Creating an Event Console with Automation for z/VM and Linux

This document can be found on the web at www.ibm.com/support/techdocs
Search for author’s name under the category of “White Papers”.

Version Date: November 2011

IBM Advanced Technical Support

Tracy Dean
Brand Manager z/VM Tools
tld1@us.ibm.com

&

Mike Sine
Consulting I/T Specialist
sine@us.ibm.com

© IBM Corporation 2011
Special Notices

This document is a collaboration between IBM Advanced Technical Skills and the IBM Software Group organizations and reflects their understanding of IBM Operations Manager for z/VM. It was produced and reviewed by the members of the IBM Advanced Technical Support organization. This document is presented “As-Is” and IBM does not assume responsibility for the statements expressed herein. It reflects the opinions of the IBM Advanced Technical Support organization. These opinions are based on the authors’ experiences. If you have questions about the contents of this document, please contact the authors at tld1@us.ibm.com or sine@us.ibm.com.

Trademarks

The following are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

IBM, the IBM logo, Candle, DB2, developerWorks, iSeries, Passport Advantage, pSeries, Redbooks, Tivoli Enterprise Console, WebSphere, z/OS, xSeries, zSeries, System z, z/VM, AIX.

A full list of U.S. trademarks owned by IBM may be found at http://www.ibm.com/legal/copytrade.shtml.

NetView, Tivoli and TME are registered trademarks and TME Enterprise is a trademark of Tivoli Systems, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, Internet Explorer, and the Windows logo are registered trademarks of Microsoft Corporation in the United States and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

Intel and Pentium are registered trademarks and MMX, Pentium II Xeon and Pentium III Xeon are trademarks of Intel Corporation in the United States and/or other countries.

Other company, product and service names may be trademarks or service marks of others.

Special thanks to the following people who contributed information to this effort:

- Val Nixon

© IBM Corporation 2011
**Introduction**

One of the many growing pains for mainframe staff associated with introducing z/VM and Linux into an existing mainframe environment is the need to provide a traditional level of event management and automation that is typical of the z/OS systems that have existed for many years. With Linux being a distributed based solution, including Linux on System z, event management and automation is often approached at the Linux virtual machine level only. While this may provide an acceptable level of Linux management, this approach is lacking in several areas:

- The z/VM hypervisor hosting all the Linux systems is omitted from this strategy.
- It is often helpful to have a single console including both z/VM and Linux events.
- Distributed only monitoring and automation events don’t fit the existing operations environment of many mainframe customers and their operations staff.
- A 3270 console that mimics the traditional z/OS consoles provides a level of event management familiar to most operations staff.

This paper will introduce IBM’s Operations Manager for z/VM and demonstrate some of the ways this management tool can be used to provide event management and automation in the structure and framework familiar to most z/OS system programmers and operations staff.

It is assumed that the reader has a working knowledge of Operations Manager for z/VM. Not all functions and features of Operations Manager for z/VM will be reviewed in detail. For more information on Operations Manager for z/VM, go to: http://www.ibm.com/software/sysmgmt/zvm/operations/

**System Console**

Most z/OS customers provide a centralized management console in their operations center. This is often the system console enhanced with products like IBM Tivoli NetView for z/OS to highlight messages, automate actions associated with known messages, and suppress messages. Highlighted and held messages are designed to grab the operator’s attention. Suppressed messages are designed to take away the distraction of messages determined to be non-essential to operational support. Most operations staff is accustomed to this type of message monitoring and quickly adapts to the look and feel. While the z/VM OPERATOR user ID is similar to a systems console, it may not be appropriate to suppress messages on the OPERATOR user ID. Additionally, providing direct access to the OPERATOR console for those other than system support is often not desired. Creating a custom console for the operations staff with appropriate authorization, message attributes, and automation often provides the perfect console for operations staff in a manner that they find very familiar.

*Note: Throughout this document the following terms will be used to refer to a virtual machine running under z/VM:*

- Virtual Machine
- Guest
- User ID

© IBM Corporation 2011
Collecting Console Messages across z/VM and Linux

Operations Manager for z/VM is defined as the secondary user of a z/VM virtual machine allowing collection of its console messages. Virtual machines of z/VM include the CMS guests (such as TCPIP, DIRMAINT, etc), as well as Linux, z/VSE, or z/OS guest consoles (if these operating systems are running as virtual machines under z/VM).

With IBM Operations Manager for z/VM you can:
- define rules and monitors that specify what problems or types of issues to look for among your z/VM and Linux on System z consoles,
- define actions that specify what should be done in response to a specific situation that requires intervention,
- create schedules that will execute a defined action at a specific time,
- setup automatic sessions that perform specific tasks,
- monitor and interact with a virtual machine console as if logged onto the virtual machine account, and
- monitor and manage spool usage and spool files.

