EEMスクリプトのトラブルシューティングとテスト

内容

はじめに

このドキュメントでは、Embedded Event Manager(EEM)スクリプトの検証について説明し、一般的な運用上の考慮事項と障害のシナリオを紹介します。

前提条件

要件

このドキュメントでは、読者がCisco IOS/IOS XE Embedded Event Manager(EEM)機能について理解していることを前提としています。この機能についてまだ詳しくない場合は、 EEM機能の概要 1つ目は、

Catalyst 9000ファミリスイッチのEEMには、Network Essentialsライセンスレベルの DNAアドオンが必要です。Network AdvantageはEEMを完全にサポートします。

使用するコンポーネント

このドキュメントの情報は、Catalystスイッチファミリに実装されているEEMバージョン4.0に関連しています。

このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されました。このドキュメントで使用するすべてのデバイスは、クリアな（デフォルト）設定で作業を開始しています。本稼働中のネットワークでは、各コマンドによって起こる可能性がある影響を十分確認してください。

背景説明

EEMは効果的に導入された場合に便利な機能ですが、EEMが作成者の意図どおりに動作することを確認することが重要です。十分に検証されていないスクリプトは、実稼働環境で致命的な問題を引き起こす可能性があります。せいぜい、スクリプトは望ましくない方法で実行されます。このドキュメントでは、CLI showコマンドを使用してEEMをテストおよび検証する方法について説明し、一般的な障害シナリオと、問題の特定と修正に使用するデバッグについても説明します。

showコマンドによるEEMの検証

タイマーがアクティブであることの確認

タイマーによってトリガーされるEEMスクリプトを展開する際に、スクリプトが期待どおりに起動しない場合は、タイマーがアクティブでカウントダウンされていることを確認します。

次のEEMスクリプト（それぞれtestおよびtest3）について考えます。

<#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"
```

- 最初のスクリプト(test)は、60秒（無題）のウォッチドッグタイマーを使用してスクリプトを起動します。
- 2番目のスクリプト(test3)は、test3という名前の300秒のウォッチドッグタイマーを使用してスクリプトを起動します。

設定されているタイマーとそのタイマーの現在の値は、show event manager statistics serverコマンドを使用して表示できます。

例

<#root>

Switch#

show event manager statistics server

EEM Queue Information

Client	Triggered Events	Dropped Events	Queue Size	Queue Max	Average 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'

トリガーイベントが発生していることを確認する

このドキュメントの「タイマーがアクティブであることを確認する」セクションで説明されているように、IOS XEはEEMアプレットが起動されるたびに、show event manager statistics serverの出力のEEM Applets client行のTriggered Events列を増やします。EEMスクリプトが期待どおりに動作することを確認するには、トリガーイベントを数回実行し、show event manager statistics serverの出力を調べて、この値が増加することを確認します。そうでない場合は、スクリプトはトリガーされていません。

このコマンドを何度か連続して実行すると、タイマー値がカウントダウンされます。タイマーが

ゼロに達してスクリプトが実行されると、EEMアプレットのトリガーされたイベントカウントもカウントアップされます。

<#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

注：この問題が発生しない場合は、スクリプトを調査して、設定されているタイマーを確認してください。

イベント履歴の確認

タイマーによってトリガーされないスクリプトの場合、アプレットが期待どおりにトリガーされることを確認するには、show event manager history eventsコマンドが便利です。

次のEEMスクリプトについて考えます。

```
<#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
```

このスクリプトは、CLIイベントマネージャのrun test_manualが実行されると実行され、syslogメッセージを出力します。syslogの出力以外に、このスクリプトの実行を確認するには、次に示すようにshow event manager history eventsの出力を確認します。

```
<#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
```

手動トリガーによるEEMの検証

EEMスクリプトを手動でトリガーして、実行フローをテストするか、1回限りのアクションを実行することが望ましいシナリオがあります。これは、次の出力に示すように、event noneのトリガーを持つEEMスクリプトを使用して実行できます。

```
<#root>

event manager

applet test_manual

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

イネーブルプロンプトから、コマンドevent manager run test_manualを使用して、スクリプトを手動で起動します。

```
<#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
```

