Dépannage et test des scripts EEM

Table des matières

Introduction

Ce document décrit la validation des scripts du gestionnaire d'événements intégré (EEM) et présente des considérations opérationnelles courantes et des scénarios d'échec.

Conditions préalables

Exigences

Ce document suppose que le lecteur est déjà familiarisé avec la fonctionnalité Cisco IOS/IOS XE Embedded Event Manager (EEM). Si vous n'êtes pas déjà familiarisé avec cette fonctionnalité, veuillez lire la Présentation des fonctionnalités EEM d'abord.

EEM sur la gamme de commutateurs Catalyst 9K nécessite le module complémentaire DNA pour le niveau de licence Network Essentials. Network Advantage prend entièrement en charge EEM.

Composants utilisés

Les informations contenues dans ce document concernent la version 4.0 d'EEM telle qu'elle est implémentée sur la famille de commutateurs Catalyst.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. Si votre réseau

est en ligne, assurez-vous de bien comprendre l’incidence possible des commandes.

Informations générales

L'ESEE est utile lorsqu'elle est déployée efficacement, mais il est important de s'assurer qu'elle fait exactement ce que l'auteur veut. Des scripts mal vérifiés peuvent entraîner des problèmes catastrophiques en production. Au mieux, le script fonctionne d'une manière non souhaitée. Ce document fournit des informations utiles sur la façon de tester et de vérifier EEM avec les commandes show de l'interface de ligne de commande, et explique également certains scénarios d'échec courants et les débogages utilisés pour identifier et corriger le problème.

Validation EEM avec commandes show

Confirmer que les minuteurs sont actifs

Lorsqu'un script EEM est déployé et déclenché par un minuteur, si le script ne se déclenche pas comme prévu, vérifiez que le minuteur est actif et compte à rebours.

Considérez ces scripts EEM nommés respectivement test et 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"
```

- Le premier script (test) utilise un minuteur de surveillance de 60 secondes (sans nom) pour déclencher le script.
- Le deuxième script (test3) utilise un minuteur de surveillance de 300 secondes nommé test3 pour déclencher le script.

Les compteurs configurés et la valeur actuelle de ces compteurs peuvent être affichés avec la commande show event manager statistics server.

Exemple

<#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'
```

Confirmer le déclenchement des événements déclencheurs

Comme indiqué dans la section Confirmer que les minuteurs sont actifs de ce document, IOS XE incrémente la colonne Evénements déclenchés pour la ligne du client Applets EEM dans le résultat de show event manager statistics server chaque fois qu'une applet EEM est déclenchée. Pour vérifier que votre script EEM fonctionne comme prévu, exécutez votre événement déclencheur plusieurs fois et examinez le résultat de show event manager statistics server pour confirmer ces incréments de valeur. Dans le cas contraire, votre script ne s'est pas déclenché.

Lorsque la commande est exécutée plusieurs fois dans l'ordre, le minuteur prend la valeur count down. Lorsque le minuteur atteint zéro et que le script s'exécute, le nombre d'événements déclenchés pour les applets EEM est également pris en compte.

```
<#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
```

Remarque : si ce n'est pas le cas, examinez votre script pour vérifier les minuteurs configurés.

Historique des événements

Pour les scripts qui ne sont pas déclenchés par des minuteurs, la commande show event manager history events est utile pour confirmer que les applets sont déclenchées comme prévu.

Considérez ce 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
```

Ce script s'exécute lorsque le gestionnaire d'événements CLI run test_manual est exécuté et imprime un message syslog. Outre le résultat dans syslog, l'exécution de ce script peut être vérifiée par un examen du résultat de show event manager history events comme indiqué :

```
<#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
```

Validation EEM avec déclencheur manuel

Dans certains cas, il est souhaitable de déclencher manuellement un script EEM, soit pour tester le flux d'exécution, soit pour effectuer une action ponctuelle. Ceci peut être accompli avec un script EEM avec un déclencheur d'événement none comme démontré dans ce résultat :

```
<#root>

event manager

applet test_manual

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

Lancez manuellement le script avec la commande event manager run test_manual à partir de l'invite enable :

```
<#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
```

Considérations opérationnelles

S'assurer que les scripts EEM sont validés avant leur utilisation en production. En général, il y a quelques façons principales qu'un script ne fonctionne pas comme prévu, trois d'entre elles sont discutées ici.

Cette section explique comment rechercher ces 3 problèmes courants avec les scripts EEM :

1. Échecs de commande CLI : la commande ne parvient pas à être analysée et ne s'exécute donc pas.
2. Le script s'exécute trop longtemps : les scripts EEM ont un délai d'exécution par défaut de 20 secondes. Si ce délai est dépassé, le script s'arrête avant l'exécution de toutes les commandes.

