Writing Embedded Event Manager Policies Using Tcl

この章では、ソフトウェア開発者が Tool command language（Tcl）スクリプトを使用して Embedded Event Manager（EEM）ポリシーを記述およびカスタマイズし、Cisco ソフトウェアの障害とイベントを処理できるようにする方法について説明します。EEM は、定義済みの Application Programming Interface（API）を介してレポートされる Cisco ソフトウェア システムの障害による、ポリシー方式のプロセスです。EEM ポリシー エンジンは、障害およびその他のイベントが発生したときに通知を受け取ります。EEM ポリシーは、システムの現在の状態に基づいて回復を実行し、該当するイベントのポリシーに指定されたアクションを実行します。回復アクションはポリシーが実行されたときにトリガーされます。

Tcl を使用した Embedded Event Manager ポリシーの記述に関する前提条件

- EEM ポリシーを記述するには、その前に「Embedded Event Manager Overview」の章を理解しておく必要があります。
- コマンドライン インターフェイス（CLI）コマンドを使用して EEM ポリシーを記述するときは、「Writing Embedded Event Manager Policies Using the Cisco IOS CLI」の章をよく理解しておいてください。

Tcl を使用した Embedded Event Manager ポリシー記述について

EEM ポリシー

EEM では、イベントをモニタし、イベント発生が検出されたとき、およびしきい値を超えたときに、情報通知や是正などの任意のアクションを実施できます。EEM ポリシーは、イベントおよびイベントが発生した場合に行う処理を定義するエンティティです。EEM ポリシーにはアプレットとスクリプトの 2 つのタイプがあります。アプレットは、コマンドライン インターフェイス（CLI）設定に定義された、ポリシーの単純な形式です。スクリプトは、Tool Command Language（Tcl）で記述されたポリシーの形式です。

EEM アプレット

EEM アプレットは、イベント スクリーニング基準とイベント発生時に実行するアクションを定義する簡潔な方法です。EEM アプレット コンフィギュレーション モードでは、3 種類のコンフィギュレーション文がサポートされます。event コマンドを使用して実行するアプレットをトリガーするイベント基準を指定し、action コマンドを使用して、EEM アプレットがトリガーされるときに実行されるアクションを指定し、**set** コマンドを使用して EEM アプレット変数の値を設定します。現在、_exit_status 変数だけが、**set** コマンドでサポートされます。

アプレット コンフィギュレーションでは、event コンフィギュレーション コマンドを 1 つだけ使用できます。アプレット コンフィギュレーション サブモードが終了し、event コマンドが存在しない場合は、アプレットにイベントが割り当てられていないことを示す警告が表示されます。イベントが指定されない場合、アプレットは登録されたと見なされません。アプレットにアクションが割り当てられない場合、イベントはトリガーされますが、アクションは実行されません。1 つのアプレット コンフィギュレーション内で複数の action コンフィギュレーション コマンドが使用できます。登録済みのアプレットを表示するには、**show event manager policy registered** コマンドを使用します。

EEM アプレットを修正する前に、アプレット コンフィギュレーション モードを終了するまで既存のアプレットを置き換えられないことに注意してください。アプレットコンフィギュレーションモードでアプレットを修正中であっても、既存のアプレットを実行できます。変更は一時ファイルに書き込まれるため、登録を解除しないでアプレットを変更するのが安全です。アプレット コンフィギュレーション モードを終了すると、古いアプレットが登録解除され、新しいバージョンが登録されます。

アプレット内の action コンフィギュレーション コマンドは、*label* 引数を使用することで一意に識別できます。label 引数には任意の文字列値が使用できます。アクションは、*label* 引数をソートキーとして、アプレット内で英数字のキーの昇順に並べ替えられ、この順序で実行されます。同じ *label* 引数を異なるアプレットで使用できます。ラベルは 1 つのアプレット内でのみ一意にする必要があります。

Embedded Event Manager 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。アプレット コンフィギュレーション モードが終了するとき、EEM は、入力された event コマンドと action コマンドを検査し、指定されたイベントの発生時に実行されるようにアプレットを登録します。

Cisco IOS CLI を使用して EEM ポリシーを記述する方法については、「Writing Embedded Event Manager Policies Using the Cisco IOS CLI」の章を参照してください。

EEM スクリプト

すべての Embedded Event Manager スクリプトは、Tcl で記述されます。Tcl は文字列ベースのコマンド言語で、実行時に解釈されます。Tclがサポートされるバージョンは、Tclバージョン 8.3.4 に、スクリプト サポートが追加されたものです。スクリプトは、ネットワーキング デバイスではなく、別のデバイスで、ASCIIエディタを使用して定義されます。続いてスクリプトはネットワーキング デバイスにコピーされ EEM に登録されます。Tcl スクリプトは EEM でサポートされます。強制適用される規則としての Embedded Event Manager ポリシーは、経過時間 20 秒未満で解釈および実行される必要がある、存続時間の短い実行時ルーチンです。20 秒よりも長い経過時間が必要な場合、event_register 文で maxrun パラメータを使用して、必要な値を指定する必要があります。

EEM ポリシーでは、すべての Tcl 言語機能が使用されます。ただし、シスコでは、EEM ポリシーの記述に活用できる Tcl コマンド拡張の形式で、Tcl 言語の機能を拡張しています。Tcl コマンド拡張のキーワードの主要なカテゴリでは、検出されたイベント、後続のアクション、ユーティリティ情報、カウンタの値、システム情報が特定されます。

EEM では、Tcl を使用して独自のポリシーを記述、実装できます。EEM スクリプトの記述には、次の作業が含まれます。

- ポリシーの実行時に決定に使用される基準を確立する、イベント Tcl コマンド拡張の選択。
- イベントの検出に関連付けられているイベント ディテクタ オプションの定義。
- 検出されたイベントのリカバリまたは検出されたイベントに対する応答を実装するアクションを選択すること。

EEM ポリシーの Tcl コマンド拡張のカテゴリ

EEM ポリシーの Tcl コマンド拡張には、さまざまなカテゴリがあります。

（注） すべての EEM ポリシーで使用するこれらの各カテゴリで使用可能な Tcl コマンドは、この資料の以降の項で説明します。

表 *1 : EEM* ポリシーの *Tcl* コマンド拡張のカテゴリ

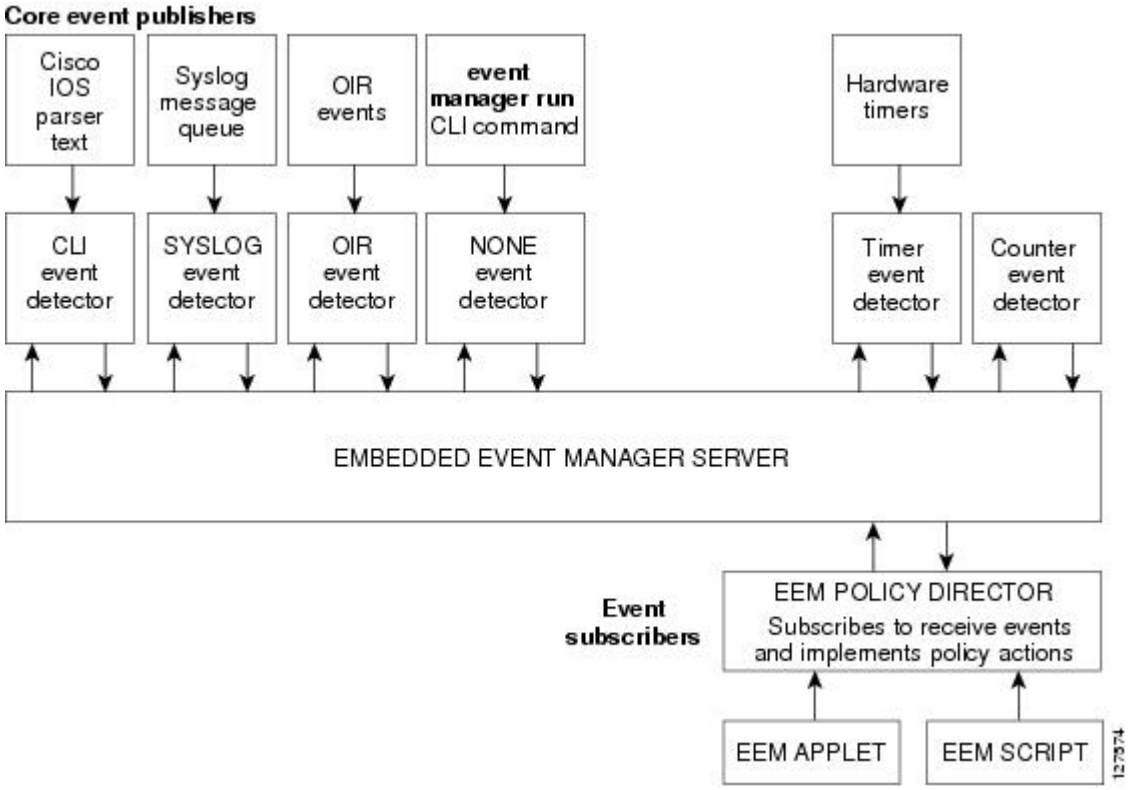
カテゴリ	定義
EEM イベントの Tcl コマンド拡張（イベント情報、イベント登録、イベント パブリッシュの 3 タイプ）	このカテゴリは、イベント固有のコマンドの **event_register_** *xxx* ファミリによって表されます。このカテゴリには、別のイベント情報 Tcl コマンド拡張の **event_reqinfo** もあります。これは、イベントについての情報を EEM に問い合わせるためにポリシーで使用されるコマンドです。アプリケーション固有のイベントをパブリッシュする、EEM イベントパブリッシュ Tcl コマンド拡張 **event_publish**> もあります。
EEM アクションの Tcl コマンド拡張	これらの Tcl コマンド拡張（たとえば、**action_syslog** など）は、イベントまたは障害への応答か、あるいは、イベントまたは障害からの回復のために、ポリシーによって使用されます。これらの拡張に加え、開発者は、Tcl 言語を使用して、必要なアクションを実装できます。
EEM ユーティリティの Tcl コマンド拡張	これらの Tcl コマンド拡張は、アプリケーション情報、カウンタ、またはタイマーの取得、保存、設定、または変更に使用されます。
EEM システム情報の Tcl コマンド拡張	このカテゴリは、システム固有の情報コマンドの **sys_reqinfo** _ *xxx* ファミリによって表されます。これらのコマンドは、システム情報を収集する目的で、ポリシーによって使用されます。
EEM コンテキストの Tcl コマンド拡張	これらの Tcl コマンド拡張は、Tcl コンテキスト（可視変数およびその値）の保存および取得に使用されます。

EEM イベントの検出および回復の一般的なフロー

EEM は、イベント ディテクタと呼ばれるソフトウェア エージェントを使用してシステム内の異なるコンポーネントのモニタリングをサポートする、柔軟でポリシードリブンのフレームワークです。次の図に、EEM サーバ、コア イベント パブリッシャ（イベント ディテクタ）、およびイベント サブスクライバ（ポリシー）の関係を示します。基本的に、イベント パブリッシャはイベントをスクリーニングして、イベント サブスクライバから提供されたイベント仕様に一致したときにイベントをパブリッシュします。イベント ディテクタは、注目するイベントが発生したときに EEM サーバに通知します。

イベントまたは障害が検出されると、Embedded Event Manager によって、たとえば次の図の OIR イベント パブリッシャなどのイベント パブリッシャから、検出された障害またはイベントの登録が発生しているかどうかが判断されます。EEM によって、イベント登録情報が、イベント データそのものと、照会されます。ポリシーによって、検出されたイベントが Tcl コマンド拡張 **event_register** _ *xxx* で登録されます。イベント情報 Tcl コマンド拡張 **event_reqinfo** は、検出されたイベントに関する情報について Embedded Event Manager に問い合わせるために、ポリシーで使用されます。

図 *1 : Embedded Event Manager* コア イベント ディテクタ



Safe-Tcl

Safe-Tcl は、安全モードで作成されたインタープリタで、非信頼 Tcl スクリプトを実行できる、安全メカニズムです。安全インタープリタには、一部のシステム リソースへのアクセスや、ホストおよび他のアプリケーションに害が及ぼされることを防ぐ、制限されたコマンドのセットがあります。たとえば、コマンドは、重要な Cisco IOS ファイル システム ディレクトリにはアクセスできません。

シスコ定義のスクリプトはフル Tcl モードで実行されますが、ユーザ定義のスクリプトは Safe-Tcl モードで実行されます。Safe-Tcl を使用すると、シスコでは、個々の Tcl コマンドのディセーブルまたはカスタマイズを行えます。Tcl コマンドの詳細については、http://www.tcl.tk/man/ を参照してください。

次のリストにある Tcl コマンドは、一部の例外によって制約されます。各コマンドまたはコマンド キーワードに対する制約事項は、次のとおりです。

- **cd** ：制約付きの Cisco ディレクトリ名の 1 つへのディレクトリ移動はできません。
- **-- encoding** コマンド **names** 、 **encoding** 、 **convertfrom**および **encoding** が許可されます **convertto** 。 **encoding** 引数のない **encoding system** コマンドは許可されています が、**?encoding?** キーワードを使用した **encoding system** コマンドは使用できません。
- **exec** ：使用できません。

- **fconfigure** ：使用できます。
- **file** ：以下は使用できます。
  - **file dirname**
  - **file exists**
  - **file extension**
  - **file isdirectory**
  - **file join**
  - **file pathtype**
  - **file rootname**
  - **file split**
  - **file stat**
  - **file tail**
- **file** ：以下は使用できません。
  - **file atime**
  - **file attributes**
  - **file channels**
  - **file copy**
  - **file delete**
  - **file executable**
  - **file isfile**
  - **file link**
  - **file lstat**
  - **file mkdir**
  - **file mtime**
  - **file nativename**
  - **file normalize**
  - **file owned**
  - **file readable**
  - **file readlink**
  - **file rename**
  - **file rootname**
  - **file separator**
  - **file size**
  - **file system**
  - **file type**
  - **file volumes**
  - **file writable**
- **glob** ：制約付きの Cisco ディレクトリの 1 つで検索する場合、**glob** コマンドは使用できません。これ以外の場合は使用できます。
- **load** ：ユーザ ポリシー ディレクトリまたはユーザ ライブラリ ディレクトリにあるファイルのみがロードできます。静的パッケージ（たとえば、C コードで構成されるライブラリ）は、**load** コマンドではロードできません。

- **open** ：**open** コマンドは、制約付きの Cisco ディレクトリの 1 つにあるファイルでは使用できません。
- **pwd** ：**pwd** コマンドは使用できません。
- **socket** ：**socket** コマンドは使用できます。
- **source** ：**source** コマンドは、ユーザポリシーディレクトリまたはユーザ ライブラリ ディレクトリにあるファイルで使用できます。

EEM 2.4 のバイトコード サポート

EEM 2.4 で、標準バイトコード スクリプト拡張子 .tbc のファイルを受け付けることによって、Bytecode Language（BCL）サポートが導入されています。Tcl バージョン 8.3.4 では、BCL が定義され、Tcl スクリプトが BCL に変換されるコンパイラが含まれています。EEM 2.4 のユーザ ポリシーおよびシステム ポリシーで有効な EEM ポリシーのファイル拡張子は、.tcl（Tcl テキスト ファイル）と .tbc（Tcl バイトコード ファイル）です。

バイトコードの Tcl スクリプトを格納すると、ポリシーの実行速度が向上します。これは、コードが事前にコンパイルされ、ポリシーサイズが小さくなり、コードを隠蔽するためです。難読化はスクリプトの変更を若干難しくし、論理を隠して知的財産権を保護します。

サポート コードおよび信頼済みコードのリリースのために別のオプションを提供するため、バイトコードのサポートが追加されています。十分に理解しているソフトウェア、信頼できるソフトウェア、またはサポートされているソフトウェアのみをネットワーク デバイスで実行することを推奨します。IOS EEM サポートの Tcl バイトコードを生成するには、TclPro バージョン 1.4 または 1.5 を使用します。

