Probleemoplossing en EEM-scripts testen

Inhoud

Inleiding

Dit document beschrijft de scriptvalidatie van Embedded Event Manager (EEM) en introduceert gemeenschappelijke operationele overwegingen en foutscenario's.

Voorwaarden

Vereisten

Dit document gaat ervan uit dat de lezer al bekend is met de functie Cisco IOS/IOS XE Embedded Event Manager (EEM). Als u nog niet bekend bent met deze functie, leest u de Overzicht van EEM-functies eerst.

EEM op de Catalyst 9K familie van switches vereist de DNA-addon voor het Network Essentials-licentieniveau. Network Advantage ondersteunt EEM volledig.

Gebruikte componenten

De informatie in dit document heeft betrekking op EEM versie 4.0 zoals geïmplementeerd op de Catalyst-reeks van switches.

De informatie in dit document is gebaseerd op de apparaten in een specifieke laboratoriumomgeving. Alle apparaten die in dit document worden beschreven, hadden een opgeschoonde (standaard)configuratie. Als uw netwerk live is, moet u zorgen dat u de potentiële impact van elke opdracht begrijpt.

Achtergrondinformatie

EEM is een nuttig kenmerk wanneer het effectief wordt ingezet, maar het is belangrijk dat het EEM precies doet wat de auteur van plan is. Slecht doorgelichte scripts kunnen catastrofale problemen in de productie veroorzaken. In het beste geval speelt het script op een ongewenste manier. Dit document biedt nuttige

informatie over het testen en verifiëren van EEM met CLI-opdrachten, en verklaart ook enkele veelvoorkomende storingsscenario's en de debugs die worden gebruikt om het probleem te identificeren en te corrigeren.

EEM-validatie met opdrachten tonen

Bevestig dat de timers actief zijn

Wanneer een EEM script wordt geïmplementeerd dat wordt geactiveerd door een timer, als het script niet vuren zoals verwacht, bevestig dan dat de timer actief is en aftelt.

Neem deze EEM scripts met de naam test en test3:

<#root>

```
event manager
```

applet test

```
 authorization bypass
event timer watchdog time 60
action 0010 syslog msg "Test script running"

event manager
```

applet test3

```
 authorization bypass
event timer watchdog name test3 time 300
action 0010 syslog msg "test3 script running"
```

- Het eerste script (test) maakt gebruik van een 60 seconden durende (naamloze) watchdog timer om het script af te vuren.
- Het tweede script (test3) gebruikt een 300 seconden durende wachdog timer genaamd test3 om het script te ontslaan.

De gevormde timers en de huidige waarde van deze timers kunnen met het bevel worden bekeken **tonen de server van de statistieken van de gebeurtenismanager**.

Voorbeeld

<#root>

```
Switch#
```

show event manager statistics server

```
EEM Queue Information
                        Triggered Dropped Queue Queue Average
Client                  Events    Events  Size  Max   Run Time
-----------------------------------------------------------------------------
Call Home               5         0       0     64    0.021
EEM Applets             181       0       0     64    0.003
EEM IOS .sh Scripts     0         0       0     128   0.000
EEM Tcl Scripts         0         0       0     64    0.000
```

```
iosp_global_eem_proc    30        0       0     16    0.004
onep event service init 0         0       0     128   0.000

EEM Policy Counters
Name Value
------------------------------------------------------------------------------
```

EEM Policy Timers

```
Name                        Type
```

Time Remaining <-- EEM Countdown timer

```
------------------------------------------------------------------------------
```

_EEMinternalname0

```
         watchdog              53.328
```

<--- Unnamed timers receive an internal name - this timer is for the 'test' policy

```
_EEMinternalname1         watchdog              37.120
```

test3

```
                      watchdog              183.232
```

<--- Named timers use their configured name - this is the named timed configured for policy 'test3'

Bevestig dat triggergebeurtenissen worden uitgevoerd

Zoals besproken in de Confirma Timers zijn actieve sectie van dit document, IOS XE verhoogt de Trigged Events kolom voor de EEM Applets clientrij in de output van show event manager statistieken server telkens als een EEM applet wordt afgevuurd. Om te verifiëren dat uw EEM script werkt zoals verwacht, voer uw trigger event meerdere malen uit en controleer de output van show event manager statistics server om deze waardestijgingen te bevestigen. Als het niet, heeft uw script niet geactiveerd.