3. Le script s'exécute trop souvent : parfois, l'événement déclencheur utilisé par le script peut se produire trop fréquemment, ce qui entraîne le déclenchement rapide du script. Il est souhaitable de contrôler la fréquence et la fréquence de déclenchement du script.

Problème : l'exécution des commandes CLI échoue

Cet exemple de script contient plusieurs problèmes. Il s'agit d'un applet simple qui ajoute le résultat de plusieurs commandes show à un fichier texte dans un média 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 | appen
action 2.0 syslog msg  "Data Capture Complete"
```

L'applet s'est exécutée correctement, mais n'a pas généré les résultats attendus :

```
<#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
```

Utilisez l'interface de ligne de commande de débogage du gestionnaire d'événements intégré pour faciliter la vérification des applets.

<#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.

Conclusion : Examinez correctement toutes les actions EEM et utilisez les débogages pour vous protéger contre les erreurs de configuration et les erreurs typographiques.

Problème : les actions EEM prennent plus de temps que le temps d'exécution maximal

Dans ce scénario, un EEM simple est utilisé pour collecter les captures de paquets du plan de contrôle à des intervalles de 120 secondes. Il ajoute de nouvelles données de capture à un fichier de sortie situé sur le support de stockage 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"

Vous pouvez facilement déterminer que l'ESEE ne se termine pas comme prévu. Vérifiez les journaux locaux pour le syslog de l'action 5.0. Ce syslog s'imprime à chaque itération réussie de l'applet. Le journal n'a pas été imprimé dans la mémoire tampon et le fichier CPUCapture.txt n'a pas été écrit dans la mémoire flash :

<#root>

Switch#

show logging | include "CPUCapture Complete"

Switch#

dir flash: | include CPUCapture.txt

Activez les débogages à analyser. Le débogage le plus couramment utilisé est debug event manager action cli. Cet utilitaire imprime une boîte de dialogue des actions dans l'ordre.

Sortie de débogage : la sortie de débogage affiche l'applet appelée avec succès. Les actions initiales s'exécutent sans problème, mais la capture n'aboutit pas.

<#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.
```

Solution : par défaut, les stratégies EEM ne s'exécutent pas pendant plus de 20 secondes. Si l'exécution des actions dans l'EEM prend plus de 20 secondes, l'EEM échoue. Assurez-vous que le temps d'exécution de votre EEM est suffisant pour permettre l'exécution des actions de votre applet. Configurez maxrun pour spécifier une valeur d'exécution maximale plus appropriée.

Exemple

<#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"
```

Problème : EEM se déclenche trop souvent

Parfois, plusieurs instances d'un déclencheur donné se produisent en peu de temps. Cela pourrait conduire à des itérations excessives de l'applet et avoir de graves conséquences dans le pire des cas.

Cette applet se déclenche sur un modèle syslog particulier, puis rassemble la sortie de la commande show et ajoute cette sortie à un fichier. Plus précisément, l'applet se déclenche lorsque le protocole de ligne abandonne une interface identifiée :

<#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 se déclenche chaque fois que le journal système est observé. Un événement tel qu'un rabat d'interface peut se produire rapidement et en peu de temps.

<#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 s'est exécutée plusieurs fois en quelques minutes, ce qui a généré un fichier de sortie indésirable contenant des données superflues. Le fichier continue également à augmenter en taille et continue à remplir les médias locaux. Cet exemple simple d'EEM ne représente pas une menace opérationnelle importante s'il est exécuté de manière répétée, mais ce scénario peut potentiellement entraîner une panne avec des scripts plus complexes.

Dans ce scénario, il serait utile de limiter la fréquence de déclenchement de l'applet.

Solution : appliquez une limite de débit pour contrôler la vitesse d'exécution d'un applet. Le mot clé ratelimit est ajouté à l'instruction trigger et est associé à une valeur en secondes.

Exemple

<#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"
```

Informations connexes

Cisco IOS Embedded Event Manager 4.0

Meilleures pratiques et scripts utiles pour EEM

À propos de cette traduction
Cisco a traduit ce document en traduction automatisée vérifiée par une personne dans le cadre d'un service mondial permettant à nos utilisateurs d'obtenir le contenu d'assistance dans leur propre langue.

Il convient cependant de noter que même la meilleure traduction automatisée ne sera pas aussi précise que celle fournie par un traducteur professionnel.

À propos de cette traduction
Cisco a traduit ce document en traduction automatisée vérifiée par une personne dans le cadre d'un service mondial permettant à nos utilisateurs d'obtenir le contenu d'assistance dans leur propre langue.

Il convient cependant de noter que même la meilleure traduction automatisée ne sera pas aussi précise que celle fournie par un traducteur professionnel.