Tcl スクリプトをバイトコードに変換するには、procomp、Free TclPro Compiler の一部、または Active State Tcl Development Kit を使用できます。Tcl スクリプトを procomp を使用してコンパイルする場合、コードはスクランブルされ、.tbc ファイルが生成されます。バイトコード ファイルはプラットフォームに依存せず、Windows、Linux、および UNIX などの、TclPro を使用できるすべてのオペレーティング システムで生成できます。Procomp は TclPro の一部であり、http://www.tcl.tk/software/tclpro で入手できます。

登録の置き換え

通常の Tcl の置き換えの他に、EEM 2.3 では、EEM イベント登録ステートメントの行内の個別のパラメータを環境変数に置き換えることができます。

EEM 2.4 では、イベント登録ステートメントの行にある複数パラメータを 1 つの環境変数で置き換える機能が導入されています。

（注） 1 つめの環境変数のみで、複数パラメータの置き換えがサポートされます。個別のパラメータを指定することも引き続き可能です。それを行うには最初の変数の後に追加の環境変数を追加します。

置き換えを示すために、1 つの環境変数 $_eem_syslog_statement が次のとおりに設定されています。

```
::cisco::eem::event_register_syslog pattern COUNT
```

登録の置き換えを使用すると、$_eem_syslog_statement 環境変数が、次の EEM ユーザ ポリシーで使用されます。

```
$_eem_syslog_statement occurs $_eem_occurs_val
action_syslog “this is test 3”
```

環境変数は、それらを使用するポリシーを登録する前に定義しておく必要があります。$_eem_syslog_statement 環境変数を定義するには、次を実行します。

```
Device(config)# event manager environment eem_syslog_statement
::cisco::eem::event_register_syslog pattern COUNT
Device(config)# event manager environment eem_occurs_val 2
```

EEM 用のシスコ ファイル命名規則

すべての Embedded Event Manager ポリシー名、ポリシー サポート ファイル（たとえば、E メール テンプレート ファイル）、およびライブラリ ファイル名は、シスコのファイル命名規則に従う必要があります。このため、Embedded Event Manager ポリシー ファイル名は、次の仕様に従っています。

- オプションのプレフィックス Mandatory. がある場合、これは、システム ポリシーがまだ登録されていない場合に、自動的に登録される必要があるシステム ポリシーであることを示します。たとえば、Mandatory.sl_text.tcl などです。
- 指定された 1 つめのイベントの 2 文字の省略形が含まれるファイル名の本体部（下の表を参照）、下線部、および、ポリシーをさらに示す説明フィールド部。
- ファイル名拡張子部は .tcl と定義されます。

Embedded Event Manager の E メール テンプレート ファイルは、email_template のファイル名のプレフィックスと、後続の E メール テンプレートの使用状況を示す省略形で構成されます。

Embedded Event Manager ライブラリ ファイル名は、ライブラリの使用状況を示す説明フィールドを含むファイル名の本体部と、後続の _lib、および .tcl というファイル名拡張子で構成されます。

表 *2:2* 文字の省略形の指定

ap	event_register_appl
cl	event_register_cli
ct	event_register_counter
go	event_register_gold
if	event_register_interface

io	event_register_ioswdsysmon
la	event_register_ipsla
nf	event_register_nf
no	event_register_none
oi	event_register_oir
pr	event_register_process
rf	event_register_rf
rs	event_register_resource
rt	event_register_routing
rp	event_register_rpc
sl	event_register_syslog
sn	event_register_snmp
st	event_register_snmp_notification
so	event_register_snmp_object
tm	event_register_timer
tr	event_register_track
ts	event_register_timer_subscriber
wd	event_register_wdsysmon

Tcl を使用した Embedded Event Manager ポリシーの記述方法

EEM Tcl スクリプトの登録と定義

環境変数を設定し、EEMポリシーを登録するには、この作業を実行します。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。EEM ポリシーが登録されると、ソフトウェアによって、ポリシーが調べられ、指定されたイベントの発生時に実行されるよう、登録されます。

始める前に

Tcl スクリプト言語で記述されたポリシーが使用できる状態である必要があります。サンプル ポリシーを示します。使用している Cisco IOS リリースのイメージで使用可能なポリシーについては、EEM サンプル ポリシー （20 ページ）を参照してください。これらのサンプル ポリシーは、システム ポリシー ディレクトリに保存されています。

手順

	コマンドまたはアクション	目的
ステップ 1	**enable** 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	**show event manager environment** [**all**| *variable-name*] 例： `Device# show event manager environment all`	（任意）EEM 環境変数の名前と値を表示します。 • オプションの **all** キーワードは、すべての EEM 環境変数を表示します。 • オプションの *variable-name* 引数は、指定された環境変数に関する情報を表示します。
ステップ 3	**configure terminal** 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 4	**event manager environment** *variable-name string* 例： `Device(config)# event manager environment _cron_entry 0-59/2 0-23/1 * * 0-6`	指定された EEM 環境変数の値を設定します。 • この例では、ソフトウェアによって、CRON タイマー環境変数が、毎日、毎時の 2 分目に設定されます。
ステップ 5	ステップ 4 を繰り返して、ステップ 6 で登録されるポリシーに必要なすべての環境変数を設定します。	--
ステップ 6	**event manager policy** *policy-filename* [**type** {**system**| **user**}] [**trap**] 例：	ポリシー内で定義された指定イベントが発生した場合に、EEM ポリシーを実行するよう、定義します。

	コマンドまたはアクション	目的
	`Device(config)# event manager policy tm_cli_cmd.tcl type system`	• **system** キーワードを使用して、シスコ定義のシステムポリシーを登録します。 • **user** キーワードを使用して、ユーザ定義のシステムポリシーを登録します。 • **trap** キーワードを使用して、ポリシーがトリガーされた場合の SNMP トラップを生成します。 • この例では、tm_cli_cmd.tcl という名前の EEM サンプル ポリシーが、システム ポリシーとして定義されます。
ステップ 7	**exit** **例**： `Device(config)# exit`	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。

例

次に、**show event manager environment** 特権 EXEC コマンドを使用して、すべての EEM 環境変数の名前と値を表示する例を示します。

```
Device# show event manager environment all
No.  Name                           Value
1    _cron_entry                    0-59/2 0-23/1 * * 0-6
2    _show_cmd                      show ver
3    _syslog_pattern                .*UPDOWN.*Ethernet1/0.*
4    _config_cmd1                   interface Ethernet1/0
5    _config_cmd2                   no shut
```

登録済みの EEM ポリシーの表示

登録済みの EEM ポリシーを表示するには、次の任意の作業を実行します。

手順

ステップ 1 **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）。

例：

```
Device> enable
```

ステップ 2 **show event manager policy registered** [**event-type** *event-name*] [**time-ordered**| **name-ordered**] [**detailed** *policy-filename*]

このコマンドを **time-ordered** キーワードとともに使用して、現在登録されているポリシーの情報を時間でソートして表示します。次に例を示します。

例：

```
Device# show event manager policy registered time-ordered
No.  Type    Event Type          Trap  Time Registered           Name
1    system  timer cron          Off   Wed May11  01:43:18 2005  tm_cli_cmd.tcl
 name {crontimer2} cron entry {0-59/1 0-23/1 * * 0-7}
 nice 0 priority normal maxrun 240
2    system  syslog              Off   Wed May11  01:43:28 2005  sl_intf_down.tcl
 occurs 1 pattern {.*UPDOWN.*Ethernet1/0.*}
 nice 0 priority normal maxrun 90
3    system  proc abort          Off   Wed May11  01:43:38 2005  pr_cdp_abort.tcl
 instance 1 path {cdp2.iosproc}
 nice 0 priority normal maxrun 20
```

このコマンドを **name-ordered** キーワードとともに使用して、現在登録されているポリシーの情報を名前でソートして表示します。次に例を示します。

例：

```
Device# show event manager policy registered name-ordered
No.  Type    Event Type          Trap  Time Registered           Name
1    system  proc abort          Off   Wed May11  01:43:38 2005  pr_cdp_abort.tcl
 instance 1 path {cdp2.iosproc}
 nice 0 priority normal maxrun 20
2    system  syslog              Off   Wed May11  01:43:28 2005  sl_intf_down.tcl
 occurs 1 pattern {.*UPDOWN.*Ethernet1/0.*}
 nice 0 priority normal maxrun 90
3    system  timer cron          Off   Wed May11  01:43:18 2005  tm_cli_cmd.tcl
 name {crontimer2} cron entry {0-59/1 0-23/1 * * 0-7}
 nice 0 priority normal maxrun 240
```

このコマンドを **event-type** キーワードとともに使用して、*event-name* 引数で指定されたイベントタイプの現在登録されているポリシーに関する情報を表示します。次に例を示します。

例：

```
Device# show event manager policy registered event-type syslog
No.  Type    Event Type          Time Registered           Name
1    system  syslog              Wed May11  01:43:28 2005  sl_intf_down.tcl
 occurs 1 pattern {.*UPDOWN.*Ethernet1/0.*}
 nice 0 priority normal maxrun 90
```

EEM ポリシーの登録解除

EEM ポリシーを実行コンフィギュレーション ファイルから削除するには、次の作業を実行します。ポリシーの実行はキャンセルされます。

手順

	コマンドまたはアクション	目的
ステップ 1	**enable** 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	**show event manager policy registered** [**event-type** *event-name*][**system**| **user**] [**time-ordered**| **name-ordered**] [**detailed** *policy-filename*] 例： `Device# show event manager policy registered`	（任意）現在登録されている EEM ポリシーを表示します。 • オプションの **system** キーワードまたは **user** キーワードによって、登録済みのシステム ポリシーまたはユーザ ポリシーが表示されます。 • キーワードが指定されない場合は、すべてのイベント タイプに対する登録された EEM ポリシーが時間順に表示されます。
ステップ 3	**configure terminal** 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 4	**no event manager policy** *policy-filename* 例： `Device(config)# no event manager policy pr_cdp_abort.tcl`	ポリシーを登録解除するために EEM ポリシーを設定から削除します。 • この例では、コマンドの **no** 形式を使用して、指定されたポリシーの登録が解除します。
ステップ 5	**exit** 例： `Device(config)# exit`	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	EEM ポリシーの登録解除 を繰り返して、ポリシーが削除されたことを確認します。	--

	コマンドまたはアクション	目的
	例： `Device# show event manager policy registered`	

例

次に、**show event manager policy registered** 特権 EXEC コマンドを使用して、現在登録されている 3 個の EEM ポリシーを表示する例を示します。

```
Device# show event manager policy registered
No.  Type    Event Type          Trap  Time Registered           Name
1    system  timer cron          Off   Tue Oct11  01:43:18 2005 tm_cli_cmd.tcl
 name {crontimer2} cron entry {0-59/1 0-23/1 * * 0-7}
 nice 0 priority normal maxrun 240.000
2    system  syslog              Off   Tue Oct11  01:43:28 2005 sl_intf_down.tcl
 occurs 1 pattern {.*UPDOWN.*Ethernet1/0.*}
 nice 0 priority normal maxrun 90.000
3    system  proc abort          Off   Tue Oct11  01:43:38 2005 pr_cdp_abort.tcl
 instance 1 path {cdp2.iosproc}
 nice 0 priority normal maxrun 20.000
```

現在のポリシーが表示されたら、**no** 形式の **event manager policy** コマンドを使用して pr_cdp_abort.tcl ポリシーの削除が決定されます。

```
Device# configure terminal
Device(config)# no event manager policy pr_cdp_abort.tcl
Device(config)# exit
```

show event manager policy registered 特権 EXEC コマンドを再度入力すると、現在登録されている EEM ポリシーが表示されます。ポリシー pr_cdp_abort.tcl は、登録されていません。

```
Device# show event manager policy registered
No.  Type    Event Type          Trap  Time Registered           Name
1    system  timer cron          Off   Tue Oct11  01:45:17 2005 tm_cli_cmd.tcl
 name {crontimer2} cron entry {0-59/1 0-23/1 * * 0-7}
 nice 0 priority normal maxrun 240.000
2    system  syslog              Off   Tue Oct11  01:45:27 2005 sl_intf_down.tcl
 occurs 1 pattern {.*UPDOWN.*Ethernet1/0.*}
 nice 0 priority normal maxrun 90.000
```

EEM ポリシー実行の一時停止

すべての EEM ポリシーの実行をただちに一時停止するには、次の作業を実行します。一時的なパフォーマンスまたはセキュリティ面での理由から、ポリシーの登録解除ではなく一時停止が必要なことがあります。

手順

	コマンドまたはアクション	目的
ステップ 1	**enable** 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	**show event manager policy registered** [**event-type** *event-name*][**system**| **user**] [**time-ordered**| **name-ordered**] [**detailed** *policy-filename*] 例： `Device# show event manager policy registered`	（任意）現在登録されている EEM ポリシーを表示します。 • オプションの **system** キーワードまたは **user** キーワードによって、登録済みのシステム ポリシーまたはユーザ ポリシーが表示されます。 • キーワードが指定されない場合は、すべてのイベント タイプに対する登録された EEM ポリシーが時間順に表示されます。
ステップ 3	**configure terminal** 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 4	**event manager scheduler suspend** 例： `Device(config)# event manager scheduler suspend`	すべての EEM ポリシーの実行がすぐに一時停止されます。
ステップ 5	**exit** 例： `Device(config)# exit`	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。

例

次に、**show event manager policy registered** 特権 EXEC コマンドを使用して、EEM のすべての登録済みポリシーを表示する例を示します。

```
Device# show event manager policy registered
No.  Type    Event Type          Trap  Time Registered           Name
1    system  timer cron          Off   Sat Oct11  01:43:18 2003  tm_cli_cmd.tcl
 name {crontimer2} cron entry {0-59/1 0-23/1 * * 0-7}
 nice 0 priority normal maxrun 240.000
2    system  syslog              Off   Sat Oct11  01:43:28 2003  sl_intf_down.tcl
```

```
 occurs 1 pattern {.*UPDOWN.*Ethernet1/0.*}
 nice 0 priority normal maxrun 90.000
3    system  proc abort          Off   Sat Oct11  01:43:38 2003  pr_cdp_abort.tcl
 instance 1 path {cdp2.iosproc}
 nice 0 priority normal maxrun 20.000
```

すべての EEM ポリシーの実行をすぐに一時停止するには、**event manager scheduler suspend** コマンドを入力します。

```
Device# configure terminal
Device(config)# event manager scheduler suspend
*Nov  2 15:34:39.000: %HA_EM-6-FMS_POLICY_EXEC: fh_io_msg: Policy execution has been
suspended
```

EEM ポリシーの管理

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するには、この作業を実行します。

（注） この作業は、Tcl スクリプトを使用して記述される EEM ポリシーのみに適用されます。

手順

	コマンドまたはアクション	目的
ステップ 1	**enable** 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	**show event manager directory user [library| policy]** 例： `Device# show event manager directory user library`	（任意）EEM ユーザ ライブラリまたはポリシーファイルの保存に使用するディレクトリを表示します。 • オプションの **library** キーワードによって、ユーザライブラリファイルに使用されるディレクトリが表示されます。 • オプションの **policy** キーワードによって、ユーザ定義 EEM ポリシーに使用されるディレクトリが表示されます。
ステップ 3	**configure terminal** 例：	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
	`Device# configure terminal`	

ステップ 4	**event manager directory user** {**library** *path*| **policy** *path*}
	例：
	`Device(config)# event manager directory user library disk0:/user_library`
	`Device(config)# event manager directory user library bootflash:/user_library`
		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。
		• ユーザ ディレクトリへの絶対パス名を指定するには、*path* 引数を指定します。

ステップ 5	**exit** 例： `Device(config)# exit`	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。

例

次に、**show event manager directory user** 特権 EXEC コマンドを使用して、EEM ユーザライブラリファイルの保存に使用されるディレクトリがある場合に、そのディレクトリを表示する例を示します。

```
Device# show event manager directory user library
disk0:/user_library

Device# show event manager directory user library
bootflash:/user_library
```

