

# Sending email when link to data source is lost

## Overview

This document describes how to configure the OPC DataHub to send an email notification whenever the link to a particular data source is lost. In this example, we will monitor a point being read from an OPC server (TOP Server) and we will generate an email notification whenever the link to the server is interrupted and the quality of the data point we are monitoring changes to NOT CONNECTED.

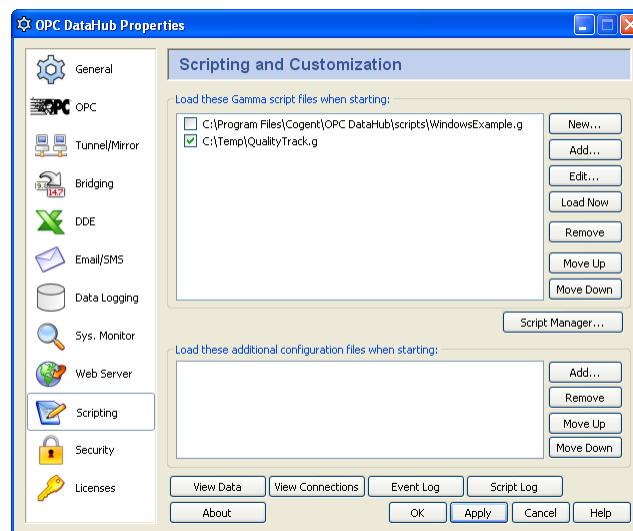
## How it works

We will use a DataHub script (as shown at the end of this document) to monitor the quality of a point being read from the TOP Server. The script will use a timer to read the quality value of the monitored point and write this as the value to a new point in the DataHub (the trigger point) that will be used to trigger the email notification.

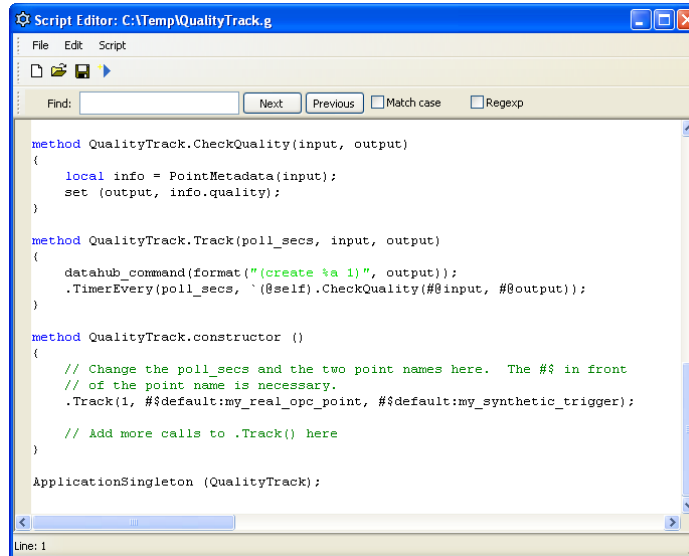
We will then configure the Email feature of the OPC DataHub to trigger a new email whenever the value of our trigger point is set to a value that equals NOT CONNECTED.

## Step 1 – Loading the DataHub script on start-up

1. Copy the DataHub script into a text file and save it as **QualityTrack.g**.
2. In the OPC DataHub properties window, click on the **Scripting** icon.
3. Click on the **Add** button and select the QualityTrack.g script file to add it to the list.
4. Make sure you **click to select the checkbox** next to the QualityTrack.g script so that the script is run each time the DataHub starts.



5. Highlight the QualityTrack.g script in the list and then click the **Edit** button.



6. In the script editor, scroll to the bottom of the page until you locate the following line:  
`.Track(1, #$$default:my_real_opc_point, #$$default:my_synthetic_trigger);`
7. This is the line in the script that you need to edit so that it watches the quality of a point coming from the data source you want to monitor. For this example we will tell the script to monitor a point being read from the TOP Server simulation data set.

