EEM 스크립트 문제 해결 및 테스트

목차

소개

이 문서에서는 EEM(Embedded Event Manager) 스크립트 검증에 대해 설명하고 일반적인 작동 고려 사항 및 실패 시나리오를 소개합니다.

사전 요구 사항

요구 사항

이 문서에서는 독자가 Cisco IOS/IOS XE EEM(Embedded Event Manager) 기능에 대해 이미 잘 알고 있다고 가정합니다. 이 기능에 대해 잘 모르는 경우 EEM 기능 개요 먼저.

Catalyst 9K 스위치 제품군의 EEM에는 Network Essentials 라이센스 레벨에 대한 DNA 추가 기능이 필요합니다. Network Advantage는 EEM을 완벽하게 지원합니다.

사용되는 구성 요소

이 문서의 정보는 Catalyst 스위치 제품군에 구현된 EEM 버전 4.0과 관련이 있습니다.

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다. 이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다. 현재 네트워크가 작동 중인 경우 모든 명령의 잠재적인 영향을 미리 숙지하시기 바랍니다.

배경 정보

EEM은 효과적으로 구축할 때 유용한 기능이지만, EEM이 저자가 의도하는 바를 정확히 수행하도록 하는 것이 중요하다. 제대로 검사되지 않은 스크립트는 생산 과정에서 심각한 문제를 야기할 수 있습니다. 기껏해야 스크립트는 바람직하지 않은 방식으로 수행됩니다. 이 문서에서는 CLI show 명령을 사용하여 EEM을 테스트하고 확인하는 방법에 대한 유용한 정보를 제공하며, 몇 가지 일반적인 실패 시나리오와 문제를 식별하고 수정하는 데 사용되는 디버그에 대해서도 설명합니다.

Show 명령을 사용한 EEM 유효성 검사

타이머가 활성 상태인지 확인합니다.

타이머에 의해 트리거되는 EEM 스크립트가 구축된 경우 스크립트가 예상대로 실행되지 않으면 타이머가 활성 상태이고 카운트다운되는지 확인합니다.

각각 test 및 test3이라는 EEM 스크립트를 고려하십시오.

<#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"
```

- 첫 번째 스크립트(테스트)는 60초(명명되지 않음) watchdog 타이머를 사용하여 스크립트를 실행합니다.
- 두 번째 스크립트(test3)는 test3이라는 300개의 두 번째 watchdog 타이머를 사용하여 스크립트를 실행합니다.

구성된 타이머와 이러한 타이머의 현재 값은 show event manager statistics server 명령을 사용하여 볼 수 있습니다.

예

<#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'
```

트리거 이벤트 실행 확인

이 문서의 Confirm Timers are Active 섹션에서 설명한 것처럼 IOS XE는 EEM 애플릿이 실행될 때마다 show event manager statistics server의 출력에서 EEM 애플릿 클라이언트 행에 대해 Triggered Events 열을 증가시킵니다. EEM 스크립트가 예상대로 작동하는지 확인하려면 트리거 이벤트를 여러 번 수행하고 show event manager statistics server의 출력을 검사하여 이 값 증분을 확인합니다. 그렇지 않으면 스크립트가 트리거되지 않습니다.

명령이 순서대로 여러 번 실행되면 카운트다운할 타이머 값이 계산됩니다. 타이머가 0에 도달하고 스크립트가 실행되면 EEM 애플릿에 대해 트리거된 이벤트 수도 계산됩니다.

```
<#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
```

참고: 이 문제가 발생하지 않으면 스크립트를 조사하여 구성된 타이머를 확인하십시오.

이벤트 기록 검토

타이머에 의해 트리거되지 않는 스크립트의 경우 show event manager history events 명령은 애플릿이 예상대로 트리거되는지 확인하는 데 유용합니다.

이 EEM 스크립트를 고려해 보십시오.

<#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
```

이 스크립트는 CLI 이벤트 관리자 run test_manual이 실행될 때 실행되고 syslog 메시지를 인쇄합니다. syslog의 출력 외에도 다음과 같이 show event manager history 이벤트의 출력을 검토하여 이 스크립트의 실행을 확인할 수 있습니다.

```
<#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
```

수동 트리거를 사용한 EEM 검증

실행 흐름을 테스트하거나 일회성 작업을 수행하기 위해 EEM 스크립트를 수동으로 트리거하는 것이 바람직한 시나리오가 있습니다. 이 출력은 다음과 같이 이벤트 없음 트리거가 있는 EEM 스크립트로 수행할 수 있습니다.

```
<#root>
```

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

enable 프롬프트에서 test_manual을 실행하는 명령 이벤트 관리자를 사용하여 스크립트를 수동으로 실행합니다.