履歴テーブル サイズの変更と EEM 履歴データの表示

履歴テーブルのサイズを変更し、EEM 履歴データを表示するには、次の任意の作業を実行します。

手順

ステップ 1 **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）。

例：

```
Device> enable
```

ステップ 2 **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。

例：

```
Device# configure terminal
```

ステップ 3 **event manager history size** {**events** | **traps**} [*size*]

このコマンドを使用して、EEM イベント履歴テーブルのサイズ、または、EEM SNMP トラップ履歴テーブルのサイズを変更します。次に、EEM イベント履歴テーブルのサイズを 30 エントリに変更する例を示します。

例：

```
Device(config)# event manager history size events 30
```

ステップ 4 **exit**

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。

例：

```
Device(config)# exit
```

ステップ 5 **show event manager history events** [**detailed**] [**maximum** *number*]

このコマンドを使用して、トリガーされた各 EEM イベントについての情報を表示します。

例：

```
Device# show event manager history events
No.  Time of Event             Event Type          Name
1    Fri Sep  9 13:48:40 2005  syslog              applet: one
2    Fri Sep  9 13:48:40 2005  syslog              applet: two
3    Fri Sep  9 13:48:40 2005  syslog              applet: three
4    Fri Sep  9 13:50:00 2005  timer cron          script: tm_cli_cmd.tcl
5    Fri Sep  9 13:51:00 2005  timer cron          script: tm_cli_cmd.tcl
```

ステップ 6 **show event manager history traps** [**server** | **policy**]

このコマンドを使用して、EEM サーバまたは EEM ポリシーのいずれかから送信された EEM SNMP トラップを表示します。

例：

```
Device# show event manager history traps
No.  Time                      Trap Type           Name
1    Fri Sep  9 13:48:40 2005  server              applet: four
2    Fri Sep  9 13:57:03 2005  policy              script: no_snmp_test.tcl
```

EEM を使用したソフトウェア モジュール方式プロセスの信頼性メトリック

Cisco IOS ソフトウェアモジュール方式プロセスの信頼性メトリックを表示するには、この任意の作業を実行します。この **show event manager metric processes** コマンド拡張は、ソフトウェアモジュール方式イメージでのみサポートされます。

手順

ステップ 1 **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）。

例：

```
Device> enable
```

ステップ 2 **show event manager metric process** {**all**| *process-name*}

このコマンドを使用して、プロセスの信頼性メトリック データを表示します。システムでは、プロセスの開始時と終了時にレコードが保存され、このデータが、信頼性分析の基本データとして使用されます。この部分の例では、システムに挿入されているすべてのカード上でのプロセスのメトリック データを示す、最初と最後のエントリが表示されます。

例：

```
Device# show event manager metric process all
====================================
process name: devc-pty, instance: 1
sub_system id: 0, version: 00.00.0000
--------------------------------
last event type: process start
recent start time: Fri Oct10  20:34:40 2005
recent normal end time: n/a
recent abnormal end time: n/a
number of times started: 1
number of times ended normally: 0
number of times ended abnormally: 0
most recent 10 process start times:
--------------------------
Fri Oct10  20:34:40 2005
--------------------------
most recent 10 process end times and types:
cumulative process available time: 6 hours 30 minutes 7 seconds 378 milliseconds
cumulative process unavailable time: 0 hours 0 minutes 0 seconds 0 milliseconds
process availability:  0.100000000
number of abnormal ends within the past 60 minutes (since reload): 0
number of abnormal ends within the past 24 hours (since reload): 0
number of abnormal ends within the past 30 days (since reload): 0
.
.
.
====================================
process name: cdp2.iosproc, instance: 1
sub_system id: 0, version: 00.00.0000
```

```
--------------------------------
last event type: process start
recent start time: Fri Oct10  20:35:02 2005
recent normal end time: n/a
recent abnormal end time: n/a
number of times started: 1
number of times ended normally: 0
number of times ended abnormally: 0
most recent 10 process start times:
--------------------------
Fri Oct10  20:35:02 2005
--------------------------
most recent 10 process end times and types:

cumulative process available time: 6 hours 29 minutes 45 seconds 506 milliseconds
cumulative process unavailable time: 0 hours 0 minutes 0 seconds 0 milliseconds
process availability:  0.100000000
number of abnormal ends within the past 60 minutes (since reload): 0
number of abnormal ends within the past 24 hours (since reload): 0
number of abnormal ends within the past 30 days (since reload): 0
```

トラブルシューティングのヒント

特権 EXEC モードで **debug event manager** コマンドを使用して、EEM コマンド操作のトラブルシューティングを行います。デバッグ コマンドは注意して使用してください。生成される出力量によってデバイスの動作が遅くなったり、停止したりすることがあります。シスコエンジニアの管理下に限ってこのコマンドを使用することを推奨します。

EEM サンプル ポリシーの変更

サンプル ポリシーの 1 つを変更するには、この作業を実行します。Cisco ソフトウェアには、Embedded Event Manager が含まれるイメージに、いくつかのサンプル ポリシーが含まれています。EEM ポリシーの開発者は、ポリシーが実行されるイベントと、イベントの記録および応答に関連付けられているオプションを、カスタマイズすることによって、これらのポリシーを変更できます。さらに、開発者は、ポリシーの実行時に実装されるアクションを選択できます。

EEM サンプル ポリシー

シスコには、次の表に示されているように、サンプル ポリシーのセットが含まれています。ユーザは、サンプル ポリシーをユーザ ディレクトリにコピーし、ポリシーを変更するか、または、独自にポリシーを記述することができます。現時点でポリシー作成のためにシスコでサポートされているスクリプト言語は、Tcl だけです。Tcl ポリシーは、Emacs などのテキストエディタを使用して変更できます。ポリシーは、定義されている経過時間の秒数以内で実行する必要があり、時間変数はポリシー内で設定できます。現在のデフォルト値は 20 秒です。

次の表で、サンプル EEM ポリシーについて説明します。

表 *3 : EEM* サンプル ポリシーの説明

ポリシーの名前	説明
pr_cdp_abort.tcl	Cisco ソフトウェア モジュラリティ イメージを使用して導入されました。このポリシーでは、cdp2.iosproc プロセスの強制終了イベントがモニタされます。SYSLOG にメッセージが記録され、強制終了の詳細が E メールで送信されます。
pr_crash_reporter.tcl	Cisco ソフトウェア モジュラリティ イメージを使用して導入されました。このポリシーでは、すべてのプロセスの強制終了イベントがモニタされます。イベントが発生すると、ポリシーによって、クラッシュダンプ ファイルを含むクラッシュ情報が、CGI スクリプトによってデータが処理される指定された URL に、送信されます。
pr_iprouting_abort.tcl	Cisco ソフトウェア モジュラリティ イメージを使用して導入されました。このポリシーでは、iprouting.iosproc プロセスの強制終了イベントがモニタされます。SYSLOG にメッセージが記録され、強制終了の詳細が E メールで送信されます。
sl_intf_down.tcl	このポリシーは、設定可能な Syslog メッセージが記録されるときに実行されます。設定可能な CLI コマンドが実行され、結果が E メールで送信されます。
tm_cli_cmd.tcl	このポリシーは、設定可能な CRON エントリを使用して実行されます。設定可能な CLI コマンドが実行され、結果が E メールで送信されます。
tm_crash_history.tcl	Cisco ソフトウェア モジュラリティ イメージを使用して導入されました。このポリシーは、毎日夜中に実行され、指定された E メール アドレスにプロセス クラッシュ履歴レポートが E メールで送信されます。
tm_crash_reporter.tcl	このポリシーは、登録後 5 秒間実行されます。ポリシーが設定に保存される場合、デバイスがリロードされるたびにも実行されます。ポリシーによって、リロード理由を示すプロンプトが表示されます。クラッシュが原因でリロードされる場合、ポリシーによって最新の crashinfo ファイルが検索され、この情報が指定された URL に送信されます。
tm_fsys_usage.tcl	Cisco ソフトウェア モジュラリティ イメージを使用して導入されました。このポリシーは、設定可能な CRON エントリを使用して実行され、ディスク領域の使用状況がモニタされます。ディスク領域の使用状況が、設定可能なしきい値を超えると、Syslog メッセージが表示されます。
wd_mem_reporter.tcl	Cisco ソフトウェア モジュラリティ イメージを使用して導入されました。使用可能なメモリ容量が、使用可能な初期システム メモリの 20% を下回った場合、このポリシーによって、システム メモリ低下の状態がレポートされます。Syslog メッセージが表示され、オプションで、E メールが送信されます。

使用可能なサンプル ポリシーおよびその実行方法についての詳細は、EEM イベント ディテクタのデモの例 （40 ページ）を参照してください。

手順

ステップ 1 **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）。

例：

```
Device> enable
```

ステップ 2 **show event manager policy available detailed** *policy-filename*

ポリシーによって使用される環境変数と、ポリシーの実行方法の説明の詳細を含む、指定された実際のサンプル ポリシーを表示します。**detailed** キーワードが **show event manager policy available** コマンドと **show event manager policy registered** コマンドに導入されました。お使いのリリースによっては、2 つの Tcl スクリプトのいずれかをこのドキュメントの設定例セクションからコピーしなければならない場合があります（Tcl のサンプル スクリプトを使用したポリシーのプログラミングの例 （49 ページ）を参照）。次に、サンプル ポリシー tm_cli_cmd.tcl についての詳細が画面上に表示される例を示します。

例：

```
Device# show event manager policy available detailed tm_cli_cmd.tcl
```

ステップ 3 画面に表示されたサンプル ポリシーの内容を、テキスト エディタにカット アンド ペーストします。

編集機能とコピー機能を使用して、デバイスから別のデバイス上のテキスト エディタに、内容を移動します。

ステップ 4 ポリシーを編集し、新しいファイル名で保存します。

テキスト エディタを使用して、ポリシーを Tcl スクリプトとして変更します。ファイルの命名規則については、EEM 用のシスコ ファイル命名規則 （8 ページ）を参照してください。

ステップ 5 新しいファイルを、デバイスのフラッシュ メモリにコピーして戻します。

デバイスのフラッシュ ファイル システム（通常は disk0:）にファイルをコピーします。ファイルのコピーの詳細については、『*Configuration Fundamentals Configuration Guide*』の「Using the Cisco IOS File System」の章を参照してください。

デバイスのフラッシュ ファイル システム（通常は bootflash:）にファイルをコピーします。ファイルのコピーの詳細については、『*Configuration Fundamentals Configuration Guide*』の「Using the Cisco IOS File System」の章を参照してください。

ステップ 6 **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。

例：

```
Device# configure terminal
```

ステップ 7 **event manager directory user** {**library** *path*| **policy** *path*}

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。次に、disk0 の user_library ディレクトリが、ユーザ ライブラリ ファイルを保存するディレクトリとして指定されます。

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。次に、bootflash の user_library ディレクトリが、ユーザ ライブラリ ファイルを保存するディレクトリとして指定されます。

例：

```
Device(config)# event manager directory user library disk0:/user_library
```

```
Device(config)# event manager directory user library bootflash:/user_library
```

ステップ 8 **event manager policy** *policy-filename* [**type** {**system**| **user**}] [**trap**]

ポリシー内で定義された指定イベントが発生した場合に、EEM ポリシーを実行するよう、定義します。次に、test.tcl という名前の EEM ポリシーが、ユーザ定義ポリシーとして登録される例を示します。

例：

```
Device(config)# event manager policy test.tcl type user
```

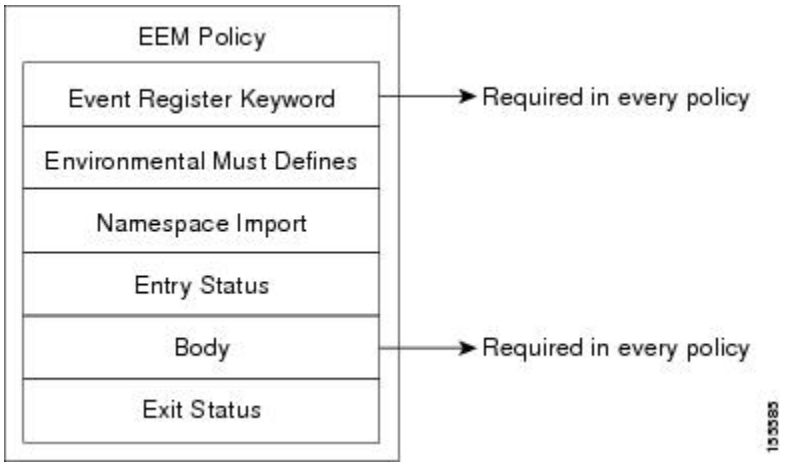
Tcl を使用した EEM ポリシーのプログラミング

Tcl コマンド拡張を使用してポリシーをプログラムするには、この作業を実行します。既存のポリシーをコピーし、変更することを推奨します。EEM Tcl ポリシーには、**event_register** Tcl コマンド拡張と本体の 2 つの必須部分が存在する必要があります。Tcl ポリシーの構造と要件（23 ページ） の概念にある他のすべてのセクションは、オプションです。

Tcl ポリシーの構造と要件

すべての EEM ポリシーでは、次の図に示されているように、同じ構造が共有されます。EEM ポリシーには、**event_register** Tcl コマンド拡張と本体という 2 つの必須部分が存在します。ポリシーの残りの部分の、環境定義必須、名前空間のインポート、開始ステータス、および終了ステータスは、オプションです。

図 *2: Tcl* ポリシーの構造と要件



各ポリシーの開始部分では、**event_register** Tcl コマンド拡張を使用して検出するイベントを記述し登録する必要があります。ポリシーのこの部分によって、ポリシーの実行がスケジュールされます。次に、**event_register_timer** Tcl コマンド拡張を登録する Tcl コードの例を示します。

```
::cisco::eem::event_register_timer cron name crontimer2 cron_entry $_cron_entry maxrun
240
```

環境定義必須セクションはオプションで、環境変数の定義が含まれます。次に、一部の環境変数をチェックし、定義する Tcl コードの例を示します。

```
# Check if all the env variables that we need exist.
# If any of them does not exist, print out an error msg and quit.
if {![info exists _email_server]} {
    set result \
      "Policy cannot be run: variable _email_server has not been set"
    error $result $errorInfo
}
if {![info exists _email_from]} {
    set result \
      "Policy cannot be run: variable _email_from has not been set"
    error $result $errorInfo
}
if {![info exists _email_to]} {
    set result \
      "Policy cannot be run: variable _email_to has not been set"
    error $result $errorInfo
```

名前空間のインポートセクションはオプションで、コードライブラリが定義されます。次に、名前空間インポート セクションを設定する Tcl コードの例を示します。

```
namespace import ::cisco::eem::*
namespace import ::cisco::lib::*
```

ポリシーの本体は必須の構造で、次のものを含める必要があります。

- 検出されたイベントに関する情報の EEM への問い合わせに使用される **event_reqinfo** イベント情報の Tcl コマンド拡張。

- EEM 特有のアクションの指定に使用される、**action_syslog** などのアクション Tcl コマンド拡張。
- 一般的なシステム情報の取得に使用される、**sys_reqinfo_routername** などのシステム情報の Tcl コマンド拡張。
- ポリシーからの、SMTP ライブラリ（電子メール通知を送信）または CLI ライブラリ（CLI コマンドを実行）の使用。
- 他のポリシーによって使用される Tcl 変数の保存に使用される **context_save** および **context_retrieve** の Tcl コマンド拡張。

次に、イベントを問い合わせ、本体セクションの一部としてメッセージを記録するコードの Tcl コードの例を示します。

```
# Query the event info and log a message.
array set arr_einfo [event_reqinfo]

if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}

global timer_type timer_time_sec
set timer_type $arr_einfo(timer_type)
set timer_time_sec $arr_einfo(timer_time_sec)

# Log a message.
set msg [format "timer event: timer type %s, time expired %s" \
        $timer_type [clock format $timer_time_sec]]

action_syslog priority info msg $msg
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
      $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
```