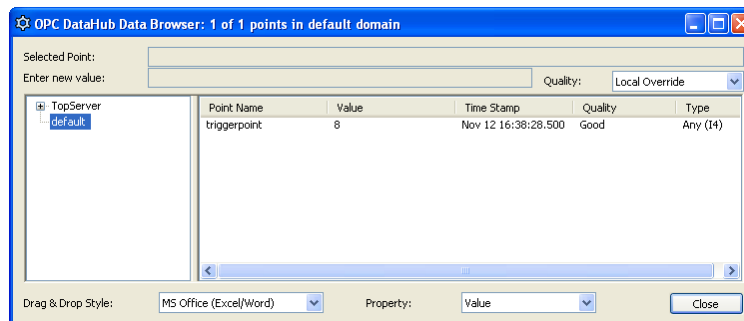
`.Track(1, #$$TopServer:Channel_0_User_Defined.Random.Random1, #$$default:triggerpoint);`

In this case, we are telling the script to watch the quality of the **Channel\_0\_User\_Defined.Random.Random1** point which is in the **TopServer** domain within the DataHub. We have also specified that the script should write to a point called **triggerpoint**, which is to be created in the **default** domain. The number 1 after **.Track()** tells the script to read the point quality every 1 second.

8. When you have defined the data point you want to monitor, and configured the trigger point name, then you should choose **File, Save** and then choose **Script, Reload Whole File** to save your changes and reload the script.