```
<#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
```

운영 고려 사항

EEM 스크립트를 프로덕션 환경에서 사용하기 전에 검증해야 합니다. 일반적으로 스크립트가 예상대로 작동하지 않는 몇 가지 주요 방법이 있으며, 이 중 세 가지 방법은 여기에서 설명합니다.

이 섹션에서는 EEM 스크립트에서 다음 3가지 일반적인 문제를 확인하는 방법을 보여줍니다.

1. CLI 명령 실패: 명령을 구문 분석할 수 없으므로 실행하지 못합니다.
2. 스크립트가 너무 오래 실행됩니다. EEM 스크립트의 기본 실행 시간 제한은 20초입니다. 이 시간을 초과하면 모든 명령이 실행되기 전에 스크립트가 중지됩니다.
3. 스크립트가 너무 자주 실행됨: 스크립트에서 사용되는 트리거 이벤트가 너무 자주 발생하여 스크립트가 빠르게 실행될 수 있습니다. 스크립트의 발생 빈도와 속도를 제어하는 것이 좋습니다.

문제: CLI 명령이 실행되지 않음

이 예제 스크립트에는 몇 가지 문제가 있습니다. 로컬 플래시 미디어의 텍스트 파일에 여러 show 명령의 출력을 추가하는 간단한 애플릿입니다.

```
<#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"
```

애플릿이 성공적으로 실행되었지만 예상 결과가 생성되지 않았습니다.

```
<#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
```

애플릿 확인을 지원하기 위해 debug embedded event manager action cli를 사용합니다.

```
<#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.

결론: 모든 EEM 작업을 올바르게 검사하고 디버그를 사용하여 잘못된 컨피그레이션 및 입력 오류를 방지합니다.

문제: EEM 작업이 최대 런타임보다 오래 걸립니다.

이 시나리오에서는 간단한 EEM을 사용하여 120초 간격으로 컨트롤 플레인 패킷 캡처를 수집합니다. 로컬 스토리지 미디어에 있는 출력 파일에 새 캡처 데이터를 추가합니다.

<#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"
```

EEM이 예상대로 완료되지 않았음을 쉽게 확인할 수 있습니다. 작업 5.0에서 syslog에 대한 로컬 로그를 확인합니다. 이 syslog는 애플릿이 성공적으로 반복될 때마다 인쇄됩니다. 로그가 버퍼 내에 인쇄되지 않았고 CPUCapture.txt 파일이 플래시에 기록되지 않았습니다.

<#root>

```
Switch#
show logging | include "CPUCapture Complete"

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

조사할 디버그를 활성화합니다. 가장 일반적으로 사용되는 debug는 debug event manager action cli입니다. 이 유틸리티는 작업의 대화 상자를 순서대로 인쇄합니다.

디버그 출력: 디버그 출력에 성공적으로 호출된 애플릿이 표시됩니다. 초기 작업은 문제 없이 실행되지만 캡처가 완료되지 않습니다.

<#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.
```

해결 방법: 기본적으로 EEM 정책은 20초를 넘지 않습니다. EEM 내의 작업을 실행하는 데 20초 이상 걸리면 EEM을 완료할 수 없습니다. EEM의 런타임이 애플릿 작업이 실행될 수 있도록 충분한지 확인합니다. 더 적절한 최대 런타임 값을 지정하도록 maxrun을 구성합니다.

예

```
<#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"
```

문제: EEM이 너무 자주 트리거됨

때때로, 주어진 트리거의 몇몇 인스턴스들이 짧은 시간 내에 발생한다. 이는 애플릿의 과도한 반복으로 이어질 수 있으며 최악의 경우 심각한 결과를 초래할 수 있다.

이 애플릿은 특정 syslog 패턴에서 트리거된 다음 show 명령 출력을 수집하고 이 출력을 파일에 추가합니다. 특히, 애플릿은 확인된 인터페이스에 대한 라인 프로토콜이 삭제될 때 실행됩니다.

```
<#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"
```

이 애플릿은 syslog가 관찰될 때마다 실행됩니다. 인터페이스 플랩과 같은 이벤트가 단시간에 신속하게 발생할 수 있다.

<#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

이 애플릿은 몇 분 동안 여러 번 실행되어 관련 없는 데이터가 포함된 바람직하지 않은 출력 파일이 생성되었습니다. 또한 파일 크기가 계속 증가하고 로컬 미디어를 계속 채웁니다. 이 간단한 예에서 EEM을 반복적으로 실행하면 운영 위협이 별로 없지만, 이 시나리오에서는 스크립트가 더 복잡해질 수 있습니다.

이 시나리오에서는 애플릿이 트리거되는 빈도를 제한하는 것이 좋습니다.

해결 방법: 속도 제한을 적용하여 애플릿의 실행 속도를 제어합니다. ratelimit 키워드는 트리거 문에 추가되고 초 단위의 값과 연결됩니다.

예

<#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"
```

관련 정보

Cisco IOS Embedded Event Manager 4.0

EEM을 위한 모범 사례 및 유용한 스크립트

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.