EEM 開始ステータス

EEM ポリシーの開始ステータスの部分は、前のポリシーが同じイベントに対して実行されたかどうかや、前のポリシーの終了ステータスを特定するために、使用されます。_entry_status 変数が定義されている場合、このイベントに対して前のポリシーがすでに実行されています。_entry_status 変数の値によって、前のポリシーの戻りコードが特定されます。

開始ステータス指定には、0（前のポリシーが正常終了した）、Not=0（前のポリシーに障害が発生した）、および Undefined（実行された前のポリシーがない）の、3 つの値のうちいずれか 1 つを使用できます。

EEM 終了ステータス

ポリシーでそのコードの実行を終了すると、終了値が設定されます。終了値は、Embedded Event Manager によって使用され、このイベントのデフォルトアクションがある場合に、それが適用

されたかどうかが判断されます。ゼロの値は、デフォルトアクションが実行されていないことを意味します。ゼロではない値は、デフォルトアクションが実行されたことを意味します。終了ステータスは、同じイベントで実行される後続ポリシーに渡されます。

EEM ポリシーと Cisco エラー番号

一部の EEM Tcl コマンド拡張によって、Cisco エラー番号の Tcl グローバル変数の _cerrno が設定されます。_cerrno が設定されるたびに、他の 4 つの Tcl グローバル変数が _cerrno から分岐し、それとともに設定されます（_cerr_sub_num、_cerr_sub_err、_cerr_posix_err、および _cerr_str）。

たとえば、次の例の **action_syslog** コマンドでは、コマンド実行の副次的な影響としてこれらのグローバル変数が設定されます。

```
action_syslog priority warning msg “A sample message generated by action_syslog”
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
```

_cerrno：32 ビット エラー戻り値

コマンドによって設定された _cerrno は、次の形式の 32 ビットの整数を表す場合があります。

```
XYSSSSSSSSSSSSSEEEEEEEEPPPPPPPPP
```

たとえば、次のエラー戻り値は、EEM Tcl コマンド拡張から戻される場合があります。

```
862439AE
```

この数字は、次の 32 ビット値として解釈されます。

```
10000110001001000011100110101110
```

この 32 ビットの整数は、次の表に示されているように、5 つの変数に分けられます。

表 *4 : _cerrno*：*32* ビット エラー戻り値の変数

変数	説明
XY	エラー クラス（エラーの重大度を示します）。この変数は、32 ビットのエラー戻り値の最初の 2 ビットに対応しています。前述のケースの 10 は、CERR_CLASS_WARNING を示します。 この変数固有の 4 つのエラー クラス エンコーディングについては、次の表を参照してください。
SSSSSSSSSSSSS	最新のエラーが生成されたサブシステム番号（13 ビット＝値 8192）。これは、32 ビット シーケンスの次の 13 ビットで、その整数値は $_cerr_sub_num に含まれています。
変数	説明

変数	説明
EEEEEEEE	サブシステム固有のエラー番号（8 ビット＝値 256）。このセグメントは、32 ビット シーケンスの次の 8 ビットで、このエラー番号に対応する文字列は、$_cerr_sub_err に含まれています。
PPPPPPPP	パススルー POSIX エラー コード（9 ビット＝値 512）。これは、32 ビット シーケンスの最後で、このエラー コードに対応する文字列は、$_cerr_posix_err に含まれています。

XY のエラー クラス エンコーディング

最初の変数 XY は、次の表に示されているように、エラー クラス エンコーディングを参照しています。

表 *5:* エラー クラス エンコーディング

00	CERR_CLASS_SUCCESS
01	CERR_CLASS_INFO
10	CERR_CLASS_WARNING
11	CERR_CLASS_FATAL

ゼロのエラー戻り値は、SUCCESS を示します。

手順

ステップ 1 **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）。

例：

```
Device> enable
```

ステップ 2 **show event manager policy available detailed** *policy-filename*

ポリシーによって使用される環境変数と、ポリシーの実行方法の説明の詳細を含む、指定された実際のサンプル ポリシーを表示します。**detailed** キーワードが **show event manager policy available** コマンドと **show event manager policy registered** コマンドに導入されました。お使いのリリースによっては、2 つの Tcl スクリプトのいずれかをこのドキュメントの設定例セクションからコピーする必要があります（Tcl のサンプル スクリプトを使用したポリシーのプログラミングの例 （49 ページ）を参照）。次に、サンプル ポリシー tm_cli_cmd.tcl についての詳細が画面上に表示される例を示します。

例：

```
Device# show event manager policy available detailed tm_cli_cmd.tcl
```

ステップ 3 画面に表示されたサンプル ポリシーの内容を、テキスト エディタにカット アンド ペーストします。

編集機能とコピー機能を使用して、デバイスから別のデバイス上のテキスト エディタに、内容を移動します。テキスト エディタを使用して、ポリシーを Tcl スクリプトとして編集します。

ステップ 4 必要な **event_register** Tcl コマンド拡張を定義します。

検出するイベントについて、適切な **event_register** Tcl コマンド拡張を次の表から選択し、ポリシーに追加します。

表 *6 : EEM* イベント登録の *Tcl* コマンド拡張

イベント登録の Tcl コマンド拡張
event_register_appl
event_register_cli
event_register_counter
event_register_gold
event_register_interface
event_register_ioswdsysmon
event_register_ipsla
event_register_nf
event_register_none
event_register_oir
event_register_process
event_register_resource
event_register_rf
event_register_routing
event_register_rpc
event_register_snmp
event_register_snmp_notification
event_register_snmp_object
event_register_syslog
event_register_timer
event_register_timer_subscriber

イベント登録の Tcl コマンド拡張
event_register_track
event_register_wdsysmon

ステップ 5 適切な名前空間を、::cisco 階層構造に追加します。

ポリシーの開発者は、Cisco IOS EEM によって使用されるすべての拡張をグループ化するため、Tcl ポリシーで新しい名前空間 ::cisco を使用できます。::cisco 階層構造の下には、2 つの名前空間があります。次の表に、各名前空間の下に属する EEM Tcl コマンド拡張のカテゴリを示します。

表 *7 : Cisco IOS EEM* 名前空間グルーピング

Namespace	Tcl コマンド拡張のカテゴリ
::cisco::eem	EEM イベント登録
	EEM イベント情報
	EEM イベント パブリッシュ
	EEM アクション
	EEM ユーティリティ
	EEM コンテキスト ライブラリ
	EEM システム情報
	CLI ライブラリ
::cisco::lib	SMTP ライブラリ

（注） 前述のコマンドの使用時に、適切な名前空間をインポートするか、または、認定コマンド名を使用します。

ステップ 6 Must Define セクションをプログラムし、このポリシーで使用される各環境変数をチェックします。

この手順は任意です。Must Define は、ポリシーによって必要とされるすべての EEM 環境変数が、回復アクションの実行前に定義されているかどうかをテストする、ポリシーのセクションです。ポリシーによって EEM 環境変数が使用されない場合、Must Define セクションは不要です。EEM スクリプトの EEM 環境変数は、ポリシーの実行前にポリシーに対して外部定義された Tcl グローバル変数です。EEM 環境変数を定義するには、Embedded Event Manager コンフィギュレーション コマンド **event manager environment** CLI コマンドを使用します。規則として、すべてのシスコ EEM 環境変数の先頭は、「_」（アンダースコア）になっています。将来的な競合を避けるため、「_」で始まる新しい変数を定義しないことを推奨します。

（注） **show event manager environment** 特権 EXEC コマンドを使用して、システムの Embedded Event Manager 環境変数セットを表示できます。

たとえば、サンプル ポリシーで定義されている Embedded Event Manager 環境変数には、E メール変数が含まれます。適切に動作させるためには、電子メールを送信するサンプル ポリシーに、次の表に示す変数が設定されている必要があります。

次の表で EEM サンプル ポリシーで使用される電子メール特有の環境変数について説明します。

表 *8*:サンプル ポリシーで使用される電子メール特有の環境変数

環境変数	説明	例
_email_server	E メール送信に使用されるシンプル メール転送プロトコル（SMTP）メール サーバ。	E メール サーバ名は、次のテンプレート形式のいずれかで使用できます。 • username:password@host • username@host • ホスト
_email_to	E メールの送信先アドレス。	engineering@example.com
_email_from	E メールの送信元アドレス。	devtest@example.com
_email_cc	E メールのコピーの送信先アドレス。	manager@example.com

次に、E メール特有の環境変数のプログラムをチェックする Must Define セクションの例を示します。

Must Define の例

例：

```
if {![info exists _email_server]} {
    set result \
        "Policy cannot be run: variable _email_server has not been set"
    error $result $errorInfo
}
if {![info exists _email_from]} {
    set result \
        "Policy cannot be run: variable _email_from has not been set"
    error $result $errorInfo
}
if {![info exists _email_to]} {
    set result \
        "Policy cannot be run: variable _email_to has not been set"
    error $result $errorInfo
}
if {![info exists _email_cc]} {
    set result \
        "Policy cannot be run: variable _email_cc has not been set"
    error $result $errorInfo
}
```

ステップ 7 スクリプトの本体をプログラムします。

スクリプトのこのセクションでは、次のいずれかを定義できます。

- 検出されたイベントに関する情報の EEM への問い合わせに使用される **event_reqinfo** イベント情報の Tcl コマンド拡張。
- EEM 特有のアクションの指定に使用される、**action_syslog** などのアクション Tcl コマンド拡張。
- 一般的なシステム情報の取得に使用される、**sys_reqinfo_routername** などのシステム情報の Tcl コマンド拡張。
- 他のポリシーによって使用される Tcl 変数の保存に使用される **context_save** および **context_retrieve** の Tcl コマンド拡張。
- ポリシーからの、SMTP ライブラリ（電子メール通知を送信）または CLI ライブラリ（CLI コマンドを実行）の使用。

ステップ 8 開始ステータスをチェックし、ポリシーがこのイベントに対して前に実行されたかどうかを判断します。

前のポリシーが正常終了した場合、現在のポリシーは、実行が必要な場合と、実行が不要な場合があります。開始ステータス指定には、0（前のポリシーが正常終了した）、Not=0（前のポリシーに障害が発生した）、および Undefined（実行された前のポリシーがない）の、3 つの値のうちいずれか 1 つを使用できます。

ステップ 9 終了ステータスをチェックし、デフォルト アクションが存在する場合に、このイベントのデフォルト アクションが適用されたかどうかを判断します。

ゼロの値は、デフォルト アクションが実行されていないことを意味します。ゼロではない値は、デフォルトアクションが実行されたことを意味します。終了ステータスは、同じイベントで実行される後続ポリシーに渡されます。

ステップ 10 Cisco エラー番号（_cerrno）の Tcl グローバル変数を設定します。

一部の EEM Tcl コマンド拡張によって、Cisco エラー番号の Tcl グローバル変数の _cerrno が設定されます。_cerrno が設定されるたびに、他の 4 つの Tcl グローバル変数が _cerrno から分岐し、それとともに設定されます（_cerr_sub_num、_cerr_sub_err、_cerr_posix_err、および _cerr_str）。

たとえば、次の例の **action_syslog** コマンドでは、コマンド実行の副次的な影響としてこれらのグローバル変数が設定されます。

例：

```
action_syslog priority warning msg "A sample message generated by action_syslog
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
```

ステップ 11 新しいファイル名で Tcl スクリプトを保存し、Tcl スクリプトをデバイスにコピーします。

Embedded Event Manager ポリシー ファイル名は、次の仕様に従っています。

- オプションのプレフィックス Mandatory. がある場合、これは、システム ポリシーがまだ登録されていない場合に、自動的に登録される必要があるシステム ポリシーであることを示します。たとえば、Mandatory.sl_text.tcl などです。
- 指定された 1 つめのイベントの 2 文字の省略形が含まれるファイル名の本体部（EEM ポリシーと Cisco エラー番号 （26 ページ） を参照）、下線文字部、および、ポリシーをさらに示す説明フィールド部。
- ファイル名拡張子部は .tcl と定義されます。

詳細については、EEM 用のシスコ ファイル命名規則 （8 ページ）を参照してください。

デバイスのフラッシュ ファイル システム（通常は disk0:）にファイルをコピーします。ファイルのコピーの詳細については、『Cisco IOS Configuration Fundamentals Configuration Guide』の「Using the Cisco IOS File System」の章を参照してください。

デバイスのフラッシュ ファイル システム（通常は bootflash:）にファイルをコピーします。ファイルのコピーの詳細については、『Cisco IOS Configuration Fundamentals Configuration Guide』の「Using the Cisco IOS File System」の章を参照してください。

ステップ 12 **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。

例：

```
Device# configure terminal
```

ステップ 13 **event manager directory user** {**library** *path*| **policy** *path*}

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。次に、disk0 の user_library ディレクトリが、ユーザ ライブラリ ファイルを保存するディレクトリとして指定されます。

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。次に、bootflash の user_library ディレクトリが、ユーザ ライブラリ ファイルを保存するディレクトリとして指定されます。

例：

```
Device(config)# event manager directory user library disk0:/user_library
```

```
Device(config)# event manager directory user library bootflash:/user_library
```

ステップ 14 **event manager policy** *policy-filename* [**type** {**system**| **user**}] [**trap**]

ポリシー内で定義された指定イベントが発生した場合に、EEM ポリシーを実行するよう、定義します。次に、cl_mytest.tcl という名前の EEM ポリシーが、ユーザ定義ポリシーとして登録される例を示します。

例：

```
Device(config)# event manager policy cl_mytest.tcl type user
```

ステップ 15 ポリシーを実行し、ポリシーを観察します。

ポリシーの実行をテストするには、ポリシーが実行される原因となる条件を生成し、ポリシーが想定どおりに実行されていることを確認します。

ステップ 16 ポリシーが正しく実行されていない場合、デバッグのテクニックを使用します。

Cisco IOS **debug event manager** CLI コマンドをそのさまざまなキーワードとともに使用して、問題をデバッグします。Tcl 特有のキーワード使用の詳細については、トラブルシューティングのヒント （33 ページ）を参照してください。

トラブルシューティングのヒント

- Tcl 拡張コマンドの問題をデバッグするには、**debug event manager tcl commands** コマンドを使用します。イネーブルの場合、このコマンドによって、CLI のやり取りを処理する TTY セッションに渡され、TTY セッションから読み戻される、すべてのデータが表示されます。このデータを使用すると、ユーザが CLI に渡すコマンドが有効になります。
- CLI ライブラリを使用すると、ユーザは、CLI コマンドを実行し、Tcl のコマンドの出力を取得できます。**debug event manager tcl cli-library** CLI コマンドを使用して、CLI ライブラリの問題をデバッグします。
- SMTP ライブラリを使用すると、ユーザは、SMTP E メール サーバへ、E メール メッセージを送信できます。**debug event manager tcl smtp_library** CLI コマンドを使用して、SMTP ライブラリの問題をデバッグします。イネーブルの場合、このコマンドによって、SMTP ライブラリ ルーチンに渡され、SMTP ライブラリ ルーチンから読み戻される、すべてのデータが表示されます。このデータを使用すると、ユーザが SMTP ライブラリに渡すコマンドが有効になります。
- Tcl は、コマンドを上書きできる融通性のある言語です。たとえば、**set** コマンドを変更し、スカラ変数が設定されたときにメッセージを表示する **set** コマンドのバージョンを作成します。ポリシーに **set** コマンドが入力されると、スカラ変数が設定されたときにはいつでもメッセージが表示され、スカラ変数をデバッグする方法が示されます。このデバッグ テクニックの例を参照するには、Tcl set コマンド操作のトレースの例 （60 ページ）を参照してください。

これらのデバッグ テクニックのいくつかの例を参照するには、Embedded Event Manager ポリシーのデバッグの例 （58 ページ）を参照してください。

