Risoluzione dei problemi e test degli script EEM

Sommario

Introduzione

In questo documento viene descritta la convalida degli script di Embedded Event Manager (EEM) e vengono presentate considerazioni operative comuni e scenari di errore.

Prerequisiti

Requisiti

in questo documento si presume che il lettore abbia già familiarità con la funzionalità Cisco IOS/IOS XE Embedded Event Manager (EEM). Se non si ha familiarità con questa funzione, leggere la Panoramica delle funzionalità EEM prima.

EEM sulla famiglia di switch Catalyst 9K richiede il componente aggiuntivo DNA per il livello di licenza Network Essentials. Network Advantage supporta completamente EEM.

Componenti usati

Le informazioni di questo documento fanno riferimento alla versione 4.0 di EEM implementata sulla famiglia di switch Catalyst.

Le informazioni discusse in questo documento fanno riferimento a dispositivi usati in uno specifico ambiente di emulazione. Su tutti i dispositivi menzionati nel documento la configurazione è stata

ripristinata ai valori predefiniti. Se la rete è operativa, valutare attentamente eventuali conseguenze derivanti dall'uso dei comandi.

Premesse

L'EEM è una funzionalità utile se implementata in modo efficace, ma è importante garantire che l'EEM faccia esattamente ciò che l'autore intende. Script scarsamente controllati possono causare problemi catastrofici nella produzione. Nel migliore dei casi, lo script si comporta in modo indesiderato. Questo documento offre informazioni utili su come testare e verificare EEM con i comandi show della CLI e spiega alcuni scenari di errore comuni e i debug utilizzati per identificare e risolvere il problema.

Convalida EEM con comandi Show

Conferma attivazione timer

Quando viene distribuito uno script EEM attivato da un timer, se lo script non viene attivato come previsto, verificare che il timer sia attivo e procedere al conto alla rovescia.

Considerare gli script EEM seguenti denominati rispettivamente test e 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"
```

- Il primo script (test) utilizza un timer di watchdog da 60 secondi (senza nome) per attivare lo script.
- Il secondo script (test3) utilizza un timer di watchdog da 300 secondi denominato test3 per attivare lo script.

I timer configurati e il valore corrente di questi timer possono essere visualizzati con il comando show event manager statistics server.

Esempio

```
<#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'
```

Attivazione degli eventi trigger di conferma

Come illustrato nella sezione Timer di conferma attivi di questo documento, IOS XE incrementa la colonna Eventi attivati per la riga del client EEM Applets nell'output di show event manager statistics server ogni volta che viene attivata un'applet EEM. Per verificare che lo script EEM funzioni come previsto, eseguire più volte l'evento trigger ed esaminare l'output di show event manager statistics server per confermare gli incrementi di questo valore. In caso contrario, lo script non viene attivato.

Quando il comando viene eseguito più volte in sequenza, i valori del timer da contare alla rovescia. Quando il timer raggiunge lo zero e lo script viene eseguito, viene conteggiato anche il

numero di eventi attivati per le applet EEM.

<#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
```

Nota: in caso contrario, esaminare lo script per verificare i timer configurati.

Rivedi cronologia eventi

Per gli script non attivati dai timer, il comando show event manager history events è utile per confermare che le applet vengono attivate come previsto.

Considerare questo script EEM:

```
<#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
```

Questo script viene eseguito quando viene eseguito test_manual dal gestore degli eventi CLI e viene stampato un messaggio syslog. Oltre all'output in syslog, l'esecuzione di questo script può essere verificata da una revisione dell'output degli eventi della cronologia di show event manager, come mostrato di seguito:

```
<#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
```

Convalida EEM con trigger manuale

In alcuni scenari è consigliabile attivare manualmente uno script EEM, sia per verificare il flusso di esecuzione che per eseguire un'azione una tantum. A tale scopo, è possibile utilizzare uno script EEM con un trigger di evento none, come illustrato in questo output:

```
<#root>

event manager

applet test_manual

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

Attivare manualmente lo script con il comando event manager run test_manual dal prompt di abilitazione:

```
<#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
```

Considerazioni operative

Verificare che gli script EEM siano convalidati prima di essere utilizzati nella produzione. In generale, ci sono alcuni modi primari in cui uno script non funziona come previsto, tre dei quali sono discussi qui.

Questa sezione illustra come verificare la presenza di questi 3 problemi comuni con gli script EEM:

1. Errori del comando CLI: il comando non può essere analizzato e quindi non può essere eseguito.
2. Lo script viene eseguito troppo a lungo: il limite di tempo di esecuzione predefinito per gli script EEM è di 20 secondi. Se questo tempo viene superato, lo script si arresta prima

dell'esecuzione di tutti i comandi.

3. Lo script viene eseguito troppo spesso: a volte l'evento trigger utilizzato dallo script può verificarsi troppo di frequente, il che provoca la rapida attivazione dello script. È consigliabile controllare la frequenza e la velocità di esecuzione dello script.

Problema: impossibile eseguire i comandi CLI

Questo script di esempio contiene diversi problemi. Si tratta di una semplice applet che aggiunge l'output di diversi comandi show a un file di testo nel supporto flash locale:

<#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 | appen
action 2.0 syslog msg  "Data Capture Complete"
```

L'applet è stata eseguita correttamente, ma non ha generato i risultati previsti:

<#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
```

Utilizzare l'azione cli di debug embedded event manager per agevolare la verifica delle applet.

