



## EEM Python モジュール

---

組み込みイベント マネージャ (EEM) ポリシーは、Python スクリプトをサポートします。Python スクリプトは、EEM アプレットで EEM アクションの一部として実行できます。

- [EEM Python モジュールの前提条件 \(1 ページ\)](#)
- [EEM Python モジュールについて \(1 ページ\)](#)
- [EEM Python ポリシーの設定方法 \(4 ページ\)](#)
- [EEM Python モジュールに関するその他の参考資料 \(10 ページ\)](#)
- [EEM Python モジュールの機能情報 \(10 ページ\)](#)

## EEM Python モジュールの前提条件

ゲスト シェルは、コンテナ内で機能する必要があります。ゲスト シェルは、デフォルトでは有効になっていません。詳細については、ゲスト シェル機能の説明を参照してください。

## EEM Python モジュールについて

### EEM の Python スクリプト

組み込みイベント マネージャ (EEM) ポリシーは、Python スクリプトをサポートします。Python スクリプトを EEM ポリシーとして登録し、対応するイベントが発生したときに、登録済みの Python スクリプトを実行することができます。EEM Python スクリプトには、EEM TCL ポリシーと同じイベント仕様の構文があります。

設定済みの EEM ポリシーは、ゲストシェル内で実行します。ゲストシェルは、仮想化された Linux ベースの環境であり、Cisco デバイスの自動制御と管理のための Python アプリケーションを含む、カスタム Linux アプリケーションを実行するように設計されています。ゲストシェル コンテナは、Python インタープリタを提供します。

## EEM Python パッケージ

EEM Python パッケージを Python スクリプトにインポートすると、EEM に固有の拡張機能を実行できます。



- (注) EEM Python パッケージは、EEM Python スクリプト内でのみ使用できます（パッケージは EEM に登録でき、スクリプトの最初の行に EEM イベント仕様が記載されます）。標準的な Python スクリプト（Python スクリプト名を使用して実行される）では使用できません。

Python パッケージには、次のアプリケーションプログラミングインターフェイス（API）が含まれています。

- アクション API : EEM アクションを実行するもので、デフォルトのパラメータがありません。
- CLI 実行 API : IOS コマンドを実行し、出力を返します。次に、CLI 実行 API のリストを示します。
  - eem\_cli\_open()
  - eem\_cli\_exec()
  - eem\_cli\_read()
  - eem\_cli\_read\_line()
  - eem\_cli\_run()
  - eem\_cli\_run\_interactive()
  - eem\_cli\_read\_pattern()
  - eem\_cli\_write()
  - eem\_cli\_close()
- 環境変数にアクセスする API : 組み込みまたはユーザ定義の変数のリストを取得します。次に、環境変数にアクセスする API を示します。
  - eem\_event\_reqinfo () : 組み込み変数のリストを返します。
  - eem\_user\_variables() : 引数の現在の値を返します。

## Python がサポートする EEM アクション

Python パッケージ（EEM スクリプト内でのみ使用可能で、標準的な Python スクリプトでは使用不可）では、次の EEM アクションをサポートしています。

- Syslog メッセージの印刷
- SNMP トラップの送信

- ボックスのリロード
- スタンバイ デバイスへの切り替え
- ポリシーの実行
- ट्रラック オブジェクトの読み取り
- ट्रラック オブジェクトセット
- Cisco ネットワーキング サービスのイベントの生成

EEM Python パッケージは、EEM アクションを実行するため、インターフェイスを公開します。これらのアクションは Python スクリプトを使用して呼び出すことができ、Cisco Plug N Play (PnP) 経由で Python パッケージからアクションハンドラに転送されます。

## EEM 変数

EEM ポリシーは、次の種類の変数を持つことができます。

- イベント固有の組み込み変数：ポリシーをトリガーしたイベントの詳細が設定される事前定義の変数のセット。eem\_event\_reqinfo () API は、組み込み変数のリストを返します。これらの変数は、ローカルマシンに保存してローカル変数として使用することができます。ローカル変数に対する変更は、組み込み変数に反映されません。
- ユーザ定義の変数：定義およびポリシーでの使用が可能な変数。これらの変数の値は、Python スクリプト内で参照できます。スクリプトを実行する際に、変数の最新の値が使用可能であることを確認してください。eem\_user\_variables() API は、API で入力された引数の現在の値を返します。