Wanneer de opdracht meerdere malen achter elkaar wordt uitgevoerd, worden de timer-waarden naar beneden geteld. Wanneer de timer nul bereikt en het script wordt uitgevoerd, wordt de geactiveerde gebeurtenis voor EEM Applets ook geteld.

<#root>

```
Switch#
```

show event manager statistics server

```
EEM Queue Information
```

Triggered

```
 Dropped Queue Queue Average
Client
```

Events

```
    Events  Size  Max   Run Time
------------------------------------------------------------------------------
Call Home               5         0       0     64    0.021

EEM Applets             183

       0       0     64    0.003

<--- "Triggered Events" column is incremented by 2 due to 2 timers firing


EEM IOS .sh Scripts     0         0       0     128   0.000
EEM Tcl Scripts         0         0       0     64    0.000
iosp_global_eem_proc    30        0       0     16    0.004
onep event service init 0         0       0     128   0.000

EEM Policy Counters
Name Value
------------------------------------------------------------------------------


EEM Policy Timers
Name                           Type

Time Remaining


------------------------------------------------------------------------------

_EEMinternalname0

              watchdog        56.215
_EEMinternalname1              watchdog        100.006

test3

                          watchdog        126.117
```

Opmerking: als dit niet gebeurt, moet u uw script onderzoeken om de ingestelde timers te controleren.

Geschiedenis van gebeurtenissen bekijken

Voor scripts die niet door timers worden geactiveerd, is de opdracht show gebeurtenismanager geschiedenisgebeurtenissen nuttig om te bevestigen dat applets worden geactiveerd zoals verwacht.

Bekijk dit EEM script:

```
<#root>

event manager

applet test_manual

 authorization bypass

event none                                                  <-- manual trigger type for testing


action 0010
```

```
syslog msg "I am a manually triggered script!" <-- message that is printed when script runs
```

Dit script wordt uitgevoerd wanneer CLI event manager run test_manual wordt uitgevoerd en drukt een syslog bericht af. Naast de output in syslog, kan de uitvoering van dit script worden geverifieerd door een beoordeling van de output van de geschiedenis van de showmanager zoals getoond:

```
<#root>

Switch#

show event manager history events


No. Job Id Proc Status   Time of Event

Event Type

           Name
1   5      Actv success  Fri Nov 6 15:45:07 2020

timer countdown


callback: Call Home process  <-- timer bases event that fired


2   18     Actv success  Mon Nov 9 14:12:33 2020   oir                 callback: Call Home process
3   19     Actv success  Mon Nov 9 14:12:40 2020   oir                 callback: Call Home process
4   20     Actv success  Fri Nov13 14:35:49 2020

none


applet: test_manual          <-- manually triggered event
```

EEM-validatie met handmatige trigger

Er zijn scenario's waar het wenselijk is om een EEM script handmatig te activeren, hetzij om de uitvoeringsstroom te testen, of om een eenmalige actie uit te voeren. Dit kan worden bereikt met een EEM script met een trigger van event none zoals aangetoond in deze output:

```
<#root>

event manager

applet test_manual

 authorization bypass
event none
action 0010 syslog msg "I am a manually triggered script!"
```

Ontsla het script handmatig met de opdracht **event manager run test_manual** vanuit de Enable prompt:

<#root>

Switch#

event manager run test_manual <-- Manually runs the script

Switch#

show log <-- Check for the log from action 10.

*Oct 26 21:24:40.762:

%HA_EM-6-LOG: test_manual: I am a manually triggered script! <-- %HA_EM logs are from EEM events. The s

Operationele overwegingen

Zorg ervoor dat EEM-scripts worden gevalideerd voordat ze in productie worden genomen. In het algemeen zijn er een paar primaire manieren waarop een script niet werkt zoals verwacht, waarvan er drie hier worden besproken.

Deze paragraaf laat zien hoe u deze 3 vaak voorkomende problemen met EEM-scripts kunt controleren:

1. CLI-opdrachtfouten: de opdracht kan niet worden geparseerd en kan daarom niet worden uitgevoerd.
2. Het script wordt te lang uitgevoerd: EEM-scripts hebben een standaard uitvoertijd van 20 seconden. Als deze tijd wordt overschreden, stopt het script voordat alle opdrachten worden uitgevoerd.
3. Het script loopt te vaak: Soms kan de trigger-gebeurtenis die door het script wordt gebruikt te vaak gebeuren, waardoor het script snel vlamt. Het is wenselijk om te controleren hoe vaak en in welke mate het script vuurt.