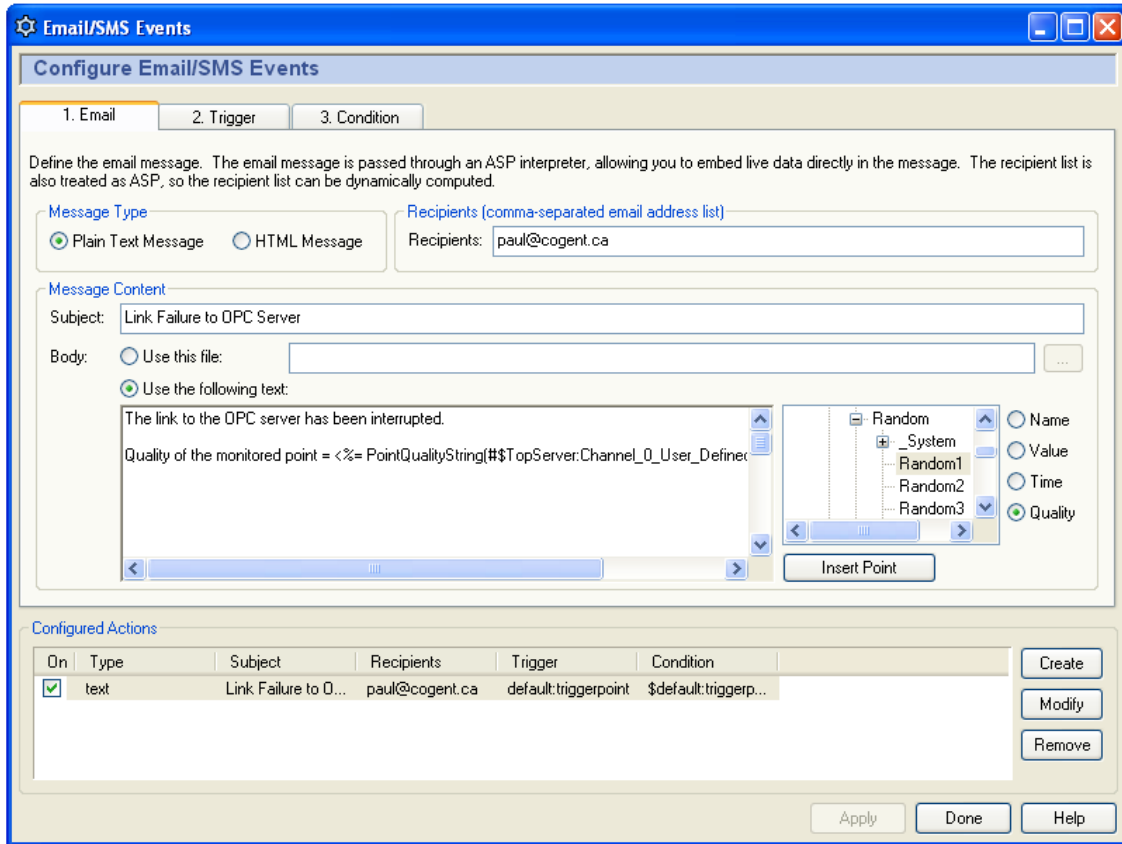
### Testing the DataHub script

To test the QualityTrack.g script, you can go to the OPC server configuration and disable the connection to the server. If you then open up the Data Browser window, you should see the trigger point with a value of 8, which indicates a quality of NOT CONNECTED in the monitored point.

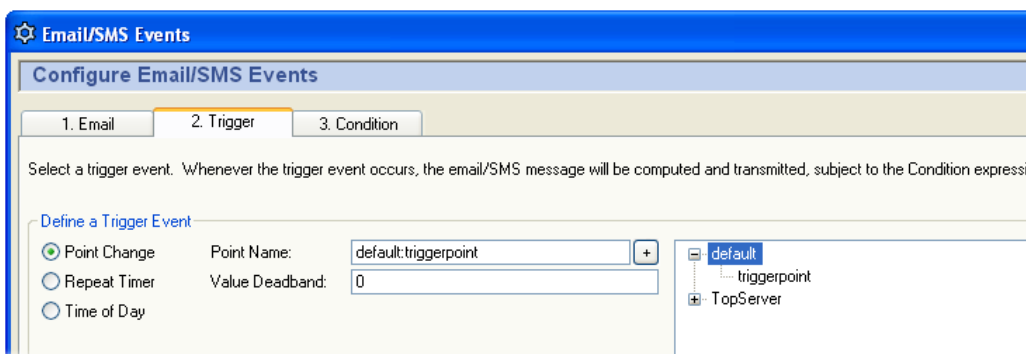


## Step 2 – Configuring the Email Notification Trigger

1. In the Email configuration window shown below, you will see that we have configured an email message to be sent that has a short plain text message and it also includes the quality of the point we are monitoring.

