Embedded Event Manager ポリシーの設定および管理

Cisco IOS XR ソフトウェアの Embedded Event Manager（EEM）は、Cisco IOS XR ソフトウェア プロセッサのフェールオーバー サービスの任意の一部で検出されたイベントの中央クリア ハウスとして機能します。EEM は、Cisco IOS XR ソフトウェア システム内の障害イベント、ディザスタ リカバリ、およびプロセス信頼性統計情報の検出を担当します。EEM イベントは、次のような重要なイベントが発生したことを示す通知です。

- 許容値を逸脱した運用統計情報またはパフォーマンス統計情報（たとえば、空きメモリがクリティカルなしきい値未満に低下したなど）。
- 活性挿抜（OIR）
- プロセスの終了。

EEM は、ソフトウェア エージェントおまたはイベント ディテクタに依存して、特定のシステム イベントが発生したときに通知します。イベントを検出すると、EEM は修正アクションを開始できます。このアクションは、*policies* という名前のルーチンに規定されています。収集されたイベントに対してアクションを適用できるようにするには、ポリシーを登録しておく必要があります。ポリシーを登録しないと、アクションは発生しません。登録されたポリシーにより、検出される特定のイベント、およびそのイベントが検出された場合に実行される修正アクションが EEM に通知されます。このようなイベントが検出されると、EEM により対応するポリシーがイネーブル化されます。登録されたポリシーは、いつでもディセーブルにできます。

EEM は、システムの各プロセスによって達成された信頼性の評価をモニタリングし、システムが全体的な信頼性または可用性を脅かすコンポーネントを検出できるようにします。

このモジュールでは、ネットワークで EEM ポリシーを設定して管理し、Tool Command Language（Tcl）スクリプトを使用して EEM ポリシーの書き込みおよびカスタマイズを実行しての障害とイベントを処理するために必要なタスクについて説明します。

Embedded Event Manager ポリシーの設定および管理の前提条件

適切なタスク ID を含むタスク グループに関連付けられているユーザ グループに属している必要があります。このコマンド リファレンスには、各コマンドに必要なタスク ID が含まれます。ユーザ グループの割り当てが原因でコマンドを使用できないと考えられる場合、AAA 管理者に連絡してください。

Embedded Event Manager ポリシーの設定および管理について

Event Management

Cisco IOS XR ソフトウェア システムの Embedded Event Manager（EEM）には、基本的にシステム イベント管理が含まれます。イベントは、システム内で起こった重要なオカレンス（エラーに限らず）です。Cisco IOS XR ソフトウェアの EEM は、これらのイベントを検出し、適切な応答を実行します。

システム管理者は、EEM を使用して、システムの現在の状態に基づいて適切なアクションを指定できます。たとえば、システム管理者は EEM を使用して、ハードウェア デバイスの交換が必要になったときに、電子メールによる通知を要求することができます。

EEM は、システムをモニタしてイベントを検出するルーチンである「イベント ディテクタ」と相互作用します。EEM は、syslog に提供したイベント ディテクタに依存して、特定のシステムイベントが発生したことを検出します。syslog メッセージとのパターン一致を使用し、タイマー イベント ディテクタにも依存して、特定の日時が生じたことを検出します。

イベントを検出すると、EEM は応答アクションを開始できます。これらのアクションは、ポリシー ハンドラと呼ばれるルーチンに含まれています。ポリシーは、Tcl API を使用してユーザにより書き込まれた Tcl スクリプト（EEM スクリプト）によって定義されます。イベント検出用のデータが収集されている間は、そのイベントに応答するポリシーが登録されるまでアクションは実行されません。登録時には、ポリシーから EEM に、ポリシーが特定のイベントを検索していることが通知されます。イベントを検出すると、EEM はポリシーをイネーブルにします。

EEM は、システムの各プロセスによって達成された信頼性の評価を監視します。テスト中にこれらのメトリックを使用して、どのコンポーネントが信頼性または可用性の目標に到達していないかを特定し、修正アクションを実行できるようにすることが可能です。

システム イベント処理

イベント通知を受信すると、EEM は次のアクションを実行します。

- 確立されたポリシー ハンドラを確認し、ポリシー ハンドラが存在する場合、EEM はコールバック ルーチン（*EEM* ハンドラ）を開始するか、Tool Command Language（Tcl）スクリプト（*EEM* スクリプト）を実行してポリシーを実装します。ポリシーには、組み込み EEM アクションが含まれる場合があります。
- イベント通知に加入しているプロセスに通知します。
- システム内の各プロセスの信頼性メトリック データを記録します。
- アプリケーション プログラム インターフェイス（API）を介して、EEM により維持されているシステム情報へのアクセスを提供します。