Probleem: CLI-opdrachten kunnen niet worden uitgevoerd

Dit voorbeeldscript bevat verschillende problemen. Het is een eenvoudige applet die de output van verscheidene showbevelen aan een tekstdossier in lokale flitsmedia toevoegt:

<#root>

event manager

applet Data_Collection

 auth bypass
event timer

watchdog time 60

action 1.0 cli command "enable"
action 1.1 cli command "show clock | append flash:DataCollection.txt"
action 1.2 cli command "show interfaces breif | append flash:DataCollection.txt"
action 1.3 cli command "show ip route | append flash:Datacollection.txt"
action 1.4 cli command "show processes cpu sorted | exclude 0.0 | append flash:DataCollection.txt"
action 1.5 cli command "show platform hardware fed switch active qos stats internal cpu policer | append
action 2.0 syslog msg "Data Capture Complete"

De applet liep succesvol, maar genereerde niet de verwachte resultaten:

<#root>

```
Switch#
show logging | in Capture


<--  Our script-generated syslog contains the string "Capture".


*Mar 11 20:40:01.941: %HA_EM-6-LOG: Data_Collection: Data Capture Complete
<-- Action 2.0 successfully ran.


Switch#
dir flash: | in .txt


<-- We only expected one .txt file, however two appear in flash:


32792 -rw- 36 Mar 11 2021 20:40:01 +00:00 DataCollection.txt
32798 -rw- 807 Mar 11 2021 20:40:01 +00:00 Datacollection.txt
Switch#
more flash:DataCollection.txt


<-- the output of our expected .txt file is empty except for the output of "show clock
"
*20:40:01.343 UTC Thu Mar 11 2021
```

Gebruik **debug embedded event manager action cloud** om te assisteren met applet verificatie.

<#root>

```
Switch#
debug embedded event manager action cli


*Mar 11 20:40:01.175: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : CTL : cli_open called.
<-- The applet is called.


*Mar 11 20:40:01.275: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : Switch>
*Mar 11 20:40:01.275: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : IN : Switch>enable
*Mar 11 20:40:01.285: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : Switch#
*Mar 11 20:40:01.285: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : IN : Switch#show clock | appenc
*Mar 11 20:40:01.396: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : Switch#
```

*Mar 11 20:40:01.396: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : IN : Switch#show interfaces bre
*Mar 11 20:40:01.507: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT :

show interfaces breif

| append flash:DataCollection.txt

<-- Here is our first problem. "brief" is misspelled, so the command does not run.

*Mar 11 20:40:01.507: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT :

^

*Mar 11 20:40:01.507: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT :

% Invalid input detected at '^' marker. <-- CLI parser failure

*Mar 11 20:40:01.507: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT :
*Mar 11 20:40:01.507: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : Switch#
*Mar 11 20:40:01.507: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : IN : Switch#

show ip route | append flash:Datacollection.txt <-- This created the second .txt file. The file name is

***Mar 11 20:40:01.618: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : Switch#**

*Mar 11 20:40:01.618: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : IN : Switch#

show processes cpu sorted | exclude 0.0 | append flash:DataCollection.txt

<-- This problem is less intuitive.

*Mar 11 20:40:01.729: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : CPU utilization for five
*Mar 11 20:40:01.729: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : PID Runtime(ms) Invoked u

the "exclude" argument reads everything beyond the pipe as the value that is to be excluded

.
*Mar 11 20:40:01.729: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : 117 57246 448028 127 0.07

A problem like this will likely not be evident in debugging

.
*Mar 11 20:40:01.729: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : 2 4488 16816 266 0.07% 0.

This underscores the importance of pre-production testing to ensure the script performs as expected

.
*Mar 11 20:40:01.729: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : 173 829 44093 18 0.07% 0.
*Mar 11 20:40:01.729: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : 205 22271 1313739 16 0.07
*Mar 11 20:40:01.729: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : 467 238 2238 106 0.07% 0.