EEM ユーザ Tcl ライブラリ索引の作成

Tcl ファイルのライブラリに含まれているすべての手順のディレクトリが含まれている、索引ファイルを作成するには、この作業を実行します。この作業を行うと、EEM Tcl のライブラリ

サポートをテストできます。この作業では、Tcl ライブラリ ファイルが含まれるライブラリ ディレクトリが作成され、ファイルがディレクトリにコピーされ、ライブラリ ファイルにあるすべての手順のディレクトリが含まれる索引 tclIndex が作成されます。索引が作成されない場合、Tcl 手順を参照する EEM ポリシーを実行するときに、Tcl 手順は見つかりません。

手順

ステップ 1 ワークステーション（UNIX、Linux、PC、または Mac）で、ライブラリ ディレクトリを作成し、Tcl ライブラリ ファイルをディレクトリにコピーします。

次の例ファイルを使用すると、Tcl シェルが実行されているワークステーション上で、tclIndex を作成できます。

lib1.tcl

例：

```
proc test1 {} {
    puts "In procedure test1"
}

proc test2 {} {
    puts "In procedure test2"
}
```

lib2.tcl

例：

```
proc test3 {} {
    puts "In procedure test3"
}
```

ステップ 2 **tclsh**

このコマンドを使用して、Tcl シェルを開始します。

例：

```
workstation% tclsh
```

ステップ 3 **auto_mkindex** *directory_name* ***.tcl**

auto_mkindex コマンドを使用して、tclIndex ファイルを作成します。すべての手順のディレクトリが含まれる tclIndex ファイルは、Tcl ライブラリ ファイルに含まれていました。どのディレクトリにも 1 つの tclIndex ファイルのみを存在させることができ、他の Tcl ファイルはグループ化しておくことが可能であるため、ディレクトリ内で auto_mkindex を実行することを推奨します。ディレクトリ内で auto_mkindex を実行すると、特定の tclIndex を使用してどの Tcl ソース ファイルを索引化できるかが判断されます。

例：

```
workstation% auto_mkindex eem_library *.tcl
```

lib1.tcl ファイルと lib2.tcl ファイルがライブラリ ファイル ディレクトリにあり、**auto_mkindex** コマンドが実行されたときに、次の例に示す tclIndex が作成されます。

tclIndex

例：

```
# Tcl autoload index file, version 2.0
# This file is generated by the "auto_mkindex" command
# and sourced to set up indexing information for one or
# more commands.  Typically each line is a command that
# sets an element in the auto_index array, where the
# element name is the name of a command and the value is
# a script that loads the command.

set auto_index(test1) [list source [file join $dir lib1.tcl]]
set auto_index(test2) [list source [file join $dir lib1.tcl]]
set auto_index(test3) [list source [file join $dir lib2.tcl]]
```

ステップ 4 ターゲット デバイス上のユーザ ライブラリ ファイルの保存に使用されるディレクトリに Tcl ライブラリ ファイルと tclIndex ファイルをコピーします。

ステップ 5 Tcl で記述されたユーザ定義 EEM ポリシー ファイルを、ターゲット デバイス上でユーザ定義 EEM ポリシーの保存に使用されるディレクトリにコピーします。

ユーザ定義 EEM ポリシーを保存するディレクトリは、ステップ 4 で使用されるディレクトリと同じディレクトリを使用できます。次に、EEM でサポートされる Tcl ライブラリのテストに、ユーザ定義 EEM ポリシーを使用できる例を示します。

libtest.tcl

例：

```
::cisco::eem::event_register_none

namespace import ::cisco::eem::*
namespace import ::cisco::lib::*

global auto_index auto_path

puts [array names auto_index]

if { [catch {test1} result]} {
    puts "calling test1 failed result = $result $auto_path"
}

if { [catch {test2} result]} {
    puts "calling test2 failed result = $result $auto_path"
}
if { [catch {test3} result]} {
    puts "calling test3 failed result = $result $auto_path"
}
```

ステップ 6 **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）。

例：

```
Device> enable
```

ステップ 7 **configure terminal**

グローバル コンフィギュレーション モードをイネーブルにします。

例：

```
Device# configure terminal
```

ステップ 8 **event manager directory user library** *path*

このコマンドを使用して、EEM ユーザ ライブラリ ディレクトリを指定します。これは、ファイルがコピーされたディレクトリです。

例：

```
Device(config)# event manager directory user library disk2:/eem_library
```

ステップ 9 **event manager directory user policy** *path*

このコマンドを使用して、EEM ユーザ ポリシー ディレクトリを指定します。これは、ファイルがコピーされたディレクトリです。

例：

```
Device(config)# event manager directory user policy disk2:/eem_policies
```

ステップ 10 **event manager policy** *policy-name* **[type {system | user} [trap]**

このコマンドを使用して、ユーザ定義 EEM ポリシーを登録します。この例では、libtest.tcl という名前のポリシーが登録されます。

例：

```
Device(config)# event manager policy libtest.tcl
```

ステップ 11 **event manager run** *policy-name*

このコマンドを使用して、手作業で EEM ポリシーを実行します。この例では、libtest.tcl という名前のポリシーが実行され、EEM の Tcl サポートがテストされます。次に、EEM の Tcl サポートが正常終了した出力例を示します。

例：

```
Device(config)# event manager run libtest.tcl
The following output is displayed:
01:24:37: %HA_EM-6-LOG: libtest.tcl: In procedure test1
01:24:37: %HA_EM-6-LOG: libtest.tcl: In procedure test2
01:24:37: %HA_EM-6-LOG: libtest.tcl: In procedure test3
```

EEM ユーザ Tcl パッケージ索引の作成

すべての Tcl パッケージのディレクトリと、Tcl パッケージ ファイルのライブラリに含まれるバージョン情報が含まれる、Tcl パッケージの索引ファイルを作成するには、この作業を実行

します。使用しているリリースによっては、Tcl **package** キーワードを使用することで Tcl パッケージがサポートされます。

Tcl パッケージは、EEM システム ライブラリ ディレクトリまたは EEM ユーザ ライブラリ ディレクトリのいずれかにあります。**package require** Tcl コマンドが実行されると、ユーザ ライブラリ ディレクトリで、まず、pkgIndex.tcl ファイルが検索されます。pkgIndex.tcl ファイルがユーザディレクトリで見つからない場合、システム ライブラリ ディレクトリが検索されます。この作業では、**pkg_mkIndex** コマンドを使用して、適切なライブラリディレクトリに Tcl パッケージディレクトリ（pkgIndex.tcl ファイル）が作成され、バージョン情報とともに、ディレクトリ内にあるすべての Tcl パッケージについての情報が含められます。索引が作成されない場合、**package require** Tcl コマンドが含まれる、EEM ポリシーが実行されたときに、Tcl パッケージは見つかりません。

EEM で Tcl パッケージ サポートを使用すると、ユーザは、Tcl の XML_RPC などのパッケージにアクセスできます。Tcl パッケージ インデックスが作成されるとき、Tcl スクリプトは、外部エンティティに対する XML-RPC 呼び出しを容易に行うことができます。

（注） C プログラミング コードで実装されるパッケージは、EEM ではサポートされません。

手順

ステップ 1 ワークステーション（UNIX、Linux、PC、または Mac）で、ライブラリ ディレクトリを作成し、Tcl パッケージ ファイルをディレクトリにコピーします。

ステップ 2 **tclsh**

このコマンドを使用して、Tcl シェルを開始します。

例：

```
workstation% tclsh
```

ステップ 3 **pkg_mkindex** *directory_name* ***.tcl**

pkg_mkindex コマンドを使用して、pkgIndex ファイルを作成します。すべてのパッケージのディレクトリが含まれる pkgIndex ファイルは、Tcl ライブラリ ファイルに含まれていました。どのディレクトリにも 1 つの pkgIndex ファイルのみを存在させることができ、他の Tcl ファイルはグループ化しておくことが可能であるため、ディレクトリ内で pkg_mkindex を実行することを推奨します。ディレクトリ内で pkg_mkindex を実行すると、特定の pkgIndex を使用してどの Tcl パッケージ ファイルを索引化できるかが判断されます。

例：

```
workstation% pkg_mkindex eem_library *.tcl
```

次に、いくつかの Tcl パッケージがライブラリ ファイル ディレクトリにあり、**pkg_mkindex** コマンドが実行されたときに、pkgIndex が作成される例を示します。

pkgIndex

例：

```
# Tcl package index file, version 1.1
# This file is generated by the "pkg_mkIndex" command
# and sourced either when an application starts up or
# by a "package unknown" script. It invokes the
# "package ifneeded" command to set up package-related
# information so that packages will be loaded automatically
# in response to "package require" commands. When this
# script is sourced, the variable $dir must contain the
# full path name of this file's directory.
package ifneeded xmlrpc 0.3 [list source [file join $dir xmlrpc.tcl]]
```

ステップ 4 ターゲット デバイス上のユーザ ライブラリ ファイルの保存に使用されるディレクトリに Tcl ライブラリ ファイルと pkgIndex ファイルをコピーします。

ステップ 5 Tcl で記述されたユーザ定義 EEM ポリシー ファイルを、ターゲット デバイス上でユーザ定義 EEM ポリシーの保存に使用されるディレクトリにコピーします。

ユーザ定義 EEM ポリシーを保存するディレクトリは、ステップ 4 で使用されるディレクトリと同じディレクトリを使用できます。次に、EEM でサポートされる Tcl パッケージのテストに、ユーザ定義 EEM ポリシーを使用できる例を示します。

packagetest.tcl

例：

```
::cisco::eem::event_register_none maxrun 1000000.000
#
# test if xmlrpc available
#
#
# Namespace imports
#
namespace import ::cisco::eem::*
namespace import ::cisco::lib::*
#
package require xmlrpc
puts "Did you get an error?"
```

ステップ 6 **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）。

例：

```
Device> enable
```

ステップ 7 **configure terminal**

グローバル コンフィギュレーション モードをイネーブルにします。

例：

```
Device# configure terminal
```

ステップ 8 **event manager directory user library** *path*

このコマンドを使用して、EEM ユーザ ライブラリ ディレクトリを指定します。これは、ファイルがコピーされたディレクトリです。

例：

```
Device(config)# event manager directory user library disk2:/eem_library
```

ステップ 9 **event manager directory user policy** *path*

このコマンドを使用して、EEM ユーザ ポリシー ディレクトリを指定します。これは、ファイルがコピーされたディレクトリです。

例：

```
Device(config)# event manager directory user policy disk2:/eem_policies
```

ステップ 10 **event manager policy** *policy-name* [**type** {**system** | **user**} [**trap**]

このコマンドを使用して、ユーザ定義 EEM ポリシーを登録します。この例では、packagetest.tcl という名前のポリシーが登録されます。

例：

```
Device(config)# event manager policy packagetest.tcl
```

ステップ 11 **event manager run** *policy-name*

このコマンドを使用して、手作業で EEM ポリシーを実行します。この例では、packagetest.tcl という名前のポリシーが実行され、EEM の Tcl パッケージ サポートがテストされます。

例：

```
Device(config)# event manager run packagetest.tcl
```

Tcl を使用した Embedded Event Manager（EEM）ポリシー記述の設定例

Tcl セッションへのユーザ名割り当ての例

次に、Tcl セッションに関連付けられるユーザ名を設定する例を示します。認証、認可、カウンティング（AAA）セキュリティを使用し、コマンドベースで認可を実装する場合、**event manager session cli username** コマンドを使用して、Tcl セッションに関連付けられるユーザ名を設定する必要があります。Tcl ポリシーによって CLI コマンドが実行されるときに、ユーザ名が使用されます。TACACS+ では、ポリシーを実行している Tcl セッションに関連付けられているユーザ名を使用して、各 CLI コマンドが確認されます。ポリシーを登録するには、デバイスが特権 EXEC モードである必要があるため、Tcl ポリシーからのコマンドは、通常、確認

されません。この例では、ユーザ名は yourname で、これは、CLI コマンド セッションが EEM ポリシー内から開始されるたびに使用されるユーザ名です。

```
configure terminal
 event manager session cli username yourname
 end
```

EEM イベント ディテクタのデモの例

EEM サンプル ポリシーの説明

この設定例では、一部の EEM サンプル ポリシーについて説明します。

- ap_perf_test_base_cpu.tcl：EEM ポリシーの CPU パフォーマンスを測定するために実行されます。
- no_perf_test_init.tcl：EEM ポリシーの CPU パフォーマンスを測定するために実行されます。
- sl_intf_down.tcl：設定可能な syslog メッセージが記録されるときに実行されます。最大で 2 つまでの CLI コマンドを実行し、結果が E メールで送信されます。
- tm_cli_cmd.tcl：設定可能な CRON エントリを使用して実行されます。設定可能な CLI コマンドが実行され、結果が電子メールで送信されます。
- tm_crash_reporter.tcl：登録後の 5 秒間と、デバイスの起動後の 5 秒間に実行されます。トリガーされると、スクリプトによって、リロード原因の検索が試行されます。リロードの原因がクラッシュの場合、ポリシーによって、関連する crashinfo ファイルが検索され、環境変数 _crash_reporter_url でユーザによって指定された URL へ、この情報が送信されます。
- tm_fsys_usage.tcl：このポリシーは、設定可能な CRON エントリを使用して実行され、ディスク領域の使用状況を監視します。ディスク領域の使用状況が、設定可能なしきい値を超えると、Syslog メッセージが表示されます。

サンプル ポリシーのイベント マネージャ環境変数

イベント マネージャ環境変数は、ポリシーの登録および実行の前に EEM ポリシーに対して外部定義された Tcl グローバル変数です。サンプル ポリシーでは、3 つの電子メール環境変数が設定されている必要があります。_email_cc のみが省略可能です。他の必須および任意の変数設定については、次の表で説明します。

次の表に、ap_perf_test_base_cpu.tcl サンプル ポリシーの実行前に設定する必要がある EEM 環境変数を示します。

表 *9 : ap_perf_test_base_cpu.tcl* ポリシーで使用される環境変数

環境変数	説明	例
_perf_iterations	測定を反復する回数。	**100**

環境変数	説明	例
_perf_cmd1	測定テストの一部として実行される最初の非インタラクティブ CLI コマンド。この変数は任意で、指定する必要はありません。	**enable**
_perf_cmd2	測定テストの一部として実行される 2 番目の非インタラクティブ CLI コマンド。_perf_cmd2 を使用するには、_perf_cmd1 を定義する必要があります。この変数は任意で、指定する必要はありません。	**show version**
_perf_cmd3	測定テストの一部として実行される 3 番目の非インタラクティブ CLI コマンド。_perf_cmd3 を使用するには、_perf_cmd1 を定義する必要があります。この変数は任意で、指定する必要はありません。	**show interface counters protocol status**

次の表に、no_perf_test_init.tcl サンプル ポリシーの実行前に設定する必要がある EEM 環境変数を示します。

表 *10 : no_perf_test_init.tcl* ポリシーで使用される環境変数

環境変数	説明	例
_perf_iterations	測定を反復する回数。	**100**
_perf_cmd1	測定テストの一部として実行される最初の非インタラクティブ CLI コマンド。この変数は任意で、指定する必要はありません。	**enable**
_perf_cmd2	測定テストの一部として実行される 2 番目の非インタラクティブ CLI コマンド。_perf_cmd2 を使用するには、_perf_cmd1 を定義する必要があります。この変数は任意で、指定する必要はありません。	**show version**
_perf_cmd3	測定テストの一部として実行される 3 番目の非インタラクティブ CLI コマンド。_perf_cmd3 を使用するには、_perf_cmd1 を定義する必要があります。この変数は任意で、指定する必要はありません。	**show interface counters protocol status**

