# EEMのベストプラクティスと便利なスクリプトを理解する

## 内容



## はじめに

このドキュメントでは、Cisco IOS® XEデバイスでのEmbedded Event Manager(EEM)スクリプト設定のベストプラクティスについて説明します。

## 前提条件

### 要件

次の項目に関する知識と知識があることが推奨されます。

- Cisco IOSおよびCisco IOS XE Embedded Event Manager(EEM)

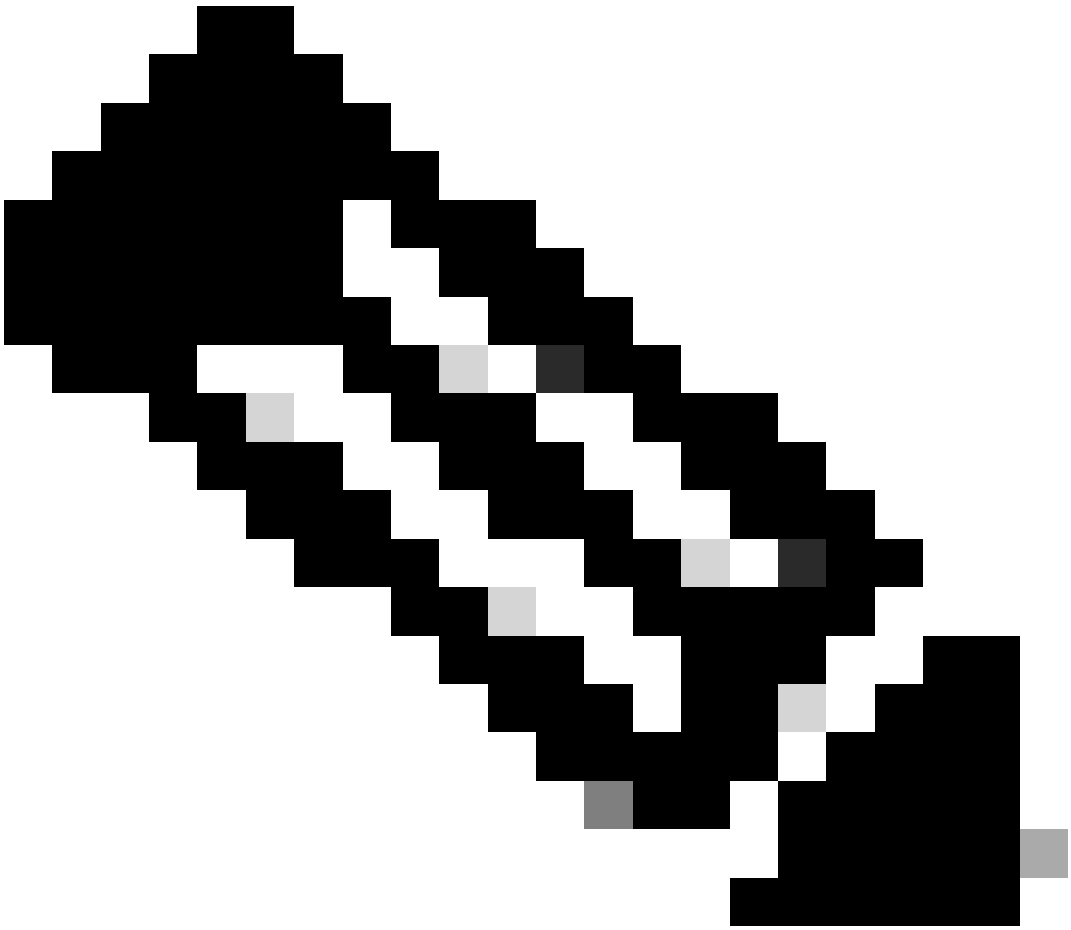
この機能について詳しくない場合は、最初に「EEM機能の概要」をお読みください。

## 使用するコンポーネント

このドキュメントの情報は、次のソフトウェアとハードウェアのバージョンに基づいています。

- Cisco Catalyst 9300、9400、および9500スイッチ
- Cisco IOSソフトウェアバージョン16.Xまたは17.X

---



> 注：これらのスクリプトはCisco TACではサポートされておらず、教育の目的で現状のまま提供されます。

---

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。

## 表記法

ドキュメント表記の詳細は、『シスコ テクニカル ティップスの表記法』を参照してください。

# ベスト プラクティス

このセクションでは、EEMスクリプトの設計と実装で発生する最も一般的な問題について説明します。EEMのベストプラクティスの詳細については、「参考資料」セクションで参照されているEEMのベストプラクティスに関するドキュメントを参照してください。