*Mar 11 20:40:01.730: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : 81 12793 151345 84 0.07%
*Mar 11 20:40:01.730: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : 232 22894 2621198 8 0.07%
*Mar 11 20:40:01.730: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : 7 0 1 0 0.00% 0.00% 0.00%
*Mar 11 20:40:01.730: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : 6 0 1 0 0.00% 0.00% 0.00%
*Mar 11 20:40:01.730: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : 8 17 2804 6 0.00% 0.00% 0
*Mar 11 20:40:01.730: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : 9 33511 11402 2939 0.00%
*Mar 11 20:40:01.730: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : 12 0 2 0 0.00% 0.00% 0.00

```
*Mar 11 20:40:01.730: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : 10 106 1402 75 0.00% 0.00
*Mar 11 20:40:01.730: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : 14 439 42047 10 0.00% 0.0
*Mar 11 20:40:01.730: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : 11 0 1 0 0.00% 0.00% 0.00
*Mar 11 20:40:01.730: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : 16 0 1 0 0.00% 0.00% 0.00

*Mar 11 20:40:01.730: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : 5 0 1 0 0.00% 0.00% 0.00%
*Mar 11 20:40:01.730: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : 18 0 3 0 0.00% 0.00% 0.00
*Mar 11 20:40:01.730: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : CTL : 20+ lines read from cli,

*Mar 11 20:40:01.730: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : IN : Switch#

show platform hardware fed switch active qos stats internal cpu policer

 | append flash:DataCollection.txt
*Mar 11 20:40:01.941: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : show platform hardware fe

<-- Here, the syntax of the command was not properly parsed out before implementation. We are missing ar


*Mar 11 20:40:01.941: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT :

  ^  <-- missing word queue


*Mar 11 20:40:01.941: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT :

% Invalid input detected at '^' marker.        <-- CLI parser failure


*Mar 11 20:40:01.941: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT :
*Mar 11 20:40:01.941: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : OUT : Switch#
*Mar 11 20:40:01.941: %HA_EM-6-LOG: Data_Collection: Data Capture Complete

<-- The syslog from Action 2.0 writes.


*Mar 11 20:40:01.941: %HA_EM-6-LOG: Data_Collection : DEBUG(cli_lib) : : CTL : cli_close called.

<-- The applet closes out as expected after executing all configured actions.
```

Conclusie: Alle EEM-acties goed doorlichten en debugs gebruiken om te bewijzen tegen misconfiguraties en typografische fouten.

Probleem: EEM-acties duren langer dan de maximale looptijd

In dit scenario wordt een eenvoudig EEM gebruikt om met tussenpozen van 120 seconden een pakket met besturingsvlakken te verzamelen. Er worden nieuwe opnamegegevens toegevoegd aan een uitvoerbestand op lokale opslagmedia.

<#root>

```
event manager

applet Capture


event timer

watchdog time 120     <-- 120 second countdown timer
```

```
action 1.0 cli command "enable"
action 1.1 cli command "no monitor capture CPUCapture"
action 2.0 cli command "monitor capture CPUCapture control-plane in match any buffer circular"
action 2.1 cli command "monitor capture CPUCapture start"
action 3.0 wait 45
action 4.0 cli command "monitor capture CPUCapture stop"
action 4.1 cli command "show clock | append flash:CPUCapture.txt"
action 4.2 cli command "show mon cap CPUCapture buff dump | append flash:CPUCapture.txt"
```

`action 5.0 syslog msg "CPUCapture Complete - Next capture in 2 minutes"`

U kunt gemakkelijk vaststellen dat het EEM niet volgens verwachting verloopt. Controleer lokale logbestanden voor de syslog vanaf actie 5.0. Deze syslog drukt op elke succesvolle iteratie van de applet af. Het logbestand is niet afgedrukt binnen de buffer en het bestand CPUCapture.txt is niet geschreven om te flitsen:

<#root>

Switch#

`show logging | include "CPUCapture Complete"`

Switch#

`dir flash: | include CPUCapture.txt`

Schakel debugs in om te onderzoeken. De meest gebruikte debug is **debug event manager action class.** Dit hulpprogramma drukt achtereenvolgens een dialoogvenster af van de acties.

Debug uitvoer: De debug uitvoer toont met succes geroepen applet. De eerste acties lopen zonder problemen, maar de vastlegging mislukt.

<#root>

Switch#

`debug event manager action cli`

*Jan 28 22:55:54.742: %HA_EM-6-LOG: Capture : DEBUG(cli_lib) : : CTL : cli_open called.

<-- This is the initial message seen when the applet is called.