Embedded Event Manager スクリプト

イベントを検出すると、EEM はポリシーと呼ばれるルーチンで規定された修正アクションを開始できます。収集されたイベントに対してアクションを適用できるようにするには、ポリシーを登録しておく必要があります。ポリシーを登録しないと、アクションは発生しません。登録されたポリシーにより、検出する特定のイベント、およびそのイベントが検出された場合に実行される修正アクションが EEM に通知されます。このようなイベントが検出されると、EEM により対応するポリシーが実行されます。Tool Command Language（Tcl）は、ポリシーを定義するスクリプト言語として使用され、Embedded Event Manager スクリプトはすべて Tcl で記述されます。EEM スクリプトは、**event manager policy** コンフィギュレーション コマンドを使用して EEM で識別されます。EEM スクリプトは、**no event manager policy** コマンドが入力されない限り、EEM によるスケジューリングが可能なままになります。

さらに、IOS XR オペレーティング システムに付属のオンボード Tcl スクリプトを使用して、ユーザは独自の TCL ベースのポリシーを記述できます。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。EEM Tcl コマンド拡張の詳細については、を参照してください。 Embedded Event Manager ポリシー Tcl コマンド拡張カテゴリ （3 ページ）

EEM スクリプトの作成には、次の手順が含まれます。

- 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。

Embedded Event Manager ポリシー Tcl コマンド拡張カテゴリ

この表には、EEM ポリシー Tcl コマンド拡張のさまざまなカテゴリの一覧を示します。

表 *1 : Embedded Event Manager Tcl* コマンド拡張カテゴリ

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）	これらの Tcl コマンド拡張は、**event_register**_xxx ファミリのイベント固有コマンド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	これらの Tcl コマンド拡張は、**sys_reqinfo**_xxx ファミリのシステム固有情報コマンド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。

Embedded Event Manager 用のシスコ ファイル命名規則

すべての EEM ポリシー名、ポリシー サポート ファイル（たとえば、電子メール 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。これに関連し、EEM ポリシーのファイル名は次の仕様に従います。

- 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）。
- 指定された 1 つ目のイベントの 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。

EEM の電子メール 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、後続の電子メール テンプレートの使用状況を示す省略形で構成されます。

EEM 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

2 文字の省略形	仕様
ap	event_register_appl
ct	event_register_counter
st	event_register_stat
no	event_register_none
oi	event_register_oir
pr	event_register_process
sl	event_register_syslog
tm	event_register_timer
ts	event_register_timer_subscriber
wd	event_register_wdsysmon

Embedded Event Manager の組み込みアクション

EEM の組み込みアクションは、EEM ハンドラが動作するときにハンドラから要求できます。

次の表に、各 EEM ハンドラ要求またはアクションを示します。

表 *3 : Embedded Event Manager* の組み込みアクション

Embedded Event Manager の組み込みアクション	説明
syslog へのメッセージの記録	メッセージを syslog に送ります。このアクションに対する引数は、優先度と記録するメッセージです。
CLI コマンドの実行	指定されたチャネル ハンドラにコマンドを書き込み、**cli_exec** コマンド拡張を使用してコマンドを実行します。
syslog メッセージの生成	**action_syslog** Tcl コマンド拡張を使用して、メッセージをログに記録します。
手動による EEM ポリシーの実行	**event manager run** コマンドを モードでポリシーを実行している間に、ポリシー内で EEM ポリシーを実行します。

Embedded Event Manager の組み込みアクション	説明
アプリケーション固有のイベントのパブリッシュ	**event_publish appl** Tcl コマンド拡張を使用して、アプリケーション固有のイベントをパブリッシュします。
Cisco IOS ソフトウェアのリロード	EEM **action_reload** コマンドを使用して、ルータをリロードします。
システム情報の要求	システム情報を収集するための、ポリシーによる sys_reqinfo_xxx ファミリのシステム固有情報コマンドを表します。
ショート メールの送信	シンプル メール転送プロトコル（SMTP）を使用して、電子メールを送信します。
カウンタの設定または変更	カウンタの値を変更します。

EEM ハンドラは、CLI コマンドを実行できる必要があります。Tcl スクリプトの中からの CLI コマンドの実行を許可するために、Tcl シェルでコマンドを使用できます。

