Comprendre les meilleures pratiques et les scripts utiles pour EEM

Table des matières

Introduction

Ce document décrit les meilleures pratiques de configuration des scripts EEM (Embedded Event Manager) sur les périphériques Cisco IOS®-XE et fournit des exemples de syntaxe courante et de scripts utiles.

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 fonction, lisez d'abord la présentation de la fonction EEM.

Composants utilisés

Les informations contenues dans ce document sont basées sur les versions de matériel et de logiciel suivantes :

- Commutateurs Cisco Catalyst 9300, 9400 et 9500 qui exécutent le logiciel Cisco IOS Version 16.X ou 17.X

Ces scripts ne sont pas pris en charge par le centre d'assistance technique Cisco et sont fournis tels quels à des fins pédagogiques.

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 lâ€™incidence possible des commandes.

Conventions

Pour plus d'informations sur les conventions utilisées dans ce document, reportez-vous à Conventions relatives aux conseils techniques Cisco.

Meilleures pratiques

Cette section couvre certains des problèmes les plus courants observés lors de la conception et de la mise en oeuvre des scripts EEM. Pour plus d'informations sur les meilleures pratiques d'ESEE, reportez-vous au document Meilleures pratiques d'ESEE référencé dans la section Références.

Confirmer que l'authentification appropriée est en place

Si votre périphérique utilise AAA, vous devez vous assurer que les scripts EEM configurés sur le périphérique sont soit configurés avec un utilisateur AAA capable d'exécuter les commandes dans le script, soit que le contournement d'autorisation est configuré avec la commande **authorization bypass** dans la définition de script.

Ajouter des contraintes pour l'exécution EEM et Rate-limit

Par défaut, les scripts EEM peuvent s'exécuter pendant 20 secondes au maximum. Si vous concevez un script qui prend plus de temps à exécuter, ou qui doit attendre entre l'exécution de la commande, spécifiez une valeur **maxrun** sur le déclencheur d'événement d'applet pour modifier le minuteur d'exécution par défaut.

Il est également important de tenir compte de la fréquence d'exécution de l'événement qui déclenche votre script EEM. Si vous déclenchez un script à partir d'une condition qui se produit rapidement en un court laps de temps (par exemple, un déclencheur syslog pour les battements MAC), il est important d'inclure une condition de limite de débit dans votre script EEM pour empêcher un nombre excessif d'exécutions en parallèle et empêcher l'épuisement des ressources du périphérique.

Éviter l'exécution hors ordre

Comme décrit dans la documentation EEM, l'ordre d'exécution des instructions d'action est contrôlé par leur étiquette (par exemple, la commande d'interface de ligne de commande action 0001 **enable** a l'étiquette 0001). Cette valeur d'étiquette n'est PAS un nombre, mais plutôt alphanumérique. Les actions sont triées dans la séquence de clés alphanumériques ascendantes, utilisent l'argument **label** comme clé de tri, et elles sont exécutées dans cette séquence. Cela peut entraîner un ordre d'exécution inattendu, en fonction de la façon dont vous structurez vos étiquettes d'action.

Considérez cet exemple :

```
event manager applet test authorization bypass
event timer watchdog time 60 maxrun 60
action 13 syslog msg "You would expect to see this message first"
action 120 syslog msg "This message prints first"
```

Puisque 120 est avant 13 dans une comparaison alphanumérique, ce script ne s'exécute pas dans l'ordre que vous attendez. Pour éviter cela, il est utile d'utiliser un système de remplissage comme ceci :

```
event manager applet test authorization bypass
event timer watchdog time 60 maxrun 60
action 0010 syslog msg "This message appears first"
action 0020 syslog msg "This message appears second"
action 0120 syslog msg "This message appears third"
```

En raison du remplissage ici, les instructions numérotées sont évaluées dans l'ordre prévu. L'incrément de 10 entre chaque étiquette permet d'insérer ultérieurement des instructions supplémentaires dans le script EEM, le cas échéant, sans avoir à renuméroter toutes les instructions suivantes.

