Solucionar problemas e testar scripts EEM

Contents

Introdução

Este documento descreve a validação de script do Embedded Event Manager (EEM) e apresenta considerações operacionais comuns e cenários de falha.

Pré-requisitos

Requisitos

Este documento supõe que o leitor já esteja familiarizado com o recurso Embedded Event Manager (EEM) do Cisco IOS/IOS XE. Se você ainda não estiver familiarizado com este recurso, leia a [Visão geral dos recursos do EEM](#) primeiro.

O EEM na família de switches Catalyst 9K requer o complemento DNA para o nível de licença do Network Essentials. O Network Advantage oferece suporte total ao EEM.

Componentes Utilizados

As informações neste documento relacionam-se ao EEM versão 4.0 conforme implementado na família de switches Catalyst.

As informações neste documento foram criadas a partir de dispositivos em um ambiente de laboratório específico. Todos os dispositivos utilizados neste documento foram iniciados com uma configuração (padrão) inicial. Se a rede estiver ativa, certifique-se de que você entenda o impacto potencial de qualquer comando.

Informações de Apoio

O EEM é um recurso útil quando implantado com eficácia, mas é importante garantir que o EEM faça

exatamente o que o autor pretende. Scripts mal verificados podem levar a problemas catastróficos na produção. Na melhor das hipóteses, o script é executado de maneira indesejada. Este documento fornece informações úteis sobre como testar e verificar o EEM com comandos show da CLI e também explica alguns cenários de falha comuns e as depurações usadas para identificar e corrigir o problema.

Validação do EEM com comandos show

Confirmar se os temporizadores estão ativos

Quando um script EEM implantado é acionado por um temporizador, se o script não for acionado como esperado, confirme se o temporizador está ativo e faz uma contagem regressiva.

Considere estes scripts EEM nomeados test e test3, respectivamente:

<#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"
```

- O primeiro script (teste) usa um temporizador watchdog de 60 segundos (sem nome) para acionar o script.
- O segundo script (test3) usa um temporizador watchdog de 300 segundos chamado test3 para acionar o script.

Os temporizadores configurados e o valor atual desses temporizadores podem ser visualizados com o comando **show event manager statistics server**.

Exemplo

<#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'
```

Confirme se os eventos de disparo estão sendo disparados

Conforme discutido na seção Confirmar que os temporizadores estão ativos deste documento, o IOS XE incrementa a coluna Eventos disparados para a linha de cliente de miniaplicativos EEM na saída do show event manager statistics server toda vez que um miniaplicativo EEM é acionado. Para verificar se o script EEM funciona conforme o esperado, execute o evento de acionamento várias vezes e examine a saída do comando show event manager statistics server para confirmar os incrementos desse valor. Caso contrário, o script não será disparado.

Quando o comando é executado várias vezes em sequência, os valores do temporizador são contados abaixo. Quando o temporizador chega a zero e o script é executado, a contagem de eventos disparados para os miniaplicativos EEM também é contada.

```
<#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
```

Observação: se isso não ocorrer, investigue seu script para verificar os temporizadores configurados.

Revisar histórico de eventos

Para scripts que não são acionados por temporizadores, o comando show event manager history events é útil para confirmar se os miniaplicativos são acionados conforme o esperado.

Considere este 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

Este script é executado quando o gerenciador de eventos CLI run test_manual é executado e imprime uma mensagem syslog. Além da saída no syslog, a execução desse script pode ser verificada por uma revisão da saída de show event manager history events conforme mostrado:

<#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

Validação do EEM com disparador manual

Há situações em que é desejável disparar manualmente um script EEM, seja para testar o fluxo de execução ou para executar uma ação única. Isso pode ser realizado com um script EEM com um disparador de evento none, conforme demonstrado nesta saída:

<#root>

```
event manager
```

applet test_manual

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

Dispare manualmente o script com o comando **event manager run test_manual** do prompt de ativação:

```
<#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
```

Considerações operacionais

Certifique-se de que os scripts EEM sejam validados antes de usá-los na produção. Em geral, há algumas maneiras principais de um script não funcionar como esperado, três das quais são discutidas aqui.