## 適切な認証が行われていることを確認する

デバイスでAAAを使用する場合、デバイスに設定されているEEMスクリプトで、スクリプトのコマンドを実行できるAAAユーザが設定されているか、またはスクリプト定義でauthorization bypassコマンドが許可バイパスに設定されていることを確認する必要があります。

## EEMランタイムとレート制限の制約の追加

デフォルトでは、EEMスクリプトは最大20秒間実行できます。実行に時間がかかる、またはコマンドの実行間隔で待機する必要があるスクリプトを設計する場合は、アプレットイベントトリガーでmaxrun値を指定して、デフォルトの実行タイマーを変更します。

また、EEMスクリプトをトリガーするイベントを実行できる頻度を考慮することも重要です。短時間で急速に発生する条件（MACフラップのsyslogトリガーなど）からスクリプトをトリガーする場合は、EEMスクリプトにレート制限条件を含めて、並行して実行される過剰な回数を防ぎ、デバイスリソースの枯渇を防ぐことが重要です。

## 順不同の実行の回避

EEMのドキュメントで説明されているように、action文の実行順序はラベルによって制御されます(たとえば、action 0001 cliコマンドenableのラベルは0001です)。このラベル値は数字ではなく、英数字です。 アクションは昇順の英数字キーシーケンスでソートされ、label引数をソートキーとして使用すると、このシーケンスで実行されます。これにより、アクションラベルの構造によっては、予期しない実行順序が発生する可能性があります。

次の例を検討します。

```
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"
```

英数字の比較では120は13より前であるため、このスクリプトは期待する順序では実行されません。これを回避するには、次のようなパディングのシステムを使用すると便利です。

```
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"
```

ここにパディングがあるため、番号付きステートメントは期待される順序で評価されます。各ラベルの間の10の増分により、後続のステートメントの番号を変更することなく、必要に応じて後からEEMスクリプトに追加ステートメントを挿入できます。

### ページ区切りを無効にする

EEMはデバイスのプロンプトを検索して、コマンド出力が完了したかどうかを判断します。デバイスのプロンプトは出力のすべてのページが表示されるまで表示されないため、1つの画面（端末の長さによって設定）に表示できるデータよりも多くのデータを出力するコマンドを使用すると、EEMスクリプトが完了しない（最終的にはmaxrunタイマーによって終了する）ことがあります。大きな出力を検査するEEMスクリプトの先頭にterm len 0を設定します。

### 将来のメンテナンスに備えた設計スクリプト

EEMスクリプトを設計する際は、アクションラベル間にギャップを残しておくと、今後EEMスクリプトロジックを簡単に更新できます。適切なギャップがある場合(つまり、action 0010とaction 0020などの2つの文によって、挿入可能な9つのラベルのギャップが残っている場合)、アクションラベルの再番号付けや再確認を行わずに、必要に応じて新しい文を追加し、アクションを期待どおりの順序で実行し続けることができます。

EEMスクリプトの開始時に実行する必要がある一般的なコマンドがあります。これには次のものが含まれます。

- 端子の長さを0に設定
- イネーブルモードに入る
- コマンド出力の自動タイムスタンプを有効にする

これは、このドキュメントに示す例の一般的なパターンです。スクリプトの多くは、これを設定するために同じ3つのaction文で始まります。

## 一般的なEEMロジックパターン

このセクションでは、EEMスクリプトで使用される一般的な論理パターンと構文ブロックについて説明します。次の例は、完全なスクリプトではなく、特定の機能を使用して複雑なEEMスクリプトを作成する方法を示しています。

### If/Elseによる分岐コードパス

EEM変数は、EEMスクリプトの実行フローを制御するために使用できます。次のEEMスクリプトについて考えてみます。

```
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
```

このスクリプトは毎分実行されます。CPU使用率のSNMP OIDの値を調べ、OIDの値に基づいて3つの異なる実行パスのいずれかを入力します。同様のステートメントを他の有効なEEM変数で使用して、EEMスクリプトで複雑な実行フローを構築できます。

## Loop Over文

実行ループを使用してEEMスクリプトを大幅に短縮し、理由を説明しやすくします。Te2/1/15のインターフェイス統計情報を1分間に6回プルして、短い期間の高使用率をチェックするように設計された、次のスクリプトについて考えてみます。

```
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"
```