Désactiver la pagination

EEM recherche l'invite du périphérique pour déterminer quand la sortie de la commande est terminée. Les commandes qui génèrent plus de données que ce qui peut être affiché sur un écran (comme configuré par la longueur de votre terminal) peuvent empêcher l'exécution des scripts EEM (et éventuellement les arrêter via le minuteur maxrun), car l'invite du périphérique n'est pas affichée tant que toutes les pages du résultat ne sont pas affichées. Configurez le **terme len 0** au début des scripts EEM qui examinent les sorties importantes.

Scripts de conception pour une maintenance future

Lorsque vous concevez un script EEM, laissez des espaces entre les étiquettes d'action pour faciliter la mise à jour de la logique du script EEM à l'avenir. Lorsque des écarts appropriés sont disponibles (c'est-à-dire que deux instructions telles que **action 0010** et **action 0020** laissent un écart de 9 étiquettes qui peuvent être insérées), de nouvelles instructions peuvent être ajoutées selon les besoins sans renuméroter ou vérifier à nouveau les étiquettes d'action et s'assurer que les actions continuent à s'exécuter dans l'ordre prévu.

Il existe des commandes courantes que vous devez exécuter au début de vos scripts EEM. Cela peut inclure :

- définir la longueur de la borne sur 0
- passez en mode enable
- enable automatic timestamp for command output

Il s'agit d'un modèle courant dans les exemples présentés dans ce document, où de nombreux scripts commencent par les mêmes 3 instructions d'action pour configurer cela.

Schémas logiques communs aux modules EEM

Cette section couvre certains modèles logiques et blocs de syntaxe courants utilisés dans les scripts EEM. Les exemples ci-dessous ne sont pas des scripts complets, mais plutôt des démonstrations de la façon dont une fonctionnalité spécifique peut être utilisée pour créer des scripts EEM complexes.

Chemins de code d'agence avec if/else

Les variables EEM peuvent être utilisées pour contrôler le flux d'exécution des scripts EEM. Considérez ce script EEM :

```
event manager applet snmp_cpu authorization bypass
event timer watchdog time 60
action 0010 info type snmp oid 1.3.6.1.4.1.9.9.109.1.1.1.1.3 get-type exact
action 0020 if $_info_snmp_value ge "50"
action 0030 syslog msg "This syslog message is sent if CPU utilization is above 50%"
action 0040 elseif $_info_snmp_value ge "30"
action 0050 syslog msg "This syslog message is sent if CPU utilization is above 30% and below 50%"
action 0060 else
action 0070 syslog msg "This syslog message is sent if CPU utilization is below 30%"
action 0080 end
```

Ce script s'exécute toutes les minutes. Examinez la valeur de l'OID SNMP pour l'utilisation du CPU, puis entrez l'un des trois chemins d'exécution différents en fonction de la valeur de l'OID. Des instructions similaires peuvent être utilisées sur toute autre variable EEM légale pour créer des flux d'exécution complexes dans des scripts EEM.

Instructions de bouclage

Les boucles d'exécution peuvent être utilisées pour raccourcir considérablement les scripts EEM et les rendre plus faciles à expliquer. Considérez ce script, conçu pour extraire les statistiques d'interface pour Te2/1/15 6 fois sur une période d'1 minute pour vérifier les petites périodes d'utilisation élevée :

```
event manager applet int_util_check auth bypass
event timer watchdog time 300 maxrun 120
action 0001 cli command "enable"
action 0002 cli command "term exec prompt timestamp"
action 0003 cli command "term length 0"
action 0010 syslog msg "Running iteration 1 of command"
action 0020 cli command "show interface te2/1/15 | append flash:interface_util.txt"
action 0030 wait 10
action 0040 syslog msg "Running iteration 2 of command"
action 0050 cli command "show interface te2/1/15 | append flash:interface_util.txt"
action 0060 wait 10
action 0070 syslog msg "Running iteration 3 of command"
action 0080 cli command "show interface te2/1/15 | append flash:interface_util.txt"
action 0090 wait 10
action 0100 syslog msg "Running iteration 4 of command"
action 0110 cli command "show interface te2/1/15 | append flash:interface_util.txt"
action 0120 wait 10
action 0130 syslog msg "Running iteration 5 of command"
action 0140 cli command "show interface te2/1/15 | append flash:interface_util.txt"
action 0150 wait 10
action 0160 syslog msg "Running iteration 6 of command"
action 0170 cli command "show interface te2/1/15 | append flash:interface_util.txt"
```