*Jan 28 22:55:54.843: %HA_EM-6-LOG: Capture : DEBUG(cli_lib) : : OUT : CoreSwitch>

The applet name can be seen within the line.

```
*Jan 28 22:55:54.843: %HA_EM-6-LOG: Capture : DEBUG(cli_lib) : : IN : CoreSwitch>enable
*Jan 28 22:55:54.854: %HA_EM-6-LOG: Capture : DEBUG(cli_lib) : : OUT : CoreSwitch#
*Jan 28 22:55:54.854: %HA_EM-6-LOG: Capture : DEBUG(cli_lib) : : IN : CoreSwitch#no monitor capture CPUC
*Jan 28 22:55:54.964: %HA_EM-6-LOG: Capture : DEBUG(cli_lib) : : OUT : Capture does not exist
*Jan 28 22:55:54.964: %HA_EM-6-LOG: Capture : DEBUG(cli_lib) : : OUT :
```

```
*Jan 28 22:55:54.964: %HA_EM-6-LOG: Capture : DEBUG(cli_lib) : : OUT : CoreSwitch#
*Jan 28 22:55:54.965: %HA_EM-6-LOG: Capture : DEBUG(cli_lib) : : IN : CoreSwitch#monitor capture CPUCapt
Jan 28 22:55:55.075: %HA_EM-6-LOG: Capture : DEBUG(cli_lib) : : OUT : CoreSwitch#
*Jan 28 22:55:55.075: %HA_EM-6-LOG: Capture : DEBUG(cli_lib) : : IN : CoreSwitch#monitor capture CPUCapt
*Jan 28 22:55:55.185: %HA_EM-6-LOG: Capture : DEBUG(cli_lib) : : OUT : Started capture point : CPUCaptur
```

<-- The applet successfully creates and starts the capture.

```
*Jan 28 22:55:55.185: %HA_EM-6-LOG: Capture : DEBUG(cli_lib) : : OUT : CoreSwitch#
*Jan 28 22:56:15.187: %HA_EM-6-LOG: Capture : DEBUG(cli_lib) : : CTL : cli_close called.
```

<-- After 20 seconds, cli_close is called and the applet begins to exit.

```
*Jan 28 22:56:15.187: fh_server: fh_io_ipc_msg: received msg FH_MSG_CALLBACK_DONE from client 27 pclient
*Jan 28 22:56:15.187: fh_io_ipc_msg: EEM callback policy Capture has ended with abnormal exit status of
```

FF

```
*Jan 28 22:56:15.187:
```

EEM policy Capture has exceeded it's elapsed time limit of 20.0 seconds <-- We are informed that the pol

```
*Jan 28 22:56:15.187: fh_io_ipc_msg: received FH_MSG_API_CLOSE client=27
*Jan 28 22:56:15.187: tty is now going through its death sequence
```

```
*Note "
```

debug event manager all

```
" is used to enable all debugs related to event manager.
```

Oplossing: EEM-beleid loopt standaard niet langer dan 20 seconden. Als het langer dan 20 seconden duurt voordat de acties binnen de EEM verlopen zijn, is de EEM niet klaar. Zorg ervoor dat de uitvoering van uw EEM voldoende is om uw applet acties te laten lopen. Configureer maxrun om een geschiktere maximale runtime waarde op te geven.

Voorbeeld

<#root>

```
event manager
```

applet Capture

```
event timer watchdog time 120
```

maxrun 60

<-- Maxrun 60 specifies the capture will run for a maximum of 60 seconds.

```
action 1.0 cli command "enable"
action 1.1 cli command "no monitor capture CPUCapture"
action 2.0 cli command "monitor capture CPUCapture control-plane in match any buffer circular"
action 2.1 cli command "monitor capture CPUCapture start"
```

```
action 3.0 wait 45
```

<-- The altered maxrun allows the capture to run for the necessary time.

```
action 4.0 cli command "monitor capture CPUCapture stop"
action 4.1 cli command "show clock | append flash:CPUCapture.txt"
action 4.2 cli command "show mon cap CPUCapture buff dump | append flash:CPUCapture.txt"
action 5.0 syslog msg "CPUCapture Complete - Next capture in 2 minutes"
```

Probleem: EEM veroorzaakt te vaak

Soms, komen verscheidene instanties van een bepaalde trekker in een korte hoeveelheid tijd voor. Dit kan leiden tot excessieve herhalingen van de applet en in het ergste geval ernstige gevolgen hebben.