次の表に、sl_intf_down.tcl サンプル ポリシーの実行前に設定する必要がある EEM 環境変数を示します。

表 *11 : sl_intf_down.tcl* ポリシーで使用される環境変数

環境変数	説明	例
_config_cmd1	実行される 1 番めのコンフィギュレーション コマンド。	**interface Ethernet1/0**

環境変数	説明	例
_config_cmd2	実行される 2 番めのコンフィギュレーション コマンド。この変数は任意で、指定する必要はありません。	**no shutdown**
_syslog_pattern	ポリシー実行時を決定するために syslog メッセージを比較するために使用する正規表現パターンマッチ文字列。	.*UPDOWN.*FastEthernet0/0.*

次の表に、tm_cli_cmd.tcl サンプル ポリシーの実行前に設定する必要がある EEM 環境変数を示します。

表 *12 : tm_cli_cmd.tcl* ポリシーで使用される環境変数

環境変数	説明	例
_cron_entry	ポリシーが実行されるときを決定する CRON 仕様。	0-59/1 0-23/1 * * 0-7
_show_cmd	ポリシーの実行時に実行される CLI コマンド。	**show version**

次の表に、tm_crash_reporter.tcl サンプル ポリシーの実行前に設定する必要がある EEM 環境変数を示します。

表 *13 : tm_crash_reporter.tcl* ポリシーで使用される環境変数

環境変数	説明	例
_crash_reporter_debug	tm_crash_reporter.tcl のデバッグ情報がイネーブルであるかどうかを決定する値。この変数は任意で、指定する必要はありません。	1
_crash_reporter_url	クラッシュ レポートが送信される URL 位置。	http://www.example.com/fm/interface_tm.cgi

次の表に、tm_fsys_usage.tcl サンプル ポリシーの実行前に設定する必要がある EEM 環境変数を示します。

表 *14 : tm_fsys_usage.tcl* ポリシーで使用される環境変数

環境変数	説明	例
_tm_fsys_usage_cron	**event_register** Tcl コマンド拡張で使用される CRON 仕様。指定されない場合、tm_fsys_usage.tcl ポリシーが 1 分に 1 回トリガーされます。この変数は任意で、指定する必要はありません。	0-59/1 0-23/1 * * 0-7

環境変数	説明	例
_tm_fsys_usage_debug	この変数が値 1 に設定された場合、システムのすべてのエントリのディスク使用率情報が表示されます。この変数は任意で、指定する必要はありません。	1
_tm_fsys_usage_freebytes	システムまたは特定のプレフィックスの空きバイト数しきい値。空きスペースが所定の値を下回ると、警告が表示されます。この変数は任意で、指定する必要はありません。	disk2:98000000
_tm_fsys_usage_percent	システムまたは特定のプレフィックスのディスク使用割合しきい値。ディスク使用割合が所定の割合を超えると、警告が表示されます。指定されない場合、すべてのシステムのデフォルトのディスク使用割合は、80%です。この変数は任意で、指定する必要はありません。	nvram:25 disk2:5

一部の EEM ポリシーの登録

ポリシーの登録後に EEM 環境変数が変更された場合、一部の EEM ポリシーは、登録を解除し、再登録する必要があります。ポリシーの開始時に表示される event_register_*xxx* ステートメントには、一部の EEM 環境変数が含まれ、このステートメントは、ポリシーが実行される条件の確立に使用されます。ポリシーの登録後に環境変数が変更された場合、条件は無効になります。いかなるエラーも回避するには、ポリシーの登録を解除し、再登録する必要があります。次の変数に影響が及ぼされます。

- _cron_entry in the tm_cli_cmd.tcl policy
- _syslog_pattern in the sl_intf_down.tcl policy

すべてのサンプル ポリシーの基本設定の詳細

Embedded Event Manager から電子メールを送信できるようにするには、**hostname** コマンドと **ip domain-name** コマンドを設定する必要があります。EEM 環境変数も設定する必要があります。Cisco IOS イメージのブート後、次の初期設定を使用し、ネットワークで適切な値を置き換えます。tm_fsys_usage サンプル ポリシーの環境変数（上の表を参照）はすべて任意で、ここではそのリストは示されていません。

```
hostname cpu
ip domain-name example.com
event manager environment _email_server ms.example.net
event manager environment _email_to username@example.net
event manager environment _email_from engineer@example.net
event manager environment _email_cc projectgroup@example.net
event manager environment _cron_entry 0-59/2 0-23/1 * * 0-7
event manager environment _show_cmd show event manager policy registered
event manager environment _syslog_pattern .*UPDOWN.*FastEthernet0/0
event manager environment _config_cmd1 interface Ethernet1/0
event manager environment _config_cmd2 no shutdown
event manager environment _crash_reporter_debug 1
event manager environment _crash_reporter_url
```

```
http://www.example.com/fm/interface_tm.cgi
end
```

サンプル ポリシーの使用

ここでは、次の設定シナリオを使用して、Tcl サンプル ポリシーを使用する方法について説明します。

Mandatory.go_*.tcl サンプル ポリシーの実行

GOLD EEM ポリシーの一部として実行される各テストに GOLD TCL スクリプトがあります。この TCL スクリプトをテスト用に変更したり、連続障害回数を指定することができ、また、デフォルトの是正アクションを変更することもできます。たとえば、他の CLI ベースのアクションをリセットするのではなく、ラインカードの電源を切ることができます。

登録済みのテストごとにデフォルトの TCL スクリプトを使用できます。このスクリプトはシステムに登録し、デフォルトのアクションと一致させることができます。これは、これらのスクリプトによってオーバーライドできます。

次の表は、GOLD が EEM にインストールした必須ポリシーのリストです。各ポリシーが、カードのリセットやポートの無効化といった何らかのアクションを実行します。

GOLD Tcl スクリプト	テスト
Mandatory.go_asicsync.tcl	TestAsicSync
Mandatory.go_bootup.tcl	すべてのブートアップテストに共通。
Mandatory.go_fabric.tcl	TestFabricHealth
Mandatory.go_fabrich0.tcl	TestFabricCh0Health
Mandatory.go_fabrich1.tcl	TestFabricCh1Health
Mandatory.go_ipsec.tcl	TestIPSecEncrypDecrypPkt
Mandatory.go_mac.tcl	TestMacNotification
Mandatory.go_nondislp.tcl	TestNonDisruptiveLoopback
Mandatory.go_scratchreg.tcl	TestScratchRegister
Mandatory.go_sprping.tcl	TestSPRPInbandPing

次に、このポリシーの使用方法を示すサンプル設定について説明します。ユーザ EXEC モードを開始し、デバイス プロンプトで **enable** コマンドを入力します。デバイスは特権 EXEC モードを開始します。このモードで **show event manager policy registered** コマンドを入力すると、現在登録されているポリシーがないことを確認できます。次のコマンドはどのポリシーがインストールできるかを表示する **show event manager policy available** コマンドです。**configure terminal** コマンドを入力してグローバル コンフィギュレーション モードが開始されたら、**event manager policy** コマンドを使用して mandatory.go_*.tcl ポリシーを EEM に登録できます。グロー

バル コンフィギュレーション モードを終了し、もう一度 **show event manager policy registered** コマンドを入力してポリシーが登録されていることを確認します。

```
enable
show event manager policy registered
show event manager policy available
configure terminal
 event manager policy Mandatory.go_spuriousisr.tcl
 end
show event manager policy registered
show event manager environment
```

ap_perf_test_base_cpu.tcl および no_perf_test_init.tcl サンプル ポリシーの実行

これらのサンプル ポリシーは、EEM ポリシーの CPU パフォーマンスを測定します。これらのポリシーは、各 EEM ポリシーの標準実行時間の検出に役立ち、CLI ライブラリ コマンドを使用して EEM 環境変数の perf_cmd1（任意で _perf_cmd2 および _perf_cmd3）で指定されているコンフィギュレーション コマンドを実行します。

次に、このポリシーの使用方法を示すサンプル設定について説明します。ユーザ EXEC モードを開始し、デバイス プロンプトで **enable** コマンドを入力します。デバイスは特権 EXEC モードを開始します。このモードで **show event manager policy registered** コマンドを入力すると、現在登録されているポリシーがないことを確認できます。次のコマンドはどのポリシーがインストールできるかを表示する **show event manager policy available** コマンドです。**configure terminal** コマンドを入力してグローバル コンフィギュレーション モードが開始された後に **service timestamps debug datetime msec** コマンドを入力すると、**event manager policy** コマンドを使用して EEM に ap_perf_test_base_cpu.tcl ポリシーと no_perf_test_init.tcl ポリシーを登録できます。グローバル コンフィギュレーション モードを終了し、もう一度 **show event manager policy registered** コマンドを入力してポリシーが登録されていることを確認します。

ポリシー ap_perf_test_base_cpu.tcl および no_perf_test_init.tcl はセットで実行されるので、一緒に登録する必要があります。no_perf_test_init.tcl ポリシーを実行し、テストを開始することができます。反復ごとに返ってくる syslog メッセージを使用して結果を分析します。反復の総回数は、変数 _perf_iterations で指定します。時間の差を測り、反復の総回数で除算して、各 EEM ポリシーの平均実行時間を計算します。

```
enable
show event manager policy registered
show event manager policy available
show event manager environment
configure terminal
 service timestamps debug datetime msec
 event manager environment _perf_iterations 100
 event manager policy ap_perf_test_base_cpu.tcl
 event manager policy no_perf_test_init.tcl
 end
show event manager policy registered
show event manager policy available
show event manager environment
event manager run no_perf_test_init.tcl
```

no_perf_test_init.tcl サンプル ポリシーの実行

このサンプルポリシーでは、EEMポリシーのCPIパフォーマンスを測定します。このポリシーは、各 EEM ポリシーの標準実行時間の検出に役立ち、CLI ライブラリ コマンドを使用して EEM 環境変数の perf_cmd1（任意で _perf_cmd2 および _perf_cmd3）で指定されているコンフィギュレーション コマンドを実行します。

次に、このポリシーの使用方法を示すサンプル設定について説明します。ユーザEXECモードを開始し、デバイス プロンプトで **enable** コマンドを入力します。デバイスは特権 EXEC モードを開始します。このモードで **show event manager policy registered** コマンドを入力すると、現在登録されているポリシーがないことを確認できます。次のコマンドはどのポリシーがインストールできるかを表示する **show event manager policy available** コマンドです。**configure terminal** コマンドを入力してグローバル コンフィギュレーション モードが開始されたら、**event manager policy** コマンドを使用して no_perf_test_init.tcl ポリシーを EEM に登録できます。グローバル コンフィギュレーション モードを終了し、もう一度 **show event manager policy registered** コマンドを入力してポリシーが登録されていることを確認します。

反復ごとに返ってくるsyslogメッセージを使用して結果を分析します。反復の総回数は、変数 _perf_iterations で指定します。時間の差を測り、反復の総回数で除算して、各EEMポリシーの平均実行時間を計算します。

```
enable
show event manager policy registered
show event manager policy available
configure terminal
 event manager policy no_perf_test_init.tcl
 end
show event manager policy registered
show event manager environment
```

sl_intf_down.tcl サンプル ポリシーの実行

このサンプル ポリシーでは、特定のパターンで Syslog メッセージが記録されるときに設定を変更する機能について説明します。ポリシーでは、イベントについての詳細情報が収集され、CLI ライブラリを使用して、EEM 環境変数 _config_cmd1 と、任意で _config_cmd2 で指定された、コンフィギュレーション コマンドが実行されます。CLI コマンドの結果とともに、電子メール メッセージが送信されます。

次に、このポリシーの使用方法を示すサンプル設定について説明します。ユーザEXECモードを開始し、デバイス プロンプトで **enable** コマンドを入力します。デバイスは特権 EXEC モードを開始します。このモードで **show event manager policy registered** コマンドを入力すると、現在登録されているポリシーがないことを確認できます。次のコマンドはどのポリシーがインストールできるかを表示する **show event manager policy available** コマンドです。**configure terminal** コマンドを入力してグローバル コンフィギュレーション モードが開始されたら、**event manager policy** コマンドを使用して sl_intf_down.tcl ポリシーを EEM に登録できます。グローバル コンフィギュレーション モードを終了し、もう一度 **show event manager policy registered** コマンドを入力してポリシーが登録されていることを確認します。

インターフェイスがダウンするときに、ポリシーが実行されます。**show event manager environment** コマンドを入力して現在の環境変数の値を表示します。_syslog_pattern EEM 環境

変数で指定されたインターフェイスのケーブルを取り外します（またはシャットダウンを設定します）。インターフェイスがダウンし、インターフェイスがダウンしていることについてのSyslogメッセージを記録するSyslogデーモンのプロンプトが表示されて、Syslogイベントディテクタが呼び出されます。

Syslog イベント ディテクタによって、未解決のイベント仕様が見直され、インターフェイスステータス変更に対する一致が検索されます。EEM サーバに通知され、サーバでは、このイベント sl_intf_down.tcl を処理するために登録されたポリシーが実行されます。

```
enable
show event manager policy registered
show event manager policy available
configure terminal
 event manager policy sl_intf_down.tcl
 end
show event manager policy registered
show event manager environment
```

tm_cli_cmd.tcl サンプル ポリシーの実行

このサンプル ポリシーでは、定期的に CLI コマンドを実行し、結果を E メールで送信する機能について説明します。CRON 仕様「0-59/2 0-23/1 * * 0-7」を使用すると、このポリシーは、毎時 2 分目に実行されます。ポリシーでは、イベントについての詳細情報が収集され、CLI ライブラリを使用して、EEM 環境変数 _show_cmd で指定された、コンフィギュレーション コマンドが実行されます。CLI コマンドの結果とともに、電子メール メッセージが送信されます。

次に、このポリシーの使用方法を示すサンプル設定について説明します。ユーザ EXEC モードを開始し、デバイス プロンプトで **enable** コマンドを入力します。デバイスは特権 EXEC モードを開始します。このモードで **show event manager policy registered** コマンドを入力すると、現在登録されているポリシーがないことを確認できます。次のコマンドはどのポリシーがインストールできるかを表示する **show event manager policy available** コマンドです。**configure terminal** コマンドを入力してグローバル コンフィギュレーション モードが開始されたら、**event manager policy** コマンドを使用して tm_cli_cmd.tcl ポリシーを EEM に登録できます。グローバル コンフィギュレーション モードを終了し、**show event manager policy registered** コマンドを入力してポリシーが登録されていることを確認します。

EEM 環境変数 _cron_entry に設定されている CRON 文字列に従って、タイマー イベント ディテクタによって、定期的にこのケースのイベントがトリガーされます。EEM サーバに通知され、サーバでは、このイベント tm_cli_cmd.tcl を処理するために登録されたポリシーが実行されます。

```
enable
show event manager policy registered
show event manager policy available
configure terminal
 event manager policy tm_cli_cmd.tcl
 end
show event manager policy registered
```

tm_crash_reporter.tcl サンプル ポリシーの実行

このサンプル ポリシーでは、ある URL へ HTTP 形式のクラッシュ レポートを送信する機能について説明します。ポリシー登録がスタートアップ コンフィギュレーション ファイルに保存されている場合、ポリシーは、ブートの 5 秒後にトリガーされます。トリガーされると、スクリプトによって、リロード原因の検索が試行されます。リロードの原因がクラッシュの場合、ポリシーによって、関連する crashinfo ファイルが検索され、環境変数 _crash_reporter_url でユーザによって指定された URL へ、この情報が送信されます。CGI スクリプト interface_tm.cgi は、tm_crash_reporter.tcl ポリシーから URL を受け取るために作成され、ターゲット URL マシン上のローカル データベースにクラッシュ情報が保存されます。

Perl CGI スクリプト interface_tm.cgi が作成され、HTTP サーバが含まれているマシン上で実行するために設計され、tm_crash_reporter.tcl ポリシーが実行されているデバイスからアクセスできます。interface_tm.cgi スクリプトによって、tm_crash_reporter.tcl から渡されたデータが解析され、テキスト ファイルの末尾にクラッシュ情報が追加され、これによって、システムのすべてのクラッシュの履歴が作成されます。さらに、各クラッシュの詳細情報は、ユーザが指定したクラッシュ データベース ディレクトリの 3 つのファイルに保存されます。別の Perl CGI スクリプト crash_report_display.cgi は、interface_tm.cgi スクリプトによって作成されたデータベースに保存されている情報を表示するために作成されました。crash_report_display.cgi スクリプトは、interface_tm.cgi が含まれているマシンと同じマシンに置く必要があります。そのマシンでは、Internet Explorer または Netscape などのブラウザが実行されている必要があります。crash_report_display.cgi スクリプトが実行されると、読み取り可能な形式でクラッシュ情報が表示されます。