運用上の考慮事項

実稼働環境で使用する前に、EEMスクリプトが検証されていることを確認します。一般に、スクリプトが期待どおりに動作しない主な方法がいくつかあります。ここでは、そのうちの3つについて説明します。

このセクションでは、EEMスクリプトに関する次の3つの一般的な問題を確認する方法を示します。

1. CLIコマンドの失敗：コマンドの解析に失敗したため、実行に失敗しました。
2. スクリプトの実行時間が長すぎます。EEMスクリプトのデフォルトの実行時間は20秒です。この時間を超えると、すべてのコマンドが実行される前にスクリプトが停止します。
3. スクリプトの実行頻度が高すぎる：スクリプトで使用されるトリガーイベントが頻繁に発生

しすぎて、スクリプトが迅速に起動する場合があります。 スクリプトが起動する頻度とレートを制御することが望ましいです。

問題：CLIコマンドの実行に失敗する

このスクリプト例には、いくつかの問題が含まれています。いくつかのshowコマンドの出力をローカルフラッシュメディアのテキストファイルに追加する簡単なアプレットです。

<#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"
```

アプレットは正常に実行されましたが、予期した結果が生成されませんでした：

<#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
```

アプレットの検証を支援するには、debug embedded event manager action cliを使用します。

<#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.
```

結論：すべてのEEMアクションを適切に検査し、デバッグを使用して設定ミスや入力ミスを防ぎます。

問題：EEMアクションが最大ランタイムよりも長くかかる

このシナリオでは、簡単なEEMを使用して、120秒間隔でコントロールプレーンパケットキャプチャを収集します。 ローカルストレージメディアにある出力ファイルに新しいキャプチャデータを追加します。

<#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"
```

EEMが期待どおりに完了していないことは簡単に判断できます。アクション5.0からのsyslogの口ーカルログをチェックします。 このsyslogは、アプレットが正常に繰り返されるたびに出力されます。バッファ内にログが出力されず、ファイルCPUCapture.txtがフラッシュに書き込まれませんでした。

<#root>

```
Switch#
```

```
show logging | include "CPUCapture Complete"


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

調査するデバッグを有効にします。最もよく使用されるデバッグは、debug event manager action cliです。このユーティリティは、アクションのダイアログを順番に印刷します。

デバッグ出力：デバッグ出力には、アプレットが正常に呼び出されたことが示されます。最初のアクションは問題なく実行されますが、キャプチャの完了に失敗します。

<#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.

解決策：デフォルトでは、EEMポリシーは20秒以内に実行されます。EEM内のアクションの実行に20秒以上かかる場合、EEMは完了しません。EEMのランタイムが、アプレットのアクションを実行するのに十分であることを確認します。maxrunを設定して、より適切な最大ランタイム値を指定します。

例

<#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"
```

問題：EEMのトリガー頻度が高すぎる

特定のトリガーの複数のインスタンスが短時間で発生することがあります。これにより、アプレットが過剰に繰り返され、最悪の場合には深刻な結果が生じる可能性があります。

このアプレットは特定のsyslogパターンでトリガーされ、showコマンドの出力を収集してファイルに追加します。具体的には、アプレットは、特定のインターフェイスで回線プロトコルがドロップしたときに起動されます。

<#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"
```

アプレットは、syslogが観察されるたびに起動します。インターフェイスフラップなどのイベントは、短時間で迅速に発生する可能性があります。

<#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
```

アプレットは数分の間に何度も実行され、無関係なデータを含む望ましくない出力ファイルが生成されました。また、ファイルのサイズが増え続け、ローカルメディアがいっぱいになる傾向があります。この単純なEEMの例は、繰り返し実行される場合、運用上の脅威にはなりませんが、このシナリオでは、より複雑なスクリプトを使用してクラッシュする可能性があります。

このシナリオでは、アプレットがトリガーされる頻度を制限することが有益です。

解決策：レート制限を適用して、アプレットの実行速度を制御します。ratelimitキーワードはtrigger文に追加され、秒単位の値に関連付けられます。

例

<#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"
```

関連情報

Cisco IOS Embedded Event Manager 4.0

EEMのベストプラクティスと便利なスクリプト

翻訳について
シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。