Dit applet activeert op een bepaald syslog patroon, dan verzamelt toont opdrachtoutput en voegt deze output toe aan een bestand. Met name de applet vuurt als het lijnprotocol voor een geïdentificeerde interface valt:

<#root>

```
event manager
```

applet MonitorLinkFlap

```
event syslog pattern "Interface GigabitEthernet1/0/23, changed state to down"
action 1.0 cli command "enable"
action 1.1 cli command "show ip route | append flash:MonitorLinkFlap.txt "
action 2.0 cli command "show interface gig1/0/23 | append flash:MonitorLinkFlap.txt"
action 3.0 cli command "show process cpu sorted | append flash:MonitorLinkFlap.txt"
action 4.0 cli command "show platform hardware fed active fwd-asic drops exceptions | append flash:Monit
action 5.0 syslog msg "Link has flapped - Data gathered"
```

De applet vuurt telkens als de syslog wordt waargenomen. Een gebeurtenis zoals een interfaceklap kan snel in een korte tijd voorkomen.

<#root>

```
Switch#
```

sh log | in Data gathered

```
*Jan 29 04:19:06.678: %HA_EM-6-LOG: MonitorLinkFlap: Link has flapped - Data gathered
```

<-- The applet generates this syslog each time it fires.

```
*Jan 29 04:19:27.367: %HA_EM-6-LOG: MonitorLinkFlap: Link has flapped - Data gathered
*Jan 29 04:19:36.779: %HA_EM-6-LOG: MonitorLinkFlap: Link has flapped - Data gathered
*Jan 29 04:19:57.472: %HA_EM-6-LOG: MonitorLinkFlap: Link has flapped - Data gathered
*Jan 29 04:20:06.570: %HA_EM-6-LOG: MonitorLinkFlap: Link has flapped - Data gathered
*Jan 29 04:20:27.671: %HA_EM-6-LOG: MonitorLinkFlap: Link has flapped - Data gathered
*Jan 29 04:20:36.774: %HA_EM-6-LOG: MonitorLinkFlap: Link has flapped - Data gathered
*Jan 29 04:20:57.264: %HA_EM-6-LOG: MonitorLinkFlap: Link has flapped - Data gathered
```

De applet liep meerdere malen in de loop van een paar minuten, wat resulteerde in een ongewenste uitvoerbestand met vreemde gegevens. Het bestand blijft ook groter worden en blijft lokale media vullen. Dit simpele voorbeeld EEM vormt niet veel operationele bedreiging als er meerdere keren wordt gelopen, maar dit scenario kan mogelijk leiden tot een crash met meer complexe scripts.

In dit scenario zou het nuttig zijn om te beperken hoe vaak de applets wordt geactiveerd.

Oplossing: Pas een snelheidslimiet toe om te controleren hoe snel een applet werkt. Het sleutelwoord rate wordt toegevoegd aan het trigger statement en is gekoppeld aan een waarde in seconden.

Voorbeeld

<#root>

```
event manager
```

applet MonitorLinkFlap

```
event syslog pattern "Interface GigabitEthernet1/0/23, changed state to down"
```

ratelimit 60

<-- Ratelimit

specifies a minimum amount of time that must pass before the applet will again trigger.

```
action 1.0 cli command "enable"
action 1.1 cli command "show clock | append flash:MonitorLinkFlap.txt "
action 2.0 cli command "show interface gig1/0/23 | append flash:MonitorLinkFlap.txt"
action 3.0 cli command "show process cpu sorted | append flash:MonitorLinkFlap.txt"
action 4.0 cli command "show platform hardware fed active fwd-asic drops exceptions | append flash:Monit
action 5.0 syslog msg "Link has flapped - Data gathered"
```

Gerelateerde informatie

Cisco IOS ingesloten gebeurtenisbeheer 4.0

Best Practices en bruikbare scripts voor EEM

Over deze vertaling

Cisco heeft dit document vertaald via een combinatie van machine- en menselijke technologie om onze gebruikers wereldwijd ondersteuningscontent te bieden in hun eigen taal. Houd er rekening mee dat zelfs de beste machinevertaling niet net zo nauwkeurig is als die van een professionele vertaler. Cisco Systems, Inc. is niet aansprakelijk voor de nauwkeurigheid van deze vertalingen en raadt aan altijd het oorspronkelijke Engelstalige document (link) te raadplegen.