EEMループ構造を使用すると、このスクリプトを大幅に短縮できます。

```
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
```

### 正規表現による出力の抽出(Regex)

EEM regexp文を使用すると、コマンド出力から値を抽出して後続のコマンドで使用したり、EEMスクリプト内で動的なコマンド作成を実行したりできます。show proc cpuの出力からSNMPエンジンPIDを抽出する例については、次のコードブロックを参照してください。 | i SNMP engineを使用して、それをsyslogメッセージに出力します。この抽出値は、PIDの実行を必要とする他のコマンドでも使用できます。

```
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"
```

## 便利なEEMスクリプト

### MACアドレス学習のための特定のMACアドレスの追跡

この例では、MACアドレスb4e9.b0d3.6a41が追跡されます。スクリプトは30秒ごとにチェックを行い、指定されたMACアドレスがARPテーブルまたはMACテーブルで学習されているかどうかを確認します。MACが検出されると、スクリプトは次のアクションを実行します。

- syslogメッセージを出力します ( これは、MACアドレスがどこで学習されたのか、いつ、どのくらいの頻度で学習されたかを確認する場合に便利です ) 。

実装

```
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
```

## SNMP OIDによる高CPUの監視

このスクリプトは、最後の5秒間にCPUビジー率を読み取るために使用されるSNMP OIDを監視します。CPUの使用率が80 %を超えると、スクリプトは次のアクションを実行します。

- show clockの出力からタイムスタンプを作成し、これを使用して一意のファイル名を作成します。
- プロセスとソフトウェアの状態に関する出力は、このファイルに書き込まれます
- 組み込みパケットキャプチャ(EPC)は、コントロールプレーン宛ての10秒のトラフィックをキャプチャしてファイルに書き込むように設定されます。
- epcキャプチャが完了すると、EPC設定が削除され、スクリプトが終了します。

実装

```
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 rat
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"
```

## PIDの動的な一致とスタック出力の記録

このスクリプトは、SNMP入力キューがいっぱいであることを示すsyslogメッセージを探し、次のアクションを実行します。

- show proc cpu sortの出力をファイルに記録します
- regex経由でSNMP ENGINEプロセスのPIDを抽出します。
- SNMP PIDを後続のコマンドで使用して、PIDのスタックデータを取得します
- 設定からスクリプトを削除し、スクリプトの実行を停止します。

実装

```
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"
```

## スイッチのアップグレード

このスクリプトは、install add file <file> activate commitコマンドで返される非標準プロンプトでパターン一致を行うように設定され、プロンプトに応答します。トリガーイベントが設定されていないため、event manager run UPGRADEを使用してアップグレードを行う必要がある場合は、ユーザがEEMスクリプトを手動でトリガーする必要があります。install addコマンドの実行には非常に長い時間がかかるため、maxrunタイマーはデフォルト値の20秒ではなく300秒に設定されています。

実装

```
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"
```

## IP SLAトラッキング対象オブジェクトがダウンした場合のファイルへの診断データのダンプ

このスクリプトは、IP SLAオブジェクト11がダウンして次のアクションが実行されたときにトリガーされます。

- MACテーブル、ARPテーブル、syslog、ルーティングテーブルの収集
- 情報をフラッシュ上のファイルに書き込む：sla_track.txt

実装

```
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"
```

## EEMからの電子メールの送信

このスクリプトは、event syslog pattern文に記述されているパターンが見つかるとトリガーされ、次のアクションを実行します。

- 内部Eメールサーバから Eメールを送信します ( 内部Eメールサーバでは、デバイスからのオープン認証が許可されていると想定しています ) 。

実装

```
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 HERE” maxrun 60
action 0010 info type routername
action 0020 mail server "$email_server" to "$email_to" from "$email_from" subject "SUBJECT OF EMAIL - S
```

## スケジュールに従ったポートのシャットダウン

このスクリプトは、ポートTe2/1/15を毎日6PMにシャットダウンします。

### 実装

```
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"
```

## 与えられたパケット/秒(PPS)レートに達した場合のインターフェイスのシャットダウン

このスクリプトは、TX方向のインターフェイスTe2/1/9のPPSレートを1秒ごとにチェックします。PPSレートが100を超えると、次のアクションが実行されます。

- インターフェイスのshow int出力をsyslogに記録します。
    - インターフェイスをシャットダウンします。

#### 実装

```
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 entry-type value
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"
```

#### 関連情報

- **Cisco EEMのベストプラクティス**
- シスコのテクニカルサポートとダウンロード

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。