Esta seção mostra como verificar estes 3 problemas comuns com scripts EEM:

1. Falhas no comando CLI: o comando não consegue analisar e, portanto, não é executado.
2. O script é executado por muito tempo: os scripts EEM têm um limite de tempo de execução padrão de 20 segundos. Se esse tempo for excedido, o script será interrompido antes da execução de todos os comandos.
3. O script é executado com muita frequência: às vezes, o evento de acionamento usado pelo script pode ocorrer com muita frequência, o que faz com que o script seja acionado rapidamente. É desejável controlar com que frequência e em que velocidade o script é acionado.

Problema: falha na execução dos comandos CLI

Este script de exemplo contém vários problemas. É um applet simples que anexa a saída de vários comandos show a um arquivo de texto na mídia flash local:

```
<#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"
```

O miniaplicativo foi executado com êxito, mas não gerou os resultados esperados:

```
<#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
```

Use **debug embedded event manager action cli** para ajudar na verificação do 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 | append
*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.

Conclusão: analise corretamente todas as ações do EEM e use depurações para comprovar contra erros de configuração e erros tipográficos.

Problema: as ações do EEM levam mais tempo que o tempo de execução máximo

Neste cenário, um EEM simples é usado para coletar capturas de pacotes de plano de controle em intervalos de 120 segundos. Ele acrescenta novos dados de captura a um arquivo de saída localizado na mídia de armazenamento local.

<#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"
```

Você pode facilmente determinar que o EEM não é concluído conforme esperado. Verifique os logs locais do syslog a partir da ação 5.0. Este syslog é impresso em cada iteração bem-sucedida do applet. O log não foi impresso dentro do buffer e o arquivo CPUCapture.txt não foi gravado na memória flash:

```
<#root>
Switch#
show logging | include "CPUCapture Complete"


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

Habilite depurações para investigar. A depuração mais comumente usada é a **cli de ação do gerenciador de eventos de depuração.** Este utilitário imprime um diálogo das ações em sequência.

Saída de depuração: a saída de depuração mostra o miniaplicativo chamado com êxito. As ações iniciais são executadas sem problemas, mas a captura não é concluída.

```
<#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.
```

Solução: por padrão, as políticas EEM não são executadas por mais de 20 segundos. Se as ações no EEM demorarem mais de 20 segundos para serem executadas, o EEM não será concluído. Certifique-se de que o tempo de execução do EEM seja suficiente para permitir que as ações do applet sejam executadas. Configure maxrun para especificar um valor de tempo de execução máximo mais apropriado.

Exemplo

<#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: o EEM é acionado com muita frequência

Às vezes, várias instâncias de um determinado gatilho ocorrem em um curto período de tempo. Isso pode levar a iterações excessivas do applet e ter consequências sérias no pior caso.

Esse applet é acionado em um padrão syslog específico, reúne a saída do comando show e anexa essa saída a um arquivo. Especificamente, o applet é acionado quando o protocolo de linha cai para uma interface identificada:

<#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"
```

O applet é acionado toda vez que o syslog é observado. Um evento como uma oscilação de interface pode ocorrer rapidamente em um curto período de 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
```

O applet foi executado várias vezes ao longo de alguns minutos, o que resultou em um arquivo de saída indesejável com dados estranhos. O arquivo também continua a aumentar de tamanho e a ocupar mídia local. Esse exemplo simples de EEM não representa muita ameaça operacional se executado repetidamente, mas esse cenário potencialmente leva a um travamento com scripts mais complexos.

Nesse cenário, seria útil limitar a frequência com que o applet é acionado.

Solução: aplique um limite de taxa para controlar a rapidez de execução de um miniaplicativo. A palavra-chave ratelimit é anexada à instrução trigger e é associada a um valor em segundos.

Exemplo

<#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"
```

Informações Relacionadas

Cisco IOS Embedded Event Manager 4.0

Práticas recomendadas e scripts úteis para EEM

Sobre esta tradução
A Cisco traduziu este documento com a ajuda de tecnologias de tradução automática e humana para oferecer conteúdo de suporte aos seus usuários no seu próprio idioma, independentemente da localização.

Observe que mesmo a melhor tradução automática não será tão precisa quanto as realizadas por um tradutor profissional.

A Cisco Systems, Inc. não se responsabiliza pela precisão destas traduções e recomenda que o documento original em inglês (link fornecido) seja sempre consultado.