We will make use of these features and functions of Operations Manager for z/VM to create an event console for z/VM and Linux that is similar to what is typically found in most z/OS Operation Centers.

Create a CMS guest to be the z/VM and Linux Event Console:

As mentioned above, it is often not desirable to provide direct access to the OPERATOR user ID of z/VM for regular console monitoring, as it may have more authorization than desired for the operations staff. Additionally, it will not necessarily have the specific messages associated with guest machine consoles. For our event console, a new CMS guest will be created to be the console on which all desired messages will be routed and viewed. The console for the z/VM and Linux messages will be a standard z/VM CMS guest user ID. This CMS user ID will only get the permissions appropriate for the operations staff (in our example privilege class G). The user ID will be named OPER8. There is nothing significant about the name. It can be any name desired that isn’t already used on the z/VM system and meets the z/VM naming conventions. The CP directory entry for OPER8 is shown below in Figure 1.
Operations Manager for z/VM will receive all console messages from all z/VM guest systems (CMS, Linux, etc) that define Operations Manager for z/VM as the secondary user. Rules can then be written to invoke an action for a subset of those messages that are deemed critical. The rules will filter out only those messages desired for forwarding to alert the operations staff. The action would be to send a message to OPER8 and apply attributes like highlighting and/or holding the message, re-wording the message, etc. Notice that the console of OPER8 has the Operations Manager for z/VM user ID, OPMGRM1 (Figure 1), defined as a secondary user. This allows Operations Manager to monitor the console of OPER8 as well, which is how attribute processing will be applied to the console messages. Operations staff can then use Operations Manager for viewing the console of OPER8.

Routing messages to OPER8

Using the Operations Manager for z/VM configuration, rules can be defined to look for critical messages to be forwarded to OPER8 (filter stage), and have attributes applied to them for viewing by operations staff (attribute stage). The following example creates a filter for a specific console message to be forwarded to OPER8 where the message will be highlighted and held on the OPER8 console.

FILTER STAGE:
The first stage of processing is to determine if the console message received is one appropriate for forwarding to OPER8. This will be referred to as the “filter stage”. Once a message meets the filter criteria via an Operations Manager for z/VM rule, an action will be defined to send the message to OPER8. The message can be sent in its original or modified form. For example, the following rule named ABEND, selects all messages that have the word “abend” and do not have the word “remote” in the text.

Note: The rule must exclude the messages on OPER8 to prevent recursively triggering the rule. Remember OPER8 is also monitored by Operations Manager for z/VM and is thus subject to its rule processing.

Any message meeting this rule’s criteria will invoke the action named “MSGOPER8”. The action MSGOPER8 issues the CP MSGNOH command to OPER8. Included in the MSGNOH command are the values of the Operations Manager &U and &T substitution variables. The &U variable passes the user ID upon which the original message appeared. The &T variable passes the message text that matched the rule (i.e. the message that met the filter criteria). This filter
The filter stage is the first phase of processing this message text. If the message is to be modified, the &T variable will be replaced or amended with the new text.

* 
DEFRULE NAME(ABEND),+ 
   MATCH(*abend*),+ 
   EXCLUDE(*remote*),+ 
   EXUSER(OPER8),+ 
   ACTION(MSGOPER8) 
* 
DEFACTN NAME(MSGOPER8),+ 
   COMMAND(CP MSGNOH OPER8 &U : &T),+ 
   OUTPUT(LOG),+ 
   ENV(LVM) 
* 
The effects of the filter stage are shown in the Figures 2 and 3 below. Figure 2 shows a test ABEND message, “this is a test abend message”, being sent to the DIRMAINT console which is monitored by Operations Manager for z/VM.

![Figure 2](image-url)
Figure 3 is an Operations Manager for z/VM VIEWLOG display showing action MSGOPER8 being invoked and successfully forwarding the message (prepended with the DIRMAINT user ID) to console OPER8. See the area inside the red outline box.

While this filter example is simplistic to make it easier to demonstrate in this document, it is important to note advanced wildcard and character position comparisons are available as one would expect in Operations Manager for z/VM. Please see the Operations Manager for z/VM Administration Guide for complete details about the DEFRULE statement.