Avec les constructions de **boucle EEM**, ce script peut être considérablement raccourci :

```
event manager applet int_util_check auth bypass
event timer watchdog time 300 maxrun 120
action 0001 cli command "enable"
action 0002 cli command "term exec prompt timestamp"
action 0003 cli command "term length 0"
action 0010 set loop_iteration 1
action 0020 while $loop_iteration le 6
action 0030 syslog msg "Running iteration $loop_iteration of command"
action 0040 cli command "show interface te2/1/15 | append flash:interface_util.txt"
action 0050 wait 10
action 0060 increment loop_iteration 1
action 0070 end
```

Extraire la sortie via des expressions régulières (Regex)

L'instruction regexp d'EEM peut être utilisée pour extraire des valeurs de la sortie de commande à utiliser dans les commandes suivantes et pour activer la création dynamique de commandes dans le script EEM lui-même. Référez-vous à ce bloc de code pour un exemple pour extraire le PID SNMP ENGINE du résultat de **show proc cpu | dans le moteur SNMP** et l'imprimer dans un message syslog. Cette valeur extraite peut également être utilisée dans d'autres commandes qui nécessitent un PID pour s'exécuter.

```
event manager applet check_pid auth bypass
event none
action 0010 cli command "show proc cpu | i SNMP ENGINE"
action 0020 regexp "^[ ]*([0-9]+) .*" $_cli_result match match1
action 0030 syslog msg "Found SNMP Engine PID $match1"
```

Scripts EEM utiles

Suivre l'adresse MAC spécifique pour l'apprentissage des adresses MAC

Dans cet exemple, l'adresse MAC ***b4e9.b0d3.6a41*** est suivie. Le script vérifie toutes les 30 secondes si l'adresse MAC spécifiée a été apprise dans les tables ARP ou MAC. Si le MAC est détecté, le script effectue les actions suivantes :

- génère un message syslog (utile lorsque vous voulez confirmer où une adresse MAC est apprise, ou quand/à quelle fréquence elle est apprise).

Mise En Oeuvre

```
event manager applet mac_trace authorization bypass
event timer watchdog time 30
action 0001 cli command "enable"
action 0002 cli command "term exec prompt timestamp"
action 0003 cli command "term length 0"
action 0010 cli command "show ip arp | in b4e9.b0d3.6a41"
action 0020 regexp ".*(ARPA).*" $_cli_result
action 0030 if $_regexp_result eq 1
action 0040 syslog msg $_cli_result
```

```
action 0050 end
action 0060 cli command "show mac add vlan 1 | in b4e9.b0d3.6a41"
action 0070 regexp ".*(DYNAMIC).*" $_cli_result
action 0080 if $_regexp_result eq 1
action 0090 syslog msg $_cli_result
action 0100 end
```

Surveiller le CPU élevé via SNMP OID

Ce script surveille un OID SNMP utilisé pour lire le pourcentage de CPU occupé au cours des 5 dernières secondes. Lorsque le processeur est occupé à plus de 80 %, le script effectue les actions suivantes :

- crée un horodatage à partir du résultat de la commande show clock et l'utilise pour créer un nom de fichier unique
- les sorties sur l'état du processus et du logiciel sont ensuite écrites dans ce fichier
- une capture de paquets intégrée (EPC) est configurée pour capturer 10 secondes de trafic destiné au plan de contrôle et l'écrit dans un fichier.
- une fois la capture EPC terminée, la configuration EPC est supprimée et le script se ferme.