アプリケーション固有の組み込みイベント管理

どの Cisco IOS XR ソフトウェア アプリケーションも、アプリケーション定義のイベントを定義してパブリッシュできます。アプリケーション定義のイベントは、コンポーネント名とイベント名の両方を含む名前で識別され、アプリケーション開発者が独自のイベント ID を割り当てることができます。アプリケーションで定義されたイベントは、サブスクライバがいない場合でも、Cisco IOS XR ソフトウェア コンポーネントによって発行できます。この場合、イベントは EEM によって解除されるため、サブスクライバはアプリケーション定義のイベントを必要に応じて受信できます。

システム イベントを受信するためにサブスクライブする EEM スクリプトは、次の順序で処理されます。

1. 次の CLI コンフィギュレーション コマンドが入力されます：**event manager policy** *scriptfilename* **username** *username*
2. EEM は EEM スクリプトをスキャンして **eem event event_type** キーワードを探し、指定したイベントに対してスケジュールされるように EEM スクリプトをサブスクライブします。
3. イベント ディテクタがイベントを検出し、EEM に通知します。
4. EEM はイベント処理をスケジュールし、EEM スクリプトが実行されます。
5. EEM スクリプト ルーチンが戻ります。

イベント検出とリカバリ

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。イベント ディテクタは、他の Cisco IOS XR ソフトウェア コンポーネントと EEM の間のインターフェイスを提供する個別のプログラムです。イベント ディテクタ（イベント パブリッシャ）はイベントを選別し、イベント サブスクライバ（ポリシー）によって提供されたイベント仕様と一致するときに、それら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。

EEM イベントは、システム内で何か重要なことが起きたことを示す通知として定義されます。イベントには次の 2 つのカテゴリがあります。

- システム EEM イベント
- アプリケーション定義イベント

システム EEM イベントは EEM に組み込まれており、イベントを発生する障害ディテクタに基づいてグループ化されます。API の中で定義されたシンボリックな ID で識別されます。

一部の EEM システム イベントは、アプリケーションがモニタリングを要求したかどうかにかかわらず EEM によってモニタされます。そのようなイベントを、組み込み EEM イベントと呼びます。他の EEM イベントは、アプリケーションが EEM イベント モニタリングを要求した場合のみモニタされます。EEM イベント モニタリングは、EEM アプリケーション API または EEM スクリプティング インターフェイスを通じて要求されます。

一部のイベント ディテクタは、同じセキュア ドメイン ルータ（SDR）または管理プレーンの中の他のハードウェア カードに分散させて、それらのカード上で動作する分散コンポーネントをサポートできます。

次のイベント ディテクタがサポートされています。

System Manager イベント ディテクタ

System Manager イベント ディテクタには、次の 4 つの役割があります。

- プロセス信頼性メトリック データを記録します。
- 未処理の EEM イベント モニタリング要求があるプロセスをスクリーニングします。
- スクリーニング条件に一致するプロセスのためのイベントをパブリッシュします。
- スクリーニング条件に一致しないイベントについて、デフォルトのアクションを実行するように System Manager に依頼します。

System Manager イベント ディテクタは、System Manager と通信して、プロセスの起動通知と終了通知を受信します。この通信は、System Manager が使用可能なプライベート API を通じて行われます。オーバーヘッドを最小化するため、API の一部は System Manager プロセス空間の中にあります。プロセスが終了するとき、System Manager は、イベント ディテクタ API を呼び出す前に、ヘルパープロセスを起動します（process.startup ファイルで指定されている場合）。

プロセスはコンポーネントID、System Managerによって割り当てられたジョブID、またはロードモジュールのパス名にプロセス インスタンス ID を加えたもので識別されます。プロセス インスタンス ID は、プロセスを同じパス名の他のプロセスと区別するために割り当てられる整数です。プロセスの最初のインスタンスにはインスタンス ID 値 1 が割り当てられ、次に 2 というように割り当てられます。

System Manager イベント ディテクタは、次の表に示す EEM イベントの EEM イベント モニタリング要求を処理します。

表 *4 : System Manager* イベント ディテクタ イベント モニタリング要求

Embedded Event Manager イベント	説明
プロセス正常終了 EEM イベント：組み込み	スクリーニング条件に一致するプロセスが終了するときに発生します。
プロセス異常終了 EEM イベント：組み込み	スクリーニング条件に一致するプロセスが異常終了するときに発生します。
プロセス起動 EEM イベント：組み込み	スクリーニング条件に一致するプロセスが起動するときに発生します。