**ATTRIBUTE STAGE:**
The second phase of processing is to apply input actions to the messages forwarded to OPER8. This is the attribute stage. These input actions will help to draw attention to the operations staff indicating the severity of the alert. This attribute stage is possible because OPER8 is a monitored console of Operations Manager for z/VM. Continuing with the message meeting the ABEND rule above, the rule ABENDHLT monitors the console of OPER8 for messages with *abend* and invokes the action HLTHOLD. Action HLTHOLD applies input actions AHI and HLD to highlight and hold the message. Specifically AHI sets the display attribute to high intensity and HLD holds the message so it doesn’t automatically scroll off the console view on the OPER8 console until it’s specifically released. Other messages will continue to scroll off the screen. The USER() keyword of DEFRULE (shown below) limits these attributes to apply only to the OPER8 console.

```
DEFRULE NAME(ABENDHLT),+
  MATCH(*abend*),+
  USER(OPER8),+
  ACTION(HLTHOLD)
```
DEFACTN NAME(HLTHOLD),+
INPUT(AHI,HLD)
*

The affects of the attribute stage are shown in Figure 4 below. Notice the message is intensified with the white highlight, and is held although additional messages continue to roll off the screen. Operations staff can remove the message by placing the cursor on the message and enter the altrcon command (default PF5 in VIEWCON unless customized in the VIEWCON PROFILE).

See the Operations Manager for z/VM Administration Guide for a detailed list of input actions available to the DEFACTN statement. Some well-known input actions are: audible alarm, blinking, high intensity, reverse video, underline, colors (e.g.: blue, cyan, green, pink, red, white, and yellow), hold, and suppress. See Figure 5 for an example of several console messages with different input actions.

Note: Throughout this document, screen captures of OPER8’s console will include “dummy” messages to simplify examples. To show a busy screen in these examples, these additional messages have been generated to show differentiation of highlighting and simulate console flow.
Figure 5

Note: It is important to understand that all messages collected by Operations Manager are saved in a log file. Any customization of messages and suppression of messages to the OPER8 console are not applied to the originating console. These customizations only affect the OPER8 console view and audit controls and message history is maintained in the Operations Manager log for each console monitored.

Manual Responses to Console Messages

Operators can respond manually based on their organization’s standard operating procedures (SOPs) via Operations Manager for z/VM. The responses can be performed on OPER8 or on the specific virtual machine reporting the message as described below.

- The operations staff can respond to the console messages on OPER8 based on OPER8’s privilege class authorization when appropriate.
- Operations Manager for z/VM can be used by authorized users to access virtual machine consoles allowing the operations staff to respond to OPER8 messages based on their Operations Manager for z/VM authorizations. This applies to all monitored consoles including Linux on System z.
Automated Responses to Console Messages

The same DEFRULE and DEFACTN statements used to filter messages and apply input actions also provide parameters to allow for automated responses triggered by known console messages. The standard benefits of automation, including immediate response, removal of human error, standardization of responses, etc., are realized for the z/VM system as well as the guest virtual machines whose consoles are being monitored. This includes the Linux guest consoles as well. Centralizing automation at the z/VM hypervisor level greatly enhances standardization and reduces duplication of automation efforts across virtual machines.

Automation Example:

For an example of automation, Operations Manager for z/VM will be used to manage the availability of a critical service virtual machine. This same capability may be used for any supported virtual machine including a Linux guest. In this scenario, the CMS virtual machine, TSTADMN2, should be running disconnected on the z/VM system at all times. If this machine were to be logged off by accident, or by a failure, Operations Manager for z/VM is to xautolog this virtual machine to make it operational and available immediately.

From the MAINT ID on this z/VM system, the “queue names” (q n) command is issued to show that TSTADMN2 is running in DSC mode, see Figure 6 below (top arrow).

![Figure 6](image)

Also shown above in Figure 6, the “force tstadmn2” command being issued to logoff tstadmn2 (bottom arrow).
Using a user ID authorized to Operations Manager for z/VM, a view of the TSTADMN2 console will show the results of the “force tstadmn2” command as well as the automation in Operations Manager for z/VM to xautolog TSTADMN2, bringing it back up immediately. The Operations Manager for z/VM view of the TSTADMN2 console (Figure 7) shows:

- The results of the “force tstadmn2” command issued by MAINT causing TSTADMN2 to logoff. This is indicated by the top red arrow.
- The bottom red arrow shows that TSTADMN2 is immediately xautolog’d and already active with the READY prompt displayed.

```
00:00:00 HCPMID6001I TIME IS 00:00:00 CDT TUESDAY 10/18/11
00:00:00
00:00:00 HCPMID6001I TIME IS 00:00:00 CDT WEDNESDAY 10/19/11
00:00:00
00:00:00 HCPMID6001I TIME IS 00:00:00 CDT THURSDAY 10/20/11
00:00:00
00:00:00 HCPMID6001I TIME IS 00:00:00 CDT FRIDAY 10/21/11
00:00:00
00:00:00 HCPMID6001I TIME IS 00:00:00 CDT SATURDAY 10/22/11
00:00:00
00:00:00 HCPMID6001I TIME IS 00:00:00 CDT SUNDAY 10/23/11
00:00:00
00:00:00 HCPMID6001I TIME IS 00:00:00 CDT MONDAY 10/24/11
00:00:00
15:57:03 CONNECT= 99:59:59 VIRTCPU= 000:00.00 TOTCPU= 000:00.00
15:57:03 LOGOFF AT 15:57:03 CDT MONDAY 10/24/11 BY MAINT
15:57:06 z/VM V5.4.0 2009-09-23 15:29
15:57:06 DMSACP732I C (198) R/O
15:57:06 Ready; T=0.01/0.01 15:57:06
PF01= SCROLL PF02= VIEWPF PF03= END PF04= HELP PF05= GOMCMD PF06= FORMAT
PF07= UP PF08= DOWN PF09= CMS CO PF10= LEFT PF11= RIGHT PF12= RECALL
```