## EEM CLI ライブラリのコマンド拡張

EEM 内では、Python スクリプトを動作させるため、次の CLI ライブラリ コマンドを使用できます。

- eem\_cli\_close() : EXEC プロセスをクローズし、コマンドに接続された、VTY および指定されたチャンネルハンドラをリリースします。
- eem\_cli\_exec : 指定されたチャンネルハンドラにコマンドを記述し、コマンドを実行します。次に、チャンネルからコマンドの出力を読み取り、出力を返します。
- eem\_cli\_open : VTY を割り当て、EXEC CLI セッションを作成し、VTY をチャンネルハンドラに接続します。チャンネルハンドラを含む配列を返します。
- eem\_cli\_read() : 読み取られている内容でデバイスプロンプトのパターンが発生するまで、指定された CLI のチャンネルハンドラからコマンド出力を読み取ります。一致するまで、読み取られたすべての内容を返します。
- eem\_cli\_read\_line() : 指定された CLI のチャンネルハンドラから、コマンド出力の 1 行を読み取ります。読み取られた行を返します。

- `eem_cli_read_pattern()` : 読み取られている内容でパターンが発生するまで、指定された CLI のチャンネルハンドラからコマンド出力を読み取ります。一致するまで、読み取られたすべての内容を返します。
- `eem_cli_run()` : `clist` にある項目を繰り返し、それぞれが、イネーブルモードで実行されるコマンドであることを前提とします。正常に実行されると、実行されたすべてのコマンドの出力を返します。失敗すると、エラーを返します。
- `eem_cli_run_interactive()` : 3つの項目がある `clist` のサブリストを用意します。正常に実行されると、実行されたすべてのコマンドの出力を返します。失敗すると、エラーを返します。可能な場合には、配列も使用します。予測と応答を別々に保持することによって、より簡単に後で読み取ることができます。
- `eem_cli_write()` : 指定された CLI チャンネルハンドラに対して実行されるコマンドを書き込みます。CLI チャンネルハンドラによって、コマンドが実行されます。

## EEM Python ポリシーの設定方法

Python スクリプトが動作できるようにするには、ゲストシェルを有効化する必要があります。詳細については、「ゲストシェル」の章を参照してください。

## Python ポリシーの登録

### 手順の概要

1. `enable`
2. `configure terminal`
3. `event manager directory user policy path`
4. `event manager policy policy-filename`
5. `exit`
6. `show event manager policy registered`
7. `show event manager history events`

### 手順の詳細

	コマンドまたはアクション	目的
ステップ 1	<b>enable</b> 例 : Device> enable	特権 EXEC モードを有効にします。 • パスワードを入力します (要求された場合)。
ステップ 2	<b>configure terminal</b> 例 : Device# configure terminal	グローバル コンフィギュレーション モードを開始します。

	コマンドまたはアクション	目的
ステップ 3	<b>event manager directory user policy path</b> 例： <pre>Device(config)# event manager directory user policy flash:/user_library</pre>	ユーザ ライブラリ ファイルまたはユーザ定義 EEM ポリシーの保存に使用するディレクトリを指定します。 (注) 指定されたパスにポリシーが必要です。たとえば、この手順では、 <code>eem_script.py</code> ポリシーが <code>flash:/user_library</code> フォルダまたはパスで使用できます。
ステップ 4	<b>event manager policy policy-filename</b> 例： <pre>Device(config)# event manager policy eem_script.py</pre>	EEM ポリシーを EEM に登録します。 <ul style="list-style-type: none"> <li>• ポリシーは、ファイル拡張子に基づいて解析されます。ファイル拡張子は <code>.py</code> で、ポリシーは Python ポリシーとして登録されます。</li> <li>• EEM は、ポリシーそのものに含まれるイベント仕様に基づいてポリシーをスケジューリングし、実行します。<b>event manager policy</b> コマンドが呼び出されると、EEM はポリシーを確認し、指定されたイベントが発生した場合に実行されるように登録します。</li> </ul>
ステップ 5	<b>exit</b> 例： <pre>Device(config)# exit</pre>	グローバル コンフィギュレーション モードを終了し、特権 EXEC モードに戻ります。
ステップ 6	<b>show event manager policy registered</b> 例： <pre>Device# show event manager policy registered</pre>	保留 EEM ポリシーを表示します。
ステップ 7	<b>show event manager history events</b> 例： <pre>Device# show event manager history events</pre>	トリガーされた EEM イベントを表示します。