```
<#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 | appen
*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 br
*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

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 0.0

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

*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% 0.0
*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 f

<-- 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.

Conclusione: esaminare in modo corretto tutte le azioni EEM e utilizzare i debug per evitare errori di configurazione e di battitura.

Problema: le azioni EEM richiedono più tempo del tempo di esecuzione massimo

In questo scenario, un EEM semplice viene utilizzato per raccogliere le acquisizioni dei pacchetti del control plane a intervalli di 120 secondi. Aggiunge i nuovi dati acquisiti a un file di output che si trova nel supporto di archiviazione locale.

<#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"

È possibile determinare facilmente se l'EEM non viene completato come previsto. Controllare i log locali per il syslog dall'azione 5.0. Il syslog viene stampato per ogni iterazione corretta dell'applet. Il log non è stato stampato nel buffer e il file CPUCapture.txt non è stato scritto su flash:

<#root>

```
Switch#
```

```
show logging | include "CPUCapture Complete"


Switch#
dir flash: | include CPUCapture.txt
```

Abilitare i debug da analizzare. Il comando di debug più comunemente usato è debug event manager action cli. Questa utilità stampa una finestra di dialogo delle azioni in sequenza.

Output di debug: l'output di debug visualizza l'applet chiamata correttamente. Le azioni iniziali vengono eseguite senza problemi, ma l'acquisizione non riesce a concludersi.

<#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 CPU
*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 CPUCap
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 CPUCap
*Jan 28 22:55:55.185: %HA_EM-6-LOG: Capture : DEBUG(cli_lib) : : OUT : Started capture point : CPUCaptu
<-- 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 pclien
*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.

Soluzione: per impostazione predefinita, i criteri EEM non vengono eseguiti per più di 20 secondi. Se l'esecuzione delle azioni all'interno dell'EEM richiede più di 20 secondi, l'EEM non viene completato. Assicurarsi che il runtime dell'EEM sia sufficiente per consentire l'esecuzione delle azioni applet. Configurare maxrun per specificare un valore di runtime massimo più appropriato.

Esempio

<#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"
```

Problema: attivazione EEM troppo spesso

A volte, diverse istanze di un determinato trigger si verificano in un breve lasso di tempo. Ciò potrebbe portare a ripetizioni eccessive dell'applet e avere gravi conseguenze nel peggiore dei casi.

Questa applet viene attivata su un particolare modello syslog, quindi raccoglie l'output del comando show e lo aggiunge a un file. In particolare, l'applet viene attivata quando il protocollo di

linea di un'interfaccia identificata viene interrotto:

<#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:Moni
action 5.0 syslog msg "Link has flapped - Data gathered"
```

L'applet viene attivata ogni volta che viene osservato il syslog. Un evento come un flap dell'interfaccia può verificarsi rapidamente in un breve periodo di tempo.

<#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
```

L'applet è stata eseguita numerose volte nel corso di alcuni minuti, il che ha determinato un file di output indesiderato con dati estranei. Le dimensioni del file aumentano continuamente e il file continua a riempire i supporti locali. Questo semplice EEM di esempio non rappresenta una grave minaccia operativa se viene eseguito ripetutamente, ma questo scenario può potenzialmente portare a un arresto anomalo con script più complessi.

In questo scenario, sarebbe utile limitare la frequenza di attivazione dell'applet.

Soluzione: applicare un limite di velocità per controllare la velocità di esecuzione di un'applet. La parola chiave ratelimit viene aggiunta all'istruzione trigger ed è associata a un valore in secondi.

Esempio

<#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:Moni
action 5.0 syslog msg "Link has flapped - Data gathered"
```

Informazioni correlate

Cisco IOS Embedded Event Manager 4.0

Procedure ottimali e script utili per EEM

Informazioni su questa traduzione

Cisco ha tradotto questo documento utilizzando una combinazione di tecnologie automatiche e umane per offrire ai nostri utenti in tutto il mondo contenuti di supporto nella propria lingua. Si noti che anche la migliore traduzione automatica non sarà mai accurata come quella fornita da un traduttore professionista. Cisco Systems, Inc. non si assume alcuna responsabilità per l’accuratezza di queste traduzioni e consiglia di consultare sempre il documento originale in inglese (disponibile al link fornito).