Figure 7

The Operations Manager for z/VM configuration statements that make this possible are shown below:

DEFEMON NAME(ADMIN2),+
  TYPE(1),+
  USER(TSTADMN2),+
  ACTION(AUTOLOG1)
* 
DEFACTN NAME(AUTOLOG1),+
  COMMAND(CP SLEEP 3 SEC),+
  NEXTACTN(AUTOLOG2),+
  OUTPUT(LOG),+
  ENV(OPMGRS1)
* 
DEFACTN NAME(AUTOLOG2),+
  COMMAND(CP XAUTOLOG &3),+

© IBM Corporation 2011
The DEFEMON command defines an event monitor. Events from the CP system service *VMEVENT are captured and compared against defined event monitors. The TYPE() keyword specifies the enumerated VMEvent Type. Type(1) is the enumeration for the LOGOFF event. The DEFEMON above comes true when virtual server TSTADMN2 is logged off. Action AUTOLOG1 is then invoked by Operations Manager. Action AUTOLOG1 issues a CP SLEEP command to cause the Operations Manager action processing server (OPMGRS1) to wait for 3 seconds before issuing the actual XAUTOLOG command in the AUTLOG2 action. Without this SLEEP, the XAUTOLOG will almost always fail, indicating that the user is in LOGOFF PENDING status. By waiting 2 or 3 seconds, we allow the user ID to complete the logoff processing before autologging it.

You’ll also notice that the user ID that triggered this event (TSTADMN2 in this example) is passed to the action routine using the &3 substitution variable. This allows you to use this same action for autologging any user ID that triggers an event.

Note: The value for the ENV parameter on both DEFACTN statements must specify the same action processing server (OPMGRS1 in this example.) This is because the SLEEP and the XAUTOLOG must be issued on the same service machine in order for the XAUTLOG to be delayed. Also note that we don’t want these commands using ENV(LVM) since we don’t want the main Operations Manager server (OPMGRM1) to sleep for 3 seconds.

Using Operations Manager for z/VM, automated responses can be issued on service machines and on the originating console of the event. Therefore, automation can be performed on all virtual machines including Linux on System z. This centralized automation more easily allows combined z/VM and Linux automation steps, runs event management on the z/VM hypervisor versus duplication across all Linux for System z virtual machines, and reduces the amount of support staff needed to maintain the automation features.

**Forwarding z/VM and Linux Console Messages to an Enterprise Event manager**

Enterprise level event monitoring is an important piece of the overall systems management strategy of most advanced enterprises. Many events should be and will be handled by operations staff in the traditional realm of 3270 event monitoring. This document shows how z/VM and Linux systems can be included in this realm of management. This does not negate the need to include select events at an enterprise level. While this document is focused on providing a 3270 based operations console designed to mimic the typical z/OS console with automation enhancements, it is important to recognize the fact that many organizations use their z/OS automation to forward select events to an enterprise event manager like IBM Tivoli Netcool/OMNibus. It is important to understand that Operations Manager for z/VM’s automation may also be used to forward select events to an enterprise event manager. It is outside the scope of this document to address integration to any specific event manager. The URL below links to another document that specifically addresses integration from Operations Manager.
Manager for z/VM to IBM Tivoli Netcool/OMNibus. For those using other enterprise event managers, the document below may be used as an example and should prove helpful in integrating Operations Manager for z/VM to another vendor or in-house enterprise event solution.

Not sure why the font for the above URL is different from the rest of the document.

Summary

One of the functions in IBM Operations Manager for z/VM is to receive console messages from virtual machines under z/VM. Using the configuration commands provided with Operations Manager for z/VM, it is possible to set up an operations console similar to those most often used by operations staff for z/OS. Operations Manager for z/VM provides message attributes that draw operator attention to the console messages. Creating an event console for z/VM and Linux provides the operations staff, which may be new to z/VM and Linux, a console for event manage that is familiar in style and function to what the staff is already accustomed to for z/OS. Automating actions to known events helps to reduce human error and provide a quicker resolution to events, providing increased up time and system support. This can be very helpful in easing the growing pains associated with the z/VM and Linux implementation.