System Manager イベント ディテクタ プロセス異常終了イベントが発生した場合、デフォルトのアクションにより、System Manager の組み込み規則に従ってプロセスが再起動されます。

EEM と System Manager の間の関係は、プロセスの起動通知と終了通知を受信する目的で EEM により System Manager に提供されているプライベート API を通じて厳格に行われます。System Manager が API を呼び出すとき、信頼性メトリック データが収集され、EEM イベント一致のためにスクリーニングが実行されます。一致が見つかった場合、System Manager イベント ディテクタにメッセージが送信されます。プロセス異常終了の場合、EEM がプロセスの再起動を処理することを通知して戻ります。一致しない場合、System Manager がデフォルトのアクションを適用すべきことを通知して戻ります。

タイマー サービス イベント ディテクタ

タイマー サービス イベント ディテクタは、時間に関連する EEM イベントを実装します。これらのイベントは、複数のプロセスが同じ EEM イベントの通知を待つことができるように、ユーザ定義 ID を通じて識別されます。

タイマー サービス イベント ディテクタは、日付/時刻経過 EEM イベントの EEM イベント モニタリング要求を処理します。このイベントは、現在の日付または時刻が、アプリケーションによって要求された指定の日付または時刻を過ぎた場合に発生します。

syslog イベント ディテクタ

syslog イベント ディテクタは、syslog EEM イベントのための syslog メッセージ スクリーニングを実現します。このルーチンは、プライベート API を通じて syslog デーモンと通信します。オーバーヘッドを最小化するため、API の一部は syslog デーモン プロセスの中にあります。

メッセージ重大度コードまたはメッセージ テキスト フィールドに対するスクリーニング機能を利用できます。

syglog イベント ディテクタは、次の表に示すイベントの EEM イベント モニタリング要求を処理します。

表 *5 : syslog* イベント ディテクタイベント モニタリング要求

Embedded Event Manager イベント	説明
syslog メッセージ EEM イベント	ログに記録されたばかりのメッセージに対して発生します。syslog メッセージの重大度コードまたは syslog メッセージ テキストパターンのいずれかが一致した場合に発生します。いずれも、アプリケーションが syslog メッセージ EEM イベントを要求するときに指定できます。
プロセス イベント マネージャ EEM イベント：組み込み	指定したプロセスのイベント処理カウントが指定した最大値以上になるか、指定した最小値以下になった場合に発生します。

None イベント ディテクタ

None イベント ディテクタは、Cisco IOS XR ソフトウェア **event manager run** CLI コマンドが EEM ポリシーを実行したときにイベントをパブリッシュします。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 ポリシーは識別される必要があり、手動での実行が許可されるように、**event manager run** コマンドが実行される前に登録される必要があります。

イベント マネージャの None ディテクタを使用すると、CLI を使用して Tcl スクリプトを実行できます。スクリプトは実行前に登録します。Cisco IOS XR ソフトウェア バージョンは、Cisco IOS EEM と同様の構文を備えているため（詳細は該当する EEM のマニュアルを参照してください）、Cisco IOS EEM を使用して作成したスクリプトは、最小限の変更により Cisco IOS XR ソフトウェアで実行できます。

Watchdog System Monitor イベント ディテクタ

Cisco IOS XR ソフトウェア用の Watchdog System Monitor（IOSXRWDSysMon）イベント ディテクタ

Cisco IOS XR ソフトウェア Watchdog System Monitor イベント ディテクタは、次のいずれかが発生したときにイベントをパブリッシュします。

- Cisco IOS XR ソフトウェア プロセスでの CPU の利用率がしきい値を超えたとき。
- Cisco IOS XR ソフトウェア プロセスでのメモリの利用率がしきい値を超えたとき。

（注） Cisco IOS XR ソフトウェア プロセスは、Cisco IOS XR ソフトウェア モジュール方式プロセスと区別するために使用されます。

同時に2つのイベントがモニタリングされることがあります。指定されたしきい値を超えるために1つのイベントを必要とするか、両方のイベントを必要とするかを、イベントパブリッシング基準で指定できます。