### 例

次に、**show event manager policy registered** コマンドの出力例を示します。

```
Device# show event manager policy registered
```

```
No.  Class      Type      Event Type      Trap  Time Registered      Name
1   script    user      multiple        Off   Tue Aug 2 22:12:15 2016  multi_1.py
1:  syslog: pattern {COUNTER}
2:  none:  policyname {multi_1.py} sync {yes}
trigger delay 10.000
correlate event 1 or event 2
attribute tag 1 occurs 1
```

```

nice 0 queue-priority normal maxrun 100.000 scheduler rp_primary Secu none

2  script      user      multiple          Off   Tue Aug 2 22:12:20 2016  multi_2.py
1: syslog: pattern {COUNTER}
2: none: policyname {multi_2.py} sync {yes}
trigger
  correlate event 1 or event 2
nice 0 queue-priority normal maxrun 100.000 scheduler rp_primary Secu none

3  script      user      multiple          Off   Tue Aug 2 22:13:31 2016  multi.tcl
1: syslog: pattern {COUNTER}
2: none: policyname {multi.tcl} sync {yes}
trigger
  correlate event 1 or event 2
  attribute tag 1 occurs 1
nice 0 queue-priority normal maxrun 100.000 scheduler rp_primary Secu none

```

## EEM アプレットアクションの一部としての Python スクリプトの実行

### Python スクリプト : eem\_script.py

アクションコマンドを使用することで、EEM アプレットに Python スクリプトを含めることができます。この例では、ユーザは標準 Python スクリプトを EEM アクションの一部として実行しようとしています。ただし、EEM Python パッケージは標準 Python スクリプトでは使用できません。IOS の標準 Python スクリプトには `from cli import cli,clip` という名前のパッケージがあり、そのパッケージは IOS コマンドを実行するために使用できます。

```

import sys
from cli import cli,clip,execute,executep,configure,configurep

intf= sys.argv[1:]
intf = ''.join(intf[0])

print ('This script is going to unshut interface %s and then print show ip interface
brief'%intf)

if intf == 'loopback55':
configurep(["interface loopback55","no shutdown","end"])
else :
cmd='int %s,no shut ,end' % intf
configurep(cmd.split(','))

executep('show ip interface brief')

```

次に、`guestshell run python` コマンドの出力例を示します。

```

Device# guestshell run python /flash/eem_script.py loop55

This script is going to unshut interface loop55 and then print show ip interface brief
Line 1 SUCCESS: int loop55
Line 2 SUCCESS: no shut

```

```

Line 3 SUCCESS: end
Interface IP-Address OK? Method Status Protocol
Vlan1 unassigned YES NVRAM administratively down down
GigabitEthernet0/0 5.30.15.37 YES NVRAM up up
GigabitEthernet1/0/1 unassigned YES unset down down
GigabitEthernet1/0/2 unassigned YES unset down down
GigabitEthernet1/0/3 unassigned YES unset down down
GigabitEthernet1/0/4 unassigned YES unset up up
GigabitEthernet1/0/5 unassigned YES unset down down
GigabitEthernet1/0/6 unassigned YES unset down down
GigabitEthernet1/0/7 unassigned YES unset down down
GigabitEthernet1/0/8 unassigned YES unset down down
GigabitEthernet1/0/9 unassigned YES unset down down
GigabitEthernet1/0/10 unassigned YES unset down down
GigabitEthernet1/0/11 unassigned YES unset down down
GigabitEthernet1/0/12 unassigned YES unset down down
GigabitEthernet1/0/13 unassigned YES unset down down
GigabitEthernet1/0/14 unassigned YES unset down down
GigabitEthernet1/0/15 unassigned YES unset down down
GigabitEthernet1/0/16 unassigned YES unset down down
GigabitEthernet1/0/17 unassigned YES unset down down
GigabitEthernet1/0/18 unassigned YES unset down down
GigabitEthernet1/0/19 unassigned YES unset down down
GigabitEthernet1/0/20 unassigned YES unset down down
GigabitEthernet1/0/21 unassigned YES unset down down
GigabitEthernet1/0/22 unassigned YES unset down down
GigabitEthernet1/0/23 unassigned YES unset up up
GigabitEthernet1/0/24 unassigned YES unset down down
GigabitEthernet1/1/1 unassigned YES unset down down
GigabitEthernet1/1/2 unassigned YES unset down down
GigabitEthernet1/1/3 unassigned YES unset down down
GigabitEthernet1/1/4 unassigned YES unset down down
Tel1/1/1 unassigned YES unset down down
Tel1/1/2 unassigned YES unset down down
Tel1/1/3 unassigned YES unset down down
Tel1/1/4 unassigned YES unset down down
Loopback55 10.55.55.55 YES manual up up

Device#
Jun 7 12:51:20.549: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback55,
changed state to up
Jun 7 12:51:20.549: %LINK-3-UPDOWN: Interface Loopback55, changed state to up