次に、このポリシーの使用方法を示すサンプル設定について説明します。ユーザ EXEC モードを開始し、デバイス プロンプトで **enable** コマンドを入力します。デバイスは特権 EXEC モードを開始します。このモードで **show event manager policy registered** コマンドを入力すると、現在登録されているポリシーがないことを確認できます。次のコマンドはどのポリシーがインストールできるかを表示する **show event manager policy available** コマンドです。**configure terminal** コマンドを入力してグローバル コンフィギュレーション モードが開始されたら、**event manager policy** コマンドを使用して tm_crash_reporter.tcl ポリシーを EEM に登録できます。グローバル コンフィギュレーション モードを終了し、**show event manager policy registered** コマンドを入力してポリシーが登録されていることを確認します。

```
enable
show event manager policy registered
show event manager policy available
configure terminal
 event manager policy tm_crash_reporter.tcl
 end
show event manager policy registered
```

tm_fsys_usage.tcl サンプル ポリシーの実行

このサンプル ポリシーでは、ディスク領域の使用状況を定期的にモニタし、値が設定可能なしきい値に近くなったときに Syslog を介してレポートする機能について説明します。

次に、このポリシーの使用方法を示すサンプル設定について説明します。ユーザ EXEC モードを開始し、デバイス プロンプトで **enable** コマンドを入力します。デバイスは特権 EXEC モードを開始します。このモードで **show event manager policy registered** コマンドを入力すると、

現在登録されているポリシーがないことを確認できます。次のコマンドはどのポリシーがインストールできるかを表示する **show event manager policy available** コマンドです。**configure terminal** コマンドを入力してグローバル コンフィギュレーション モードが開始されたら、**event manager policy** コマンドを使用して tm_fsys_usage.tcl ポリシーを EEM に登録できます。グローバル コンフィギュレーション モードを終了し、もう一度 **show event manager policy registered** コマンドを入力してポリシーが登録されていることを確認します。tm_fsys_usage.tcl ポリシーで使用されるオプション環境変数のいずれかを設定した場合、**show event manager environment** コマンドによって、設定された変数が表示されます。

```
enable
show event manager policy registered
show event manager policy available
configure terminal
 event manager policy tm_fsys_usage.tcl
 end
show event manager policy registered
show event manager environment
```

Tcl のサンプル スクリプトを使用したポリシーのプログラミングの例

ここでは、EEM システム ポリシーとして含まれているいくつかのサンプル ポリシーについて説明します。これらのポリシーの詳細については、EEM イベント ディテクタのデモの例 （40 ページ）を参照してください。

Mandatory.go_ipsec.tcl サンプル ポリシー

次のサンプルポリシーは、TestIPSecEncrypDecrypPkt テスト用です。

```
::cisco::eem::event_register_gold card all testing_type monitoring test_name Tes
tIPSecEncrypDecrypPkt consecutive_failure 6 platform_action 0 queue_priority las
t
#
#  GOLD TestIPSecEncrypDecrypPkt Test TCL script
#
#  March 2005, Hai Qiu
#
#  Copyright (c) 2005-2007 by cisco Systems, Inc.
#  All rights reserved.
#
#
# Register for TestIPSecEncrypDecrypPkt test even
# the elements for register the event
# card [all | card #]
# sub_card [all | sub_card #]
# severity_major | severity_minor | severity_normal  default :  severity_normal
# new_failure [true | false]  default: dont_care
# testing_type [bootup | ondemand | schedule | monitoring]
# test_name [ test name ]
# test_id [ test # ]
# consecutive_failure [ consecutive_failure # ]
# platform_action [action_flag]
# action_flag [ 0 | 1 | 2 ]
# queue_priority [ normal | low | high | last]   default: normal
#
# Note:
#    1: "card" element is required. If other elements are not specified,
```

```
#       treat them as dont care, or default.
#
#    2: action_flag is platform specific. It is up to platform to
#       determine what action need to be taken based on the value
#       For Cat6k platform
#       action_flag 0 : TCL script take action to reset card
#       action_flag 1 : TCL script doesn't take action to reset card
#       action_flag 2 : TCL script takes action to reset card for bootup diag
#                       when there is major error
#       action_flag 3 : TCL script doesn't take action to reset card for
#                       bootup diag when there is major error
#
#    3: "queue_priority last" would guarantee this policy will be executed last
#       if there are other EEM events in queue with queue priority other
#       than "last"
namespace import ::cisco::eem::*
namespace import ::cisco::lib::*
# 1. query the information of latest triggered eem event
array set arr_einfo [event_reqinfo]
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
puts "GOLD EEM TCL policy for TestIPSecEncrypDecrypPkt"
#set msg [format "array=%s", array names arr_einfo]
#puts "msg $msg"
#set msg $arr_einfo(msg)
set card $arr_einfo(card)
set sub_card $arr_einfo(sub_card)
#set overall_result $arr_einfo(overall_result)
#puts "GOLD event msg recieved: $card/$sub_card overall_result= $overall_result"
# 2. execute the user-defined config commands
if [catch {cli_open} result] {
    error $result $errorInfo
} else {
    array set cli1 $result
}
if [catch {cli_exec $cli1(fd) "en"} result] {
    error $result $errorInfo
}
# Use "diagn action mod mod# test testname default" command
# for default platform action
if [catch {cli_exec $cli1(fd) "diagnostic action mod $card test TestIPSecEncrypD
ecrypPkt default"} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
    error $result $errorInfo
}
```

ap_perf_test_base_cpu.tcl サンプル ポリシー

次のサンプル ポリシーは、EEM ポリシーの CPU パフォーマンスを測定します。

```
::cisco::eem::event_register_appl sub_system 798 type 9999
#----------------------------------
# EEM policy used for measuring the cpu performance of EEM policies.
#
# July 2005, Cisco EEM team
#
```

```
# Copyright (c) 2005, 2006 by cisco Systems, Inc.
# All rights reserved.
#------------------
###
### Input arguments:
###
### arg1 $iter                    - current iteration count
###
### The following EEM environment variables are used:
###
### _perf_iterations (mandatory) - number of iterations over which we
###                                will run our measurement.
### Example:
### event manager environment _perf_iterations 100
###
### _perf_cmd1 (optional)        - optional non interactive cli command
###                                to be executed as part of the
###                                measurement test.
### Example:
### event manager environment _perf_cmd1 enable
###
### _perf_cmd2 (optional)        - optional non interactive cli command
###                to be executed as part of the
###                                measurement test.
###                                To use _perf_cmd2, _perf_cmd1 MUST
###                be defined.
### Example:
### event manager environment _perf_cmd2 show ver
###
### _perf_cmd3 (optional)        - optional non interactive cli command
###                                to be executed as part of the
###                                measurement test.
###                                To use _perf_cmd3, _perf_cmd1 MUST
###                                be defined.
### Example:
### event manager environment _perf_cmd3 show int counters protocol status
###
### Description:
###   Iterate through _perf_iterations of this policy.
###   It is up to the user to calculate the average
###   execution time based on the system timestamps.
###   Optional commands _perf_cmd1,
###   _perf_cmd2 and _perf_cmd3 are executed if defined.
###
###   A value of 100 is a good starting point.
###
### Outputs:
###   Console output.
###
### Usage example:
###   >conf t
###   >service timestamps debug datetime msec
###   >event manager environment _perf_iterations 100
###   >event manager policy ap_perf_base_cpu.tcl
###   >event manager policy no_perf_test_init.tcl
###   >end
###   2d19h: %SYS-5-CONFIG_I: Configured from console by console
###   >event manager run no_perf_test_init.tcl
###
###   Oct 16 14:57:17.284: %SYS-5-CONFIG_I: Configured from console by console
###   >event manager run no_perf_test_init.tcl
###
###   Oct 16 19:32:02.772: %HA_EM-6-LOG:
###     eem_policy/no_perf_test_init.tcl: EEM performance test start
```

```
###   Oct 16 19:32:03.115: %HA_EM-6-LOG:
###     eem_policy/ap_perf_test_base_cpu.tcl: EEM performance test iteration 1
###   Oct 16 19:32:03.467: %HA_EM-6-LOG:
###     eem_policy/ap_perf_test_base_cpu.tcl: EEM performance test iteration 2
###   ...
###   Oct 16 19:32:36.936: %HA_EM-6-LOG:
###     eem_policy/ap_perf_test_base_cpu.tcl: EEM performance test iteration 100
###   Oct 16 19:32:36.936: %HA_EM-6-LOG:
###     eem_policy/ap_perf_test_base_cpu.tcl: EEM performance test end
###
###   The user must calculate execution time and average time of execution.
###   In this example, total time = 19:32:36.936 - 19:32:02.772 = 34.164
###   Average script execution time = 341.64 milliseconds
###
# check if all the env variables we need exist
# If any of them doesn't exist, print out an error msg and quit
if {![info exists _perf_iterations]} {
    set result \
        "Policy cannot be run: variable _perf_iterations has not been set"
    error $result $errorInfo
}
# ensure our target iteration count > 0
if {$_perf_iterations <= 0} {
    set result \
        "Policy cannot be run: variable _perf_iterations <= 0"
    error $result $errorInfo
}
namespace import ::cisco::eem::*
namespace import ::cisco::lib::*
# query the event info
array set arr_einfo [event_reqinfo]
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
set iter $arr_einfo(data1)
set iter [expr $iter + 1]
# if _perf_cmd1 is defined
if {[info exists _perf_cmd1]} {
    # open the cli library
    if [catch {cli_open} result] {
        error $result $errorInfo
    } else {
        array set cli1 $result
    }
    # execute the comamnd defined in _perf_cmd1
    if [catch {cli_exec $cli1(fd) $_perf_cmd1} result] {
        error $result $errorInfo
    }
    # if _perf_cmd2 is defined
    if {[info exists _perf_cmd2]} {
        # execute the comamnd defined in _perf_cmd2
        if [catch {cli_exec $cli1(fd) $_perf_cmd2} result] {
            error $result $errorInfo
        } else {
            set cmd_output $result
        }
    }
    # if _perf_cmd3 is defined
    if {[info exists _perf_cmd3]} {
        # execute the comamnd defined in _perf_cmd3
        if [catch {cli_exec $cli1(fd) $_perf_cmd3} result] {
            error $result $errorInfo
```

```
        } else {
            set cmd_output $result
        }
    }
    # close the cli library
    if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
        error $result $errorInfo
    }
}

# log a message
set msg [format "EEM performance test iteration %s" $iter]
action_syslog priority info msg $msg
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
# use the context info from the previous run to determine when to end
if {$iter >= $_perf_iterations} {
    #log the final messages
    action_syslog priority info msg "EEM performance test end"
    if {$_cerrno != 0} {
        set result [format \
                "component=%s; subsys err=%s; posix err=%s;\n%s" \
                $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
        error $result
    }
    exit 0
}
# cause the next iteration to run
event_publish sub_system 798 type 9999 arg1 $iter
if {$_cerrno != 0} {
    set result [format \
            "component=%s; subsys err=%s; posix err=%s;\n%s" \
            $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
```

tm_cli_cmd.tcl サンプル ポリシー

次に、設定可能な CRON エントリが実行されるサンプル ポリシーについて説明します。ポリシーでは、設定可能なCisco IOS CLI コマンドが実行され、結果が電子メールで送信されます。タイムスタンプとともに出力が末尾に追加される任意のログ ファイルを定義することができます。

```
::cisco::eem::event_register_timer cron name crontimer2 cron_entry $
_cron_entry maxrun 240
#-------------------------------------------------------------------
# EEM policy that will periodically execute a cli command and email the
# results to a user.
#
# July 2005, Cisco EEM team
#
# Copyright (c) 2005 by cisco Systems, Inc.
# All rights reserved.
#-------------------------------------------------------------------
### The following EEM environment variables are used:
###
### _cron_entry (mandatory)            - A CRON specification that determines
###                                      when the policy will run. See the
```

```
###                                     IOS Embedded Event Manager
###                                     documentation for more information
###                                     on how to specify a cron entry.
### Example: _cron_entry                0-59/1 0-23/1 * * 0-7
###
### _log_file (mandatory without _email_....)
###                                   - A filename to append the output to.
###                                     If this variable is defined, the
###                                     output is appended to the specified
###                                     file with a timestamp added.
### Example: _log_file                  bootflash:/my_file.log
###
### _email_server (mandatory without _log_file)
###                                   - A Simple Mail Transfer Protocol (SMTP)
###                                     mail server used to send e-mail.
### Example: _email_server              mailserver.example.com
###
### _email_from (mandatory without _log_file)
###                                   - The address from which e-mail is sent.
### Example: _email_from                devtest@example.com
###
### _email_to (mandatory without _log_file)
###                                   - The address to which e-mail is sent.
### Example: _email_to                  engineering@example.com
###
### _email_cc (optional)              - The address to which the e-mail must
###                                     be copied.
### Example: _email_cc                  manager@example.com
###
### _show_cmd (mandatory)             - The CLI command to be executed when
###                                     the policy is run.
### Example: _show_cmd                  show version
###
# check if all required environment variables exist
# If any required environment variable does not exist, print out an error msg and quit
if {![info exists _log_file]} {
    if {![info exists _email_server]} {
 set result \
  "Policy cannot be run: variable _log_file or _email_server has not been set"
 error $result $errorInfo
    }
    if {![info exists _email_from]} {
 set result \
  "Policy cannot be run: variable _log_file or _email_from has not been set"
 error $result $errorInfo
    }
    if {![info exists _email_to]} {
 set result \
  "Policy cannot be run: variable _log_file ore _email_to has not been set"
 error $result $errorInfo
    }
    if {![info exists _email_cc]} {
 #_email_cc is an option, must set to empty string if not set.
 set _email_cc ""
    }
}
if {![info exists _show_cmd]} {
    set result \
        "Policy cannot be run: variable _show_cmd has not been set"
    error $result $errorInfo
}
namespace import ::cisco::eem::*
namespace import ::cisco::lib::*
# query the event info and log a message
```

```
array set arr_einfo [event_reqinfo]
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
global timer_type timer_time_sec
set timer_type $arr_einfo(timer_type)
set timer_time_sec $arr_einfo(timer_time_sec)
# log a message
set msg [format "timer event: timer type %s, time expired %s" \
        $timer_type [clock format $timer_time_sec]]
action_syslog priority info msg $msg
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
 $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
# 1. execute the command
if [catch {cli_open} result] {
    error $result $errorInfo
} else {
    array set cli1 $result
}
if [catch {cli_exec $cli1(fd) "en"} result] {
    error $result $errorInfo
}
# save exact execution time for command
set time_now [clock seconds]
# execute command
if [catch {cli_exec $cli1(fd) $_show_cmd} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
    # format output: remove trailing router prompt
    regexp {\n*(.*\n)([^\n]*)$} $result dummy cmd_output
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
    error $result $errorInfo
}

# 2. log the success of the CLI command
set msg [format "Command \"%s\" executed successfully" $_show_cmd]
action_syslog priority info msg $msg
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
# 3. if _log_file is defined, then attach it to the file
if {[info exists _log_file]} {
    # attach output to file
    if [catch {open $_log_file a+} result] {
        error $result
    }
    set fileD $result
    # save timestamp of command execution
    #      (Format = 00:53:44 PDT Mon May 02 2005)
    set time_now [clock format $time_now -format "%T %Z %a %b %d %Y"]
    puts $fileD "%%% Timestamp = $time_now"
    puts $fileD $cmd_output
    close $fileD
}
# 4. if _email_server is defined send the email out
```

```
if {[info exists _email_server]} {
    set routername [info hostname]
    if {[string match "" $routername]} {
 error "Host name is not configured"
    }
    if [catch {smtp_subst [file join $tcl_library email_template_cmd.tm]} \
     result] {
 error $result $errorInfo
    }
    if [catch {smtp_send_email $result} result] {
 error $result $errorInfo
    }
}
```

