Formatting MQ Events as JSON

[Mark E Taylor](javascript:;" \o "Mark E Taylor) |Nov 7 2016 Updated

IBM MQ has always been able to generate event messages when something "interesting" has occurred in the queue manager. These events could be showing that a [queue is full](http://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.0.0/com.ibm.mq.mon.doc/q036320_.htm?view=kc), or that there has been an [authorisation failure](http://www.ibm.com/support/knowledgecenter/en/SSFKSJ_8.0.0/com.ibm.mq.mon.doc/q036160_.htm?view=kc); someone neededing an audit trail might want to capture the [command](http://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.0.0/com.ibm.mq.mon.doc/q036520_.htm?view=kc) and [configuration](http://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.0.0/com.ibm.mq.mon.doc/q036480_.htm?view=kc) events. These events are written as MQ messages to well-known queues, using PCF structures which can be decoded to give the full description of the event.

**The amqsevt sample program**

IBM MQ V8.0.0.4 included a new sample program, [amqsevt](http://www.ibm.com/support/knowledgecenter/en/SSFKSJ_8.0.0/com.ibm.mq.mon.doc/q036590_.htm?view=kc" \t "_blank), designed to format these event messages. That fixpack contained both the executable program and source code for it. You can see more about the formatter in this [video](https://youtu.be/FGb_XMmdLB0?t=78). Here is an event as printed by the program:

**\*\*\*\* Message #7 (236 Bytes) on Queue SYSTEM.ADMIN.PERFM.EVENT \*\*\*\***

**Event Type                       : Perfm Event [45]**

**Reason                           : Queue Full [2053]**

**Event created                    : 2016/10/21 07:11:28.58 GMT**

**Queue Mgr Name                 : V9000\_A**

**Base Object Name               : FULLEVT**

**Time Since Reset               : 0**

**High Queue Depth               : 4**

**Msg Enq Count                  : 0**

**Msg Deq Count                  : 0**

Displaying events in this more readable, English-like, style can be very useful for administrators who want to see what is going on, without using a more formal monitoring product. However, there are times when a format more sutitable for programmatic  processing are needed.

**A modified version for JSON**

The common MQ management and monitoring tools such as Omegamon are all able to decode the PCF messages, and take appropriate actions or generate alerts from these events. But many customers now want to integrate these events with other tools, that perhaps do not have an MQ-specific component or heritage. And so I've produced a modified version of the amqsevt program that can print the events in JSON, which is a simple format, but one which is capable of being parsed and searched with a variety of tools.

This example shows the same event as above, but formatted as JSON:

**{**

**"eventSource" : {**

**"objectName" : "SYSTEM.ADMIN.PERFM.EVENT",**

**"objectType" : "Queue"**

**},**

**"eventType" : {**

**"name" : "Perfm Event",**

**"value" : 45**

**},**

**"eventReason" : {**

**"name" : "Queue Full",**

**"value" : 2053**

**},**

**"eventCreation" : "2016/10/21 07:11:28.58 GMT",**

**"eventData" : {**

**"queueMgrName" : "V9000\_A",**

**"baseObjectName" : "FULLEVT",**

**"timeSinceReset" : 0,**

**"highQueueDepth" : 4,**

**"msgEnqCount" : 0,**

**"msgDeqCount" : 0**

**}**

**}**

**Using the program**

The modified amqsevt has an extra command line option, "-j" to indicate that the output should be in JSON. When selected, some of the status (eg program starting or ending) messages from the program are disabled, so that the only information sent to stdout are the JSON events themselves. When needed, error messages are now sent to stderr. That means that the standard output from the program can be sent directly to any JSON-aware program without needing further filtering. There is a blank line between each event if you want to split output at a convenient point.