Mise En Oeuvre

```
event manager applet high-cpu authorization bypass
event snmp oid 1.3.6.1.4.1.9.9.109.1.1.1.1.3 get-type next entry-op gt entry-val 80 poll-interval 1 rate
action 0001 cli command "enable"
action 0002 cli command "term exec prompt timestamp"
action 0003 cli command "term length 0"
action 0010 syslog msg "High CPU detected, gathering system information."
action 0020 cli command "show clock"
action 0030 regex "([0-9]|[0-9][0-9]):([0-9]|[0-9][0-9]):([0-9]|[0-9][0-9])" $_cli_result match match1
action 0040 string replace "$match" 2 2 "."
action 0050 string replace "$_string_result" 5 5 "."
action 0060 set time $_string_result
action 0070 cli command "show proc cpu sort | append flash:tac-cpu-$time.txt"
action 0080 cli command "show proc cpu hist | append flash:tac-cpu-$time.txt"
action 0090 cli command "show proc cpu platform sorted | append flash:tac-cpu-$time.txt"
action 0100 cli command "show interface | append flash:tac-cpu-$time.txt"
action 0110 cli command "show interface stats | append flash:tac-cpu-$time.txt"
action 0120 cli command "show log | append flash:tac-cpu-$time.txt"
action 0130 cli command "show ip traffic | append flash:tac-cpu-$time.txt"
action 0140 cli command "show users | append flash:tac-cpu-$time.txt"
action 0150 cli command "show platform software fed switch active punt cause summary | append flash:tac-
action 0160 cli command "show platform software fed switch active cpu-interface | append flash:tac-cpu-$
action 0170 cli command "show platform software fed switch active punt cpuq all | append flash:tac-cpu-$
action 0180 cli command "no monitor capture tac_cpu"
action 0190 cli command "monitor capture tac_cpu control-plane in match any file location flash:tac-cpu-
action 0200 cli command "monitor capture tac_cpu start" pattern "yes"
action 0210 cli command "yes"
action 0220 wait 10
action 0230 cli command "monitor capture tac_cpu stop"
action 0240 cli command "no monitor capture tac_cpu"
```

Faire correspondre dynamiquement un PID et le résultat de la pile d'enregistrements

Ce script recherche un message syslog indiquant que la file d'attente d'entrée SNMP est pleine et effectue les actions suivantes :

- enregistre le résultat de la commande **show proc cpu sort** dans un fichier
- extrait le PID du processus SNMP ENGINE via regex
- utilise le PID SNMP dans les commandes suivantes pour obtenir les données de pile du PID
- supprime le script de la configuration afin d'éviter toute nouvelle exécution

Mise En Oeuvre

```
event manager applet TAC-SNMP-INPUT-QUEUE-FULL authorization bypass
event syslog pattern "INPUT_QFULL_ERR" ratelimit 40 maxrun 120
action 0010 cli command "en"
action 0020 cli command "show proc cpu sort | append flash:TAC-SNMP.txt"
action 0030 cli command "show proc cpu | i SNMP ENGINE"
action 0040 regexp "^[ ]*([0-9]+) .*" $_cli_result match match1
action 0050 syslog msg "Found SNMP Engine PID $match1"
action 0060 cli command "show stacks $match1 | append flash:TAC-SNMP.txt"
action 0070 syslog msg "$_cli_result"
action 0080 cli command "configure terminal"
action 0090 cli command "no event manager applet TAC-SNMP-INPUT-QUEUE-FULL"
action 0100 cli command "end"
```

Mettre à niveau un commutateur

Ce script est configuré pour la correspondance de modèle sur l'invite non-standard retournée par la commande **install add file <file> activate commit** et répondre aux invites. Aucun événement déclencheur n'est configuré, de sorte que le script EEM doit être déclenché manuellement par un utilisateur lorsque la mise à niveau doit avoir lieu via le **gestionnaire d'événements et exécuter UPGRADE**. Le compteur maxrun est défini pour 300 secondes au lieu de la valeur par défaut de 20 secondes, car l'exécution de la commande **install add** prend un temps important.