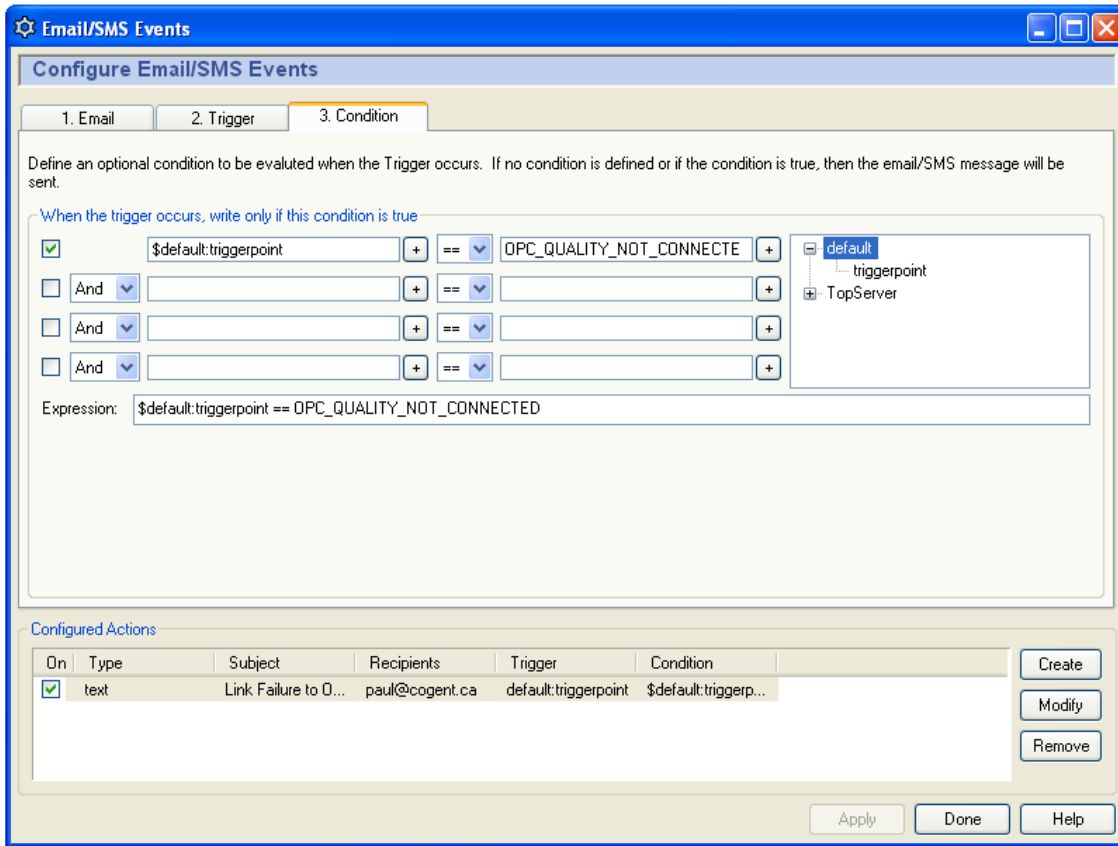


2. In the trigger tab, we have selected the name of our trigger point as the trigger event.



3. In the condition tab, you will see that we have defined a condition that tests to see if the value of the `triggerpoint` is equal to `OPC_QUALITY_NOT_CONNECTED` which is the internal OPC quality

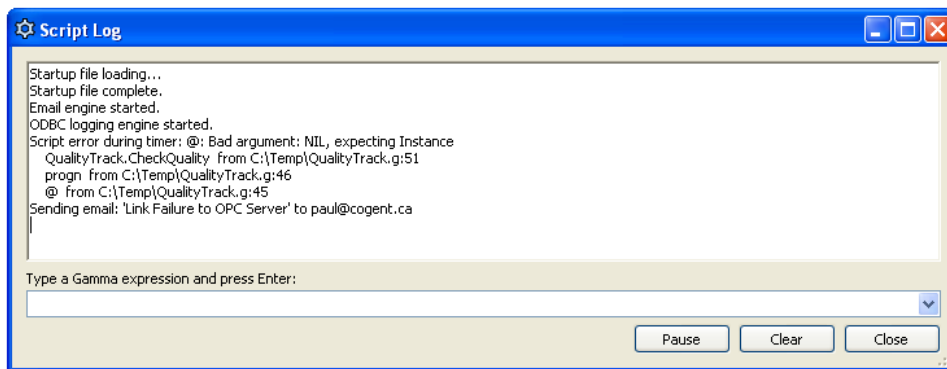
constant that maps the quality value of 8 to a NOT CONNECTED string value (see the image below). The DataHub script itself gives a complete list of quality constants you can choose from.



For more information on how to configure the Email feature of the OPC DataHub, please refer to the documentation here <http://www.opcdatahub.com/Docs/dho-mailer.html>.