Cisco IOS XR ソフトウェア Watchdog System Monitor イベント ディテクタは、以下の表に示すイベントを処理します。

表 *6 : Watchdog System Monitor* イベント ディテクタ要求

Embedded Event Manager イベント	説明
プロセス パーセント CPU EEM イベント：組み込み	指定したプロセスの CPU 時間が、使用可能 CPU 時間の指定した最大パーセンテージ以上になるか、使用可能 CPU 時間の指定した最小パーセンテージ以下になった場合に発生します。
合計パーセント CPU EEM イベント：組み込み	指定したプロセッサ コンプレックスの CPU 時間が、使用可能 CPU 時間の指定した最大パーセンテージ以上になるか、使用可能 CPU 時間の指定した最小パーセンテージ以下になった場合に発生します。
プロセス パーセント メモリ EEM イベント：組み込み	指定したプロセスで使用されているメモリが、指定した値だけ増加または減少した場合に発生します。
合計パーセント使用可能メモリ EEM イベント：組み込み	指定したプロセッサ コンプレックスの使用可能メモリが、指定した値だけ増加または減少した場合に発生します。
合計パーセント使用メモリ EEM イベント：組み込み	指定したプロセッサ コンプレックスの使用メモリが、指定した値だけ増加または減少した場合に発生します。

Cisco IOS XR ソフトウェア モジュール方式のための Watchdog System Monitor（WDSysMon）イベント ディテクタ

Cisco IOS XR ソフトウェア ソフトウェア モジュール方式 Watchdog System Monitor イベント ディテクタは、Cisco IOS XR ソフトウェア モジュール方式プロセスの無限ループ、デッドロック、メモリ リークを検出します。

分散イベント ディテクタ

EEM イベント ディテクタと通信し、非常に独立した実装を持つ、分散したハードウェア カード上で動作する Cisco IOS XR ソフトウェア コンポーネントには、分散 EEM イベント ディテクタが必要です。分散イベント ディテクタでは、EEM 通信チャネルへのローカル ハードウェ

アカードをアクティブにしなくても、ローカルプロセスのEEMイベントをスケジューリングできます。

次のイベント ディテクタが Cisco IOS XR ソフトウェア ラインカードで動作します。

- System Manager 障害ディテクタ
- Wdsysmon 障害ディテクタ
- Counter イベント ディテクタ
- OIR イベント ディテクタ
- Statistic イベント ディテクタ

Embedded Event Manager イベントのスケジューリングおよび通知

EEM ハンドラがスケジュールされている場合、EEM ハンドラは、イベント要求を作成するプロセスのコンテキスト（EEM スクリプトの場合は、Tcl シェル プロセス コンテキスト）で動作します。EEM ハンドラを実行するプロセスに対して発生するイベントの場合、イベント スケジューリングは、ハンドラが終了するまでブロックされます。代わりに、定義されたデフォルトのアクション（存在する場合）が実行されます。

EEM サーバは、要求された場合に、クライアント プロセスの再起動にまたがって、イベント スケジューリングと通知項目が格納されたキューを保持します。

信頼性統計情報

システムの信頼性メトリック データが EEM によって保持されています。データはチェックポイントに定期的に書き込まれます。信頼性メトリック データは、各ハードウェア カードと、System Manager によって処理される各プロセスについて保持されます。

ハードウェア カードの信頼性メトリック データ

ハードウェア カードの信頼性メトリック データは、ディスク ID で索引付けされた表に記録されます。

ハードウェア カードによって保持されるデータは、次のとおりです。

- 最新の起動時刻
- 最新の正常終了時刻（制御されたスイッチオーバー）
- 最新の異常終了時刻（非同期スイッチオーバー）
- 最新の異常のタイプ
- 累積使用可能時間
- 累積使用不能時間
- ハードウェア カード開始回数

- ハードウェア カード正常シャットダウン回数
- ハードウェア カード異常シャットダウン回数

プロセス信頼性メトリック データ

信頼性メトリック データは、System Manager によって処理される各プロセスについて保持されます。このデータには、プライマリまたはバックアップ ハードウェア カードで動作するスタンバイ プロセスが含まれています。データは、ハードウェア カード ディスク ID、プロセス パス名、複数のインスタンスがあるプロセスの場合はプロセス インスタンスを組み合わせたものでインデックスが作成されたテーブルに記録されます。

プロセスの終了には次の場合が含まれます。

- 正常終了：プロセスは終了値 0 で終了します。
- プロセスによる異常終了：プロセスは 0 でない終了値で終了します。
- Linux による異常終了：Linux オペレーティング システムがプロセスを中断します。
- プロセス終了 API によるプロセス異常終了：プロセス終了 API によりプロセスが終了します。

プロセスが保持するデータは次のとおりです。

- 最新のプロセス起動時刻
- 最新のプロセス正常終了時刻
- 最新のプロセス異常終了時