```

次に示すのは、syslog へのメッセージ出力のサンプル スクリプトです。このスクリプトは、ファイルに保存され、デバイス上のファイルシステムにコピーされ、イベントマネージャのポリシー ファイルを使用して登録される必要があります。

```

::cisco::eem::event_register_syslog tag "1" pattern COUNTER maxrun 200

import eem
import time

eem.action_syslog("SAMPLE SYSLOG MESSAGE","6","TEST")

```

次に示すのは、EEM 環境変数を出力するサンプル スクリプトです。このスクリプトは、ファイルに保存され、デバイス上のファイルシステムにコピーされ、イベントマネージャのポリシー ファイルを使用して登録される必要があります。

```

::cisco::eem::event_register_syslog tag "1" pattern COUNTER maxrun 200

```

```

import eem
import time

c = eem.env_reqinfo()

print "EEM Environment Variables"
for k,v in c.iteritems():
    print "KEY : " + k + str(" ---> ") + v

print "Built in Variables"
for i,j in a.iteritems():
    print "KEY : " + i + str(" ---> ") + j

```

## EEM アプレットでの Python スクリプトの追加

### 手順の概要

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **event** [*tag event-tag*] **syslog pattern** *regular-expression*
5. **action** *label cli command cli-string*
6. **action** *label cli command cli-string* [ **pattern** *pattern-string* ]
7. **end**
8. **show event manager policy active**
9. **show event manager history events**

### 手順の詳細

	コマンドまたはアクション	目的
ステップ 1	<b>enable</b> 例： Device> enable	特権 EXEC モードを有効にします。  • パスワードを入力します（要求された場合）。
ステップ 2	<b>configure terminal</b> 例： Device# configure terminal	グローバル コンフィギュレーション モードを開始します。
ステップ 3	<b>event manager applet</b> <i>applet-name</i> 例： Device(config)# event manager applet interface_Shutdown	Embedded Event Manager (EEM) にアプレットを登録し、アプレット コンフィギュレーション モードを開始します。
ステップ 4	<b>event</b> [ <i>tag event-tag</i> ] <b>syslog pattern</b> <i>regular-expression</i> 例：	syslog メッセージのパターン一致を実行する正規表現を指定します。



	コマンドまたはアクション	目的
	Device(config-applet)# event syslog pattern "Interface Loopback55, changed state to administratively down"	
ステップ 5	<b>action label cli command cli-string</b>  例： Device(config-applet)# action 0.0 cli command "en"	EEM アプレットがトリガーされたときに実行される IOS コマンドを指定します。
ステップ 6	<b>action label cli command cli-string [ pattern pattern-string ]</b>  例： Device(config-applet)# action 1.0 cli command "guestshell run python3 /bootflash/eem_script.py loop55"	<b>pattern</b> キーワードで指定されるアクションを指定します。  • 次の要請プロンプトに一致する正規表現パターン文字列を指定します。
ステップ 7	<b>end</b>  例： Device(config-applet)# end	アプレット コンフィギュレーション モードを終了し、特権 EXEC モードに戻ります。
ステップ 8	<b>show event manager policy active</b>  例： Device# show event manager policy active	実行している EEM ポリシーを表示します。
ステップ 9	<b>show event manager history events</b>  例： Device# show event manager history events	トリガーされた EEM イベントを表示します。