sl_intf_down.tcl サンプル ポリシー

次に、設定可能な Syslog メッセージが記録されるときに実行されるサンプル ポリシーを示します。ポリシーでは、設定可能な CLI コマンドが実行され、結果が電子メールで送信されます。

```
::cisco::eem::event_register_syslog occurs 1 pattern $_syslog_pattern maxrun 90

#------------------------------------------------------------------
# EEM policy to monitor for a specified syslog message.
# Designed to be used for syslog interface-down messages.
# When event is triggered, the given config commands will be run.
#
# July 2005, Cisco EEM team
#
# Copyright (c) 2005 by cisco Systems, Inc.
# All rights reserved.
#------------------------------------------------------------------

### The following EEM environment variables are used:
###
### _syslog_pattern (mandatory)        - A regular expression pattern match string
###                                      that is used to compare syslog messages
###                                      to determine when policy runs
### Example: _syslog_pattern             .*UPDOWN.*FastEthernet0/0.*
###
### _email_server (mandatory)          - A Simple Mail Transfer Protocol (SMTP)
###                                      mail server used to send e-mail.
### Example: _email_server               mailserver.example.com
###
### _email_from (mandatory)            - The address from which e-mail is sent.
### Example: _email_from                 devtest@example.com
###
### _email_to (mandatory)              - The address to which e-mail is sent.
### Example: _email_to                   engineering@example.com
###
### _email_cc (optional)               - The address to which the e-mail must
###                                      be copied.
### Example: _email_cc                   manager@example.com
###
### _config_cmd1 (optional)            - The first configuration command that
###                                      is executed.
### Example: _config_cmd1                interface Ethernet1/0
###
### _config_cmd2 (optional)            - The second configuration command that
###                                      is executed.
### Example: _config_cmd2                no shutdown
###
```

```
# check if all the env variables we need exist
# If any of them doesn't exist, print out an error msg and quit
if {![info exists _email_server]} {
    set result \
        "Policy cannot be run: variable _email_server has not been set"
    error $result $errorInfo
}
if {![info exists _email_from]} {
    set result \
        "Policy cannot be run: variable _email_from has not been set"
    error $result $errorInfo
}
if {![info exists _email_to]} {
    set result \
        "Policy cannot be run: variable _email_to has not been set"
    error $result $errorInfo
}
if {![info exists _email_cc]} {
     #_email_cc is an option, must set to empty string if not set.
     set _email_cc ""
}

namespace import ::cisco::eem::*
namespace import ::cisco::lib::*

# 1. query the information of latest triggered eem event
array set arr_einfo [event_reqinfo]

if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
      $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}

set msg $arr_einfo(msg)
set config_cmds ""

# 2. execute the user-defined config commands
if [catch {cli_open} result] {
    error $result $errorInfo
} else {
    array set cli1 $result
}
if [catch {cli_exec $cli1(fd) "en"} result] {
    error $result $errorInfo
}
if [catch {cli_exec $cli1(fd) "config t"} result] {
    error $result $errorInfo
}

if {[info exists _config_cmd1]} {
    if [catch {cli_exec $cli1(fd) $_config_cmd1} result] {
        error $result $errorInfo
    }
    append config_cmds $_config_cmd1
}

if {[info exists _config_cmd2]} {
    if [catch {cli_exec $cli1(fd) $_config_cmd2} result] {
        error $result $errorInfo
    }
    append config_cmds "\n"
    append config_cmds $_config_cmd2
```

```
}

if [catch {cli_exec $cli1(fd) "end"} result] {
    error $result $errorInfo
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
    error $result $errorInfo
}

after 60000
# 3. send the notification email
set routername [info hostname]
if {[string match "" $routername]} {
    error "Host name is not configured"
}

if [catch {smtp_subst [file join $tcl_library email_template_cfg.tm]} result] {
    error $result $errorInfo
}
if [catch {smtp_send_email $result} result] {
    error $result $errorInfo
}
```

次に、前述の EEM サンプル ポリシーで使用される電子メール テンプレート ファイルの使用例を示します。

```
email_template_cfg.tm
Mailservername: $_email_server
From: $_email_from
To: $_email_to
Cc: $_email_cc
Subject: From router $routername: Periodic $_show_cmd Output
$cmd_output
```

Embedded Event Manager ポリシーのデバッグの例

次に、CLI ライブラリおよび SMTP ライブラリのデバッグ例を示します。

CLI ライブラリのデバッグ

CLI ライブラリを使用すると、ユーザは、CLI コマンドを実行し、Tcl のコマンドの出力を取得できます。Embedded Event Manager の **debug** コマンドは、このライブラリのユーザ向けに用意されています。CLI ライブラリのデバッグを有効にするコマンドは、**debug event manager tcl cli_library** です。イネーブルの場合、このコマンドによって、CLI のやり取りを処理する TTY セッションに渡され、TTY セッションから読み戻される、すべてのデータが表示されます。このデータを使用すると、ユーザが CLI に渡すコマンドが有効になります。

デバッグ イベント マネージャ tcl cli_library コマンドの例

この例では、サンプル ポリシー sl_intf_down.tcl が使用されます。トリガーされると、sl_intf_down.tcl によって、CLI ライブラリを介して CLI にコンフィギュレーション コマンドが渡されます。次で渡されるコマンドは、**show event manager environment** です。このコマンドは、コンフィギュレーション モードでは有効ではありません。**debug** コマンドが有効ではない場合、出力は次のとおりです。

```
00:00:57:sl_intf_down.tcl[0]:config_cmds are show eve man env
00:00:57:%SYS-5-CONFIG_I:Configured from console by vty0
```

前述の出力で、ユーザは、CLI でコマンドが正常終了したかどうかはわかりません。**debug event manager tcl cli_library** コマンドが有効である場合は、次が表示されます。

```
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : CTL : cli_open called.
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT : nelson>
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : IN  : nelson>enable
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT : nelson#
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : IN  : nelson#configure terminal
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT : Enter configuration commands, one
per line.  End with CNTL/Z.
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT : nelson(config)#
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : IN  : nelson(config)#show event manager
environment
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT :                  ^
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT : % Invalid input detected at '^'
marker.
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT : nelson(config)#
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : IN  : nelson(config)#end
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : OUT : nelson#
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : CTL : cli_close called.
01:17:07: sl_intf_down.tcl[0]: DEBUG(cli_lib) : IN  : nelson#exit
01:17:07: sl_intf_down.tcl[0]: config_cmds are show event manager environment
01:17:07: %SYS-5-CONFIG_I: Configured from console by vty0
```

前述の出力には、**show event manager environment** コマンドがコンフィギュレーション モードでは無効であることが示されています。IN キーワードによって、CLI ライブラリを介して TTY へすべてのデータが渡されることが指定されます。OUT キーワードによって、CLI ライブラリを介して TTY からすべてのデータが読み戻されることが指定されます。CTL キーワードによって、CLI ライブラリで使用されるヘルパー機能が指定されます。これらのヘルパー機能は、CLI への接続の設定や、接続の削除に使用されます。

SMTP ライブラリのデバッグ

SMTP ライブラリを使用すると、ユーザは、SMTP E メール サーバへ、E メール メッセージを送信できます。Embedded Event Manager の **debug** コマンドは、このライブラリのユーザ向けに用意されています。SMTP ライブラリのデバッグを有効にするコマンドは、**debug event manager tcl smtp_library** です。イネーブルの場合、このコマンドによって、SMTP ライブラリ ルーチンに渡され、SMTP ライブラリ ルーチンから読み戻される、すべてのデータが表示されます。このデータを使用すると、ユーザが SMTP ライブラリに渡すコマンドが有効になります。

デバッグ イベント マネージャ tcl smtp_library コマンドの例

この例では、サンプル ポリシー tm_cli_cmd.tcl が使用されます。トリガーされると、tm_cli_cmd.tcl は CLI ライブラリを介して **show event manager policy available system** コマンドを実行します。結果は、SMTP ライブラリを介してメールでユーザに送信されます。出力を参考に、SMTP ライブラリを使用して、関連する問題をデバッグできます。

debug event manager tcl smtp_library コマンドが有効の場合は、コンソールに次が表示されます。

```
00:39:46: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read  : 220 XXXX.example.com ESMTP
XXXX 1.1.0; Tue,
25 Jun 2002 14:20:39 -0700 (PDT)
00:39:46: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : HELO XXXX.example.com
00:39:46: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read  : 250 XXXX.example.com Hello
XXXX.example.com [XXXX],
pleased to meet you
00:39:46: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : MAIL FROM:<XX@example.com>
00:39:46: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read  : 250 <XX@example.com>...
Sender ok
00:39:46: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : RCPT TO:<XX@example.com>
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read  : 250 <XX@example.com>...
Recipient ok
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : RCPT TO:<XX@example.com>
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read  : 250 <XX@example.com>...
Recipient ok
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : DATA
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read  : 354 Enter mail, end with "."
 on a line by itself
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : Date: 25 Jun 2002 14:35:00
UTC
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : Message-ID:
<20020625143500.2387058729877@XXXX.example.com>
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : From: XX@example.com
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : To: XX@example.com
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : Cc: XX@example.com
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : Subject: From router nelson:

Periodic show eve man po ava system Output
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : No. Type    Time Created
          Name
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : 1   system  Fri May3
20:42:34 2002  pr_cdp_abort.tcl
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : 2   system  Fri May3
20:42:54 2002  pr_iprouting_abort.tcl
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : 3   system  Wed Apr3
02:16:33 2002  sl_intf_down.tcl
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : 4   system  Mon Jun24
23:34:16 2002  tm_cli_cmd.tcl
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : 5   system  Wed Mar27
05:53:15 2002  tm_crash_hist.tcl
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : nelson#
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write :
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : .
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read  : 250 ADE90179 Message accepted
 for delivery
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_write : QUIT
00:39:47: tm_cli_cmd.tcl[0]: DEBUG(smtp_lib) : smtp_read  : 221 XXXX.example.com closing
 connection
```

Tcl set コマンド操作のトレースの例

Tcl は、融通性のある言語です。Tcl の融通性の 1 つは、コマンドを上書きできることです。この例では、Tcl **set** コマンドの名前が _set に変更されます。また、テキスト「setting」が含まれるメッセージを表示し、設定しているスカラ変数を末尾に追加する、新バージョンの **set** コマンドが作成されます。この例を使用すると、設定しているスカラ変数のすべてのインスタンスをトレースできます。

```
rename set _set
```

```
proc set {var args} {
   puts [list setting $var $args]
   uplevel _set $var $args
};
```

これがポリシーに置かれると、スカラ変数が設定されるたびに、たとえば次のようなメッセージが表示されます。

```
02:17:58: sl_intf_down.tcl[0]: setting test_var 1
```

RPC イベント ディテクタのの例

```
TCL script (rpccli.tcl):
::cisco::eem::event_register_rpc
namespace import ::cisco::eem::*
namespace import ::cisco::lib::*
proc run_cli { clist } {
    set rbuf ""
    if {[llength $clist] < 1} {
    return -code ok $rbuf
    }
    if {[catch {cli_open} result]} {
        return -code error $result
    } else {
    array set cliarr $result
    }
    if {[catch {cli_exec $cliarr(fd) "enable"} result]} {
        return -code error $result
    }
    if {[catch {cli_exec $cliarr(fd) "term length 0"} result]} {
        return -code error $result
    }
    foreach cmd $clist {
    if {[catch {cli_exec $cliarr(fd) $cmd} result]} {
            return -code error $result
    }
    append rbuf $result
    }
    if {[catch {cli_close $cliarr(fd) $cliarr(tty_id)} result]} {
        puts "WARNING: $result"
    }
    return -code ok $rbuf
}
proc run_cli_interactive { clist } {
    set rbuf ""
    if {[llength $clist] < 1} {
    return -code ok $rbuf
    }
    if {[catch {cli_open} result]} {
        return -code error $result
    } else {
    array set cliarr $result
    }
    if {[catch {cli_exec $cliarr(fd) "enable"} result]} {
        return -code error $result
    }
    if {[catch {cli_exec $cliarr(fd) "term length 0"} result]} {
        return -code error $result
    }
    foreach cmd $clist {
```

```
        array set sendexp $cmd
    if {[catch {cli_write $cliarr(fd) $sendexp(send)} result]} {
            return -code error $result
    }
    foreach response $sendexp(responses) {
        array set resp $response
        if {[catch {cli_read_pattern $cliarr(fd) $resp(expect)} result]} {
                return -code error $result
        }
        if {[catch {cli_write $cliarr(fd) $resp(reply)} result]} {
                return -code error $result
        }
    }
    if {[catch {cli_read $cliarr(fd)} result]} {
            return -code error $result
    }
    append rbuf $result
    }
    if {[catch {cli_close $cliarr(fd) $cliarr(tty_id)} result]} {
        puts "WARNING: $result"
    }
    return -code ok $rbuf
}
array set arr_einfo [event_reqinfo]
set args $arr_einfo(argc)
set cmds [list]
for { set i 0 } { $i < $args } { incr i } {
    set arg "arg${i}"
    # Split each argument on the '^' character.  The first element is
    # the command, and each subsequent element is a prompt followed by
    # a response to that prompt.
    set cmdlist [split $arr_einfo($arg) "^"]
    set cmdarr(send) [lindex $cmdlist 0]
    set cmdarr(responses) [list]
    if { [expr ([llength $cmdlist] - 1) % 2] != 0 } {
    return -code 88
    }
    set cmdarr(responses) [list]
    for { set j 1 } { $j < [llength $cmdlist] } { incr j 2 } {
    set resps(expect) [lindex $cmdlist $j]
    set resps(reply) [lindex $cmdlist [expr $j + 1]]
    lappend cmdarr(responses) [array get resps]
    }
    lappend cmds [array get cmdarr]
}
set rc [catch {run_cli_interactive $cmds} output]
if { $rc != 0 } {
    error $output $errorInfo
    return -code 88
}
puts $output
```

その他の参考資料

次の項では、Tcl を使用した Embedded Event Manager ポリシー記述についての関連資料を示します。

関連資料

関連項目	マニュアル タイトル
Cisco IOS コマンド	『Cisco IOS Master Commands List, All Releases』
EEM コマンド：コマンド構文の詳細、デフォルト、コマンドモード、コマンド履歴、使用上の注意事項、および例	Cisco IOS Embedded Event Manager のコマンド リファレンス
Embedded Event Manager 概要	「Embedded Event Manager の概要」の章
CLI を使用して Embedded Event Manager ポリシーを記述する	「Writing Embedded Event Manager Policies Using the Cisco IOS CLI」の章
Embedded Resource Manager	「Embedded Resource Manager」の章

MIB

MIB	MIB のリンク
CISCO-EMBEDDED-EVENT-MGR-MIB	選択したプラットフォーム、Cisco IOS リリース、およびフィーチャ セットに関する MIB を探してダウンロードするには、次の URL にある Cisco MIB Locator を使用します。 http://www.cisco.com/go/mibs

RFC

RFC	タイトル
この機能によりサポートされた新規 RFC または改訂 RFC はありません。またこの機能による既存 RFC のサポートに変更はありません。	--

シスコのテクニカル サポート

説明	リンク
★枠で囲まれた Technical Assistance の場合★右の URL にアクセスして、シスコのテクニカル サポートを最大限に活用してください。これらのリソースは、ソフトウェアをインストールして設定したり、シスコの製品やテクノロジーに関する技術的問題を解決したりするために使用してください。この Web サイト上のツールにアクセスする際は、Cisco.com のログイン ID およびパスワードが必要です。	http://www.cisco.com/cisco/web/support/index.html