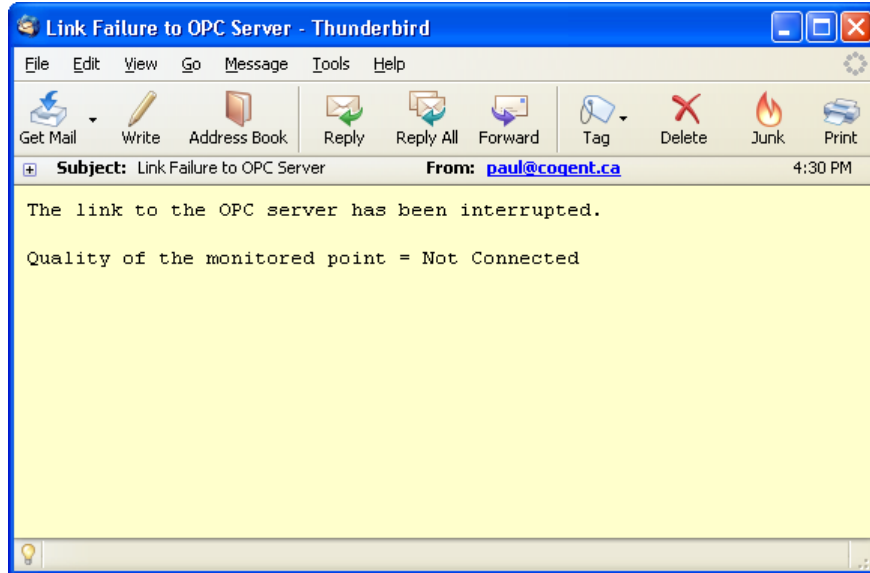
### Testing the Email Notification

To test the email notification, you may find it useful to open the Script Log to see the email being sent out. Once again, disable the connection to the OPC server and you should see the email notification recorded in the Script Log window.



If you see the script error as shown above, you can ignore this. It just means that at start-up, the QualityTrack.g script tried to evaluate the monitored point, before it had been read from the OPC server.

In our example, the email message that is generated will look similar to the image below.



## QualityTrack.g DataHub Script

/\*  
 This script polls the quality of an "input" point and writes the quality as the value of another "output" point.  
 The input point is typically an actual point from an OPC server. The output point is a synthetic point that can be used to trigger email and ODBC events.

To alter this script, just edit the QualityTrack.constructor to change the parameters of the .Track() call. The parameters are:

- poll\_secs - the polling rate. This can be fractional, as in 0.5
- input - the name of the input point as a symbol
- output - the name of the output point as a symbol

You can add as many .Track() calls as you need to monitor multiple points.

In an email or ODBC Condition configuration, the quality of a point can be compared to the OPC quality constants:

```
OPC_QUALITY_BAD
OPC_QUALITY_COMM_FAILURE
OPC_QUALITY_CONFIG_ERROR
OPC_QUALITY_DEVICE_FAILURE
OPC_QUALITY_EGU_EXCEEDED
OPC_QUALITY_GOOD
OPC_QUALITY_LAST_KNOWN
OPC_QUALITY_LAST_USABLE
OPC_QUALITY_LOCAL_OVERRIDE
OPC_QUALITY_MASK
OPC_QUALITY_NOT_CONNECTED
OPC_QUALITY_OUT_OF_SERVICE
OPC_QUALITY_SENSOR_CAL
OPC_QUALITY_SENSOR_FAILURE
OPC_QUALITY_SUB_NORMAL
OPC_QUALITY_UNCERTAIN
```

```
*/

require ("Application");
require ("Time");

class QualityTrack Application
{
}

method QualityTrack.CheckQuality(input, output)
{
    local info = PointMetadata(input);
    set (output, info.quality);
}

method QualityTrack.Track(poll_secs, input, output)
{
    datahub_command(format("(create %a 1)", output));
    .TimerEvery(poll_secs, `(@self).CheckQuality(#@input, #@output));
}

method QualityTrack.constructor ()
{
    // Change the poll_secs and the two point names here. The #$$ in front
    // of the point name is necessary.
    .Track(1, #$$default:my_real_opc_point, #$$default:my_synthetic_trigger);

    // Add more calls to .Track() here
}

ApplicationSingleton (QualityTrack);
```