## 次のタスク

次の例では、タスクに設定されている Python スクリプトをトリガーする方法を示しています。

```
Device(config)# interface loopback 55
Device(config-if)# shutdown
Device(config-if)# end
Device#

Mar 13 10:53:22.358 EDT: %SYS-5-CONFIG_I: Configured from console by console
Mar 13 10:53:24.156 EDT: %LINK-5-CHANGED: Line protocol on Interface Loopback55, changed
state to down
Mar 13 10:53:27.319 EDT: %LINK-3-UPDOWN: Interface Loopback55, changed state to
administratively down
Enter configuration commands, one per line. End with CNTL/Z.
Mar 13 10:53:35.38 EDT: %LINEPROTO-5-UPDOWN: Line protocol on Interface Loopback55,
changed state to up
*Mar 13 10:53:35.39 EDT %LINK-3-UPDOWN: Interface Loopback55, changed state to up
+++ 10:54:33 edi37(default) exec +++
show ip interface br
Interface          IP-Address      OK? Method Status          Protocol
GigabitEthernet0/0/0 unassigned      YES unset  down           down
GigabitEthernet0/0/1 unassigned      YES unset  down           down
GigabitEthernet0/0/2 10.1.1.31      YES DHCP    up             up
```

GigabitEthernet0/0/3	unassigned	YES	unset	down	down
GigabitEthernet0	192.0.2.1	YES	manual	up	up
Loopback55	198.51.100.1	YES	manual	up	up
Loopback66	172.16.0.1	YES	manual	up	up
Loopback77	192.168.0.1	YES	manual	up	up
Loopback88	203.0.113.1	YES	manual	up	up

## EEM Python モジュールに関するその他の参考資料

### 関連資料

関連項目	マニュアルタイトル
EEM 設定	『 <a href="#">Embedded Event Manager Configuration Guide</a> 』
EEM コマンド	『 <a href="#">Embedded Event Manager Command Reference</a> 』
ゲスト シェル設定	『 <a href="#">ゲスト シェル</a> 』

### シスコのテクニカル サポート

説明	リンク
<p>シスコのサポート Web サイトでは、シスコの製品やテクノロジーに関するトラブルシューティングにお役立ていただけるように、マニュアルやツールをはじめとする豊富なオンラインリソースを提供しています。</p> <p>お使いの製品のセキュリティ情報や技術情報を入手するために、Cisco Notification Service (Field Notice からアクセス)、Cisco Technical Services Newsletter、Really Simple Syndication (RSS) フィードなどの各種サービスに加入できます。</p> <p>シスコのサポート Web サイトのツールにアクセスする際は、Cisco.com のユーザ ID およびパスワードが必要です。</p>	<a href="http://www.cisco.com/support">http://www.cisco.com/support</a>

## EEM Python モジュールの機能情報

次の表に、このモジュールで説明した機能に関するリリース情報を示します。この表は、ソフトウェア リリース トレインで各機能のサポートが導入されたときのソフトウェア リリースだけを示しています。その機能は、特に断りがない限り、それ以降の一連のソフトウェア リリースでもサポートされます。

プラットフォームのサポートおよびシスコソフトウェアイメージのサポートに関する情報を検索するには、Cisco Feature Navigator を使用します。Cisco Feature Navigator にアクセスするには、[www.cisco.com/go/cfn](http://www.cisco.com/go/cfn) に移動します。Cisco.com のアカウントは必要ありません。

表 1: EEM Python モジュールの機能情報

機能名	リリース	機能情報
EEM Python モジュール	Cisco IOS XE Everest 16.5.1a Cisco IOS XE Everest 16.5.1b	この機能は、EEM ポリシーとして Python スクリプトをサポートします。 追加された新規コマンドはありません。 Cisco IOS XE Everest 16.5.1a では、この機能は次のプラットフォームに実装されていました。 <ul style="list-style-type: none"> <li>• Cisco Catalyst 3650 シリーズ スイッチ</li> <li>• Cisco Catalyst 3850 シリーズ スイッチ</li> <li>• Cisco Catalyst 9300 シリーズ スイッチ</li> <li>•</li> </ul> Cisco IOS XE Everest 16.5.1b では、この機能は次のプラットフォームに実装されていました。 <ul style="list-style-type: none"> <li>• Cisco ISR 4000 シリーズ サービス統合型ルータ</li> </ul>
	Cisco IOS XE Everest 16.6.2	この機能は、Cisco IOS XE Everest 16.6.2 で、Cisco Catalyst 9400 シリーズ スイッチに実装されました。
	Cisco IOS XE Fuji 16.8.1a	Cisco IOS XE Fuji 16.8.1a では、この機能は Cisco Catalyst 9500 ハイパフォーマンス シリーズ スイッチに実装されていました。



## 翻訳について

このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。