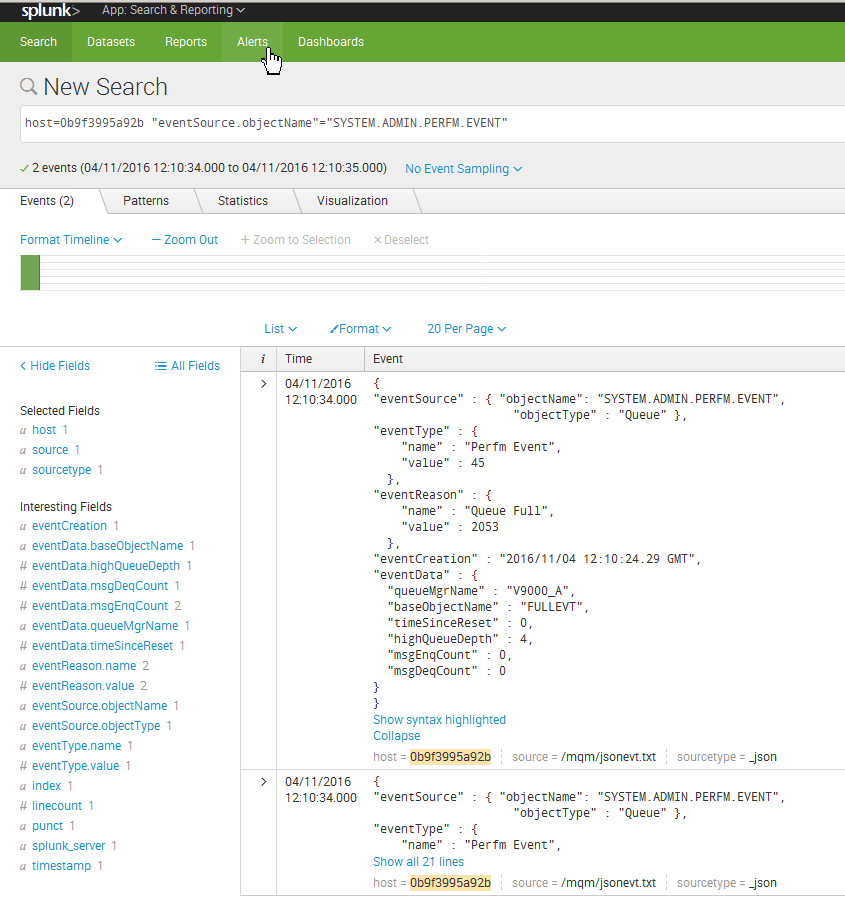
This example uses the [jq](https://stedolan.github.io/jq/download/" \t "_blank) command, which is designed to filter JSON, to find which queues have hit their full depth. It reads all the events from the queue manager and waits 2 seconds for any further input (the -w parameter):

**$ amqsevt -m QM1 -j -w 2 | jq 'select(.eventReason == 2053) | .eventData.baseObjectName '  
"FULLEVT"**

**$**

Note that the output includes the quotes around the queue name; any further processing may want to remove that. The "'-r" option to jq will do that for you.

In this next example, I redirected the output to a file, and then let the [Splunk](https://www.splunk.com/) monitoring product read it. Then I applied a filter to look for all events from the PERFM event queue:

[](https://www.ibm.com/developerworks/community/blogs/messaging/resource/BLOGS_UPLOADED_IMAGES/splunk.png)

**Where to get the program**

Source code for the program can be downloaded from this [gist](https://gist.github.com/ibmmqmet/fabd57f4ff5c6e1b8d78284b2bc65f9e). There is a "download zip" button on the page to make it easy. The program can be compiled like any of the other sample programs. You just have to ensure you link with the threaded versions of the MQI libraries (libmqm\_r on platforms where there are separate threaded and non-threaded versions).

**Feedback**

I'd be very interested in how useful this program is, and any further examples of using it. Please leave comments here or on the [github gist](https://gist.github.com/ibmmqmet/fabd57f4ff5c6e1b8d78284b2bc65f9e" \t "_blank) page.