Mise En Oeuvre

```
event manager applet UPGRADE authorization bypass
event none maxrun 300
action 0001 cli command "enable"
action 0002 cli command "term length 0"
action 0020 cli command "install add file flash:cat9k_iosxe.16.06.02.SPA.bin activate commit" pattern "y
action 0030 cli command "y" pattern "y\/n"
action 0040 syslog msg  "Reloading device to upgrade code"
action 0050 cli command "y"
```

Vider les données de diagnostic dans un fichier lorsqu'un objet suivi IP SLA tombe en panne

Ce script est déclenché lorsque l'objet IP SLA 11 tombe en panne et exécute les actions suivantes :

- Collecter la table MAC, la table ARP, les syslogs et la table de routage
- Écrire des informations dans un fichier sur la mémoire flash : appelé sla_track.txt

Mise En Oeuvre

```
ip sla 10
icmp-echo 10.10.10.10 source-ip 10.10.10.10
frequency 10
exit
ip sla schedule 10 life forever start-time now
track 11 ip sla 10 reachability
exit
event manager applet track-10 authorization bypass
event track 11 state down
action 0001 cli command "enable"
action 0002 cli command "term exec prompt timestamp"
action 0003 cli command "term length 0"
action 0010 syslog msg  "IP SLA object 10 has gone down"
action 0020 cli command "show mac address-table detail | append flash:sla_track.txt"
action 0030 cli command "show ip arp | append flash:sla_track.txt"
action 0040 cli command "show log | append flash:sla_track.txt"
action 0050 cli command "show ip route | append flash:sla_track.txt"
```

Envoyer un e-mail de EEM

Ce script est déclenché lorsque le modèle décrit dans l'instruction de modèle syslog d'événement est vu et exécute les actions suivantes :

- envoie un e-mail à partir d'un serveur de messagerie interne (cela suppose que le serveur de messagerie interne autorise l'authentification ouverte à partir du périphérique).

Mise En Oeuvre

```
event manager environment email_from email_address@company.test
event manager environment email_server 192.168.1.1
event manager environment email_to dest_address@company.test
event manager applet email_syslog
event syslog pattern "SYSLOG PATTERN HEREâ€ maxrun 60
action 0010 info type routername
action 0020 mail server "$email_server" to "$email_to" from "$email_from" subject "SUBJECT OF EMAIL - Sy
```

Arrêt d'un port sur une planification

Ce script arrête le port Te2/1/15 tous les jours à 18 heures.

Mise En Oeuvre

```
event manager applet shut_port authorization bypass
event timer cron cron-entry "0 18 * * *"
action 0001 cli command "enable"
action 0002 cli command "term exec prompt timestamp"
action 0003 cli command "term length 0"
action 0010 syslog msg  "shutting port Te2/1/15 down"
```

```
action 0030 cli command "config t"
action 0040 cli command "int Te2/1/15"
action 0050 cli command "shutdown"
action 0060 cli command "end"
```

Arrêter une interface si un débit de paquets par seconde (PPS) donné est atteint

Ce script vérifie le débit PPS sur l'interface Te2/1/9 dans la direction TX toutes les secondes. Si le débit PPS dépasse 100, il prend les mesures suivantes :

- consigne la sortie **show int** pour l'interface dans syslog
- arrête lâ€™interface

Mise En Oeuvre

```
event manager applet disable_link authorization bypass
event interface name te2/1/9  parameter  transmit_rate_pps entry-op ge entry-val 100 poll-interval 1 ent
action 0001 cli command "enable"
action 0002 cli command "term length 0"
action 0010 syslog msg  "Detecting high input rate on interface te2/1/9.  Shutting interface down."
action 0020 cli command "show int te2/1/9"
action 0030 syslog msg  $_cli_result
action 0040 cli command "config t"
action 0050 cli command "int te2/1/9"
action 0060 cli command "shutdown"
action 0070 cli command "end"
```

Références

- Meilleures pratiques Cisco 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.