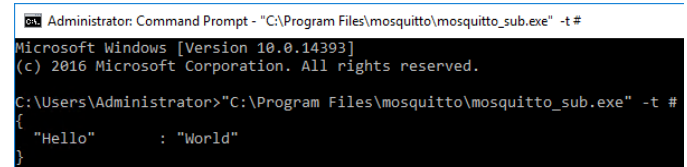
1. If you haven't already done so, follow the steps in "Basic Setup".
2. Open **Workbench**.
3. Expand **Bridging**.
4. In the **Templates** folder, create a new template.
5. Set the name to **MQTTPublisher**.
6. Add an **Information Broker Publisher** activity to the diagram.
7. In the **Information Broker Activity** section, select the **browse** button next to **Publish Point**.
8. Browse to **My Computer > Internet of Things > MQTTSubscriber** then select **Publish**.
9. Select **OK**.
10. Set **MQTT Topic** to **data/a**.
11. Under **JSON Payload Preview**, select **JSON Designer**.
12. Select **ScalarPropertyX**.
13. Select **Delete Node**.
14. Select **ScalarPropertyY**.
15. Set **Node Name** to **Hello**.
16. Set **Value Expression** to **x="World"**.
17. Select **OK**.
18. Make sure the start node points to your Information Broker Publisher activity, and your Information Broker Publisher activity points to the end node.

19. Apply your changes.
20. Expand **Transactions**.
21. In the **Sample Configuration**, create a new transaction.
22. Set the name to **PublishToMQTT**.
23. Check **The transaction is enabled**.
24. Go to the **Triggers** tab.
25. Add a new item to the list.
26. Select the **browse** icon to bring up the tag browser.
27. Browse to **My Computer > Actions > Triggers > Date/Time Triggers** then select **Every Minute**.
28. Select **OK**.
29. Apply the changes.
30. In the **Home** menu, use the stoplight to make sure the **Point Manager and Scheduler** service is running.

Note: If you do not see the **Point Manager and Scheduler** service stoplight in the **Home** menu, make sure you have **Bridging** or a child item selected in the **Project Explorer**.

31. Run this command to see the messages being published to the MQTT broker:


```
"C:\Program Files\mosquitto\mosquitto_sub.exe"
-t #
```
32. Wait for your computer's clock to tick over to a new minute. You should see a "Hello World" JSON appear in the subscriber window.



```
Administrator: Command Prompt - "C:\Program Files\mosquitto\mosquitto_sub.exe" -t #
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>"C:\Program Files\mosquitto\mosquitto_sub.exe" -t #
{
  "Hello"      : "World"
}
```

Note: When the test is complete, you may want to disable the PublishToMQTT transaction to avoid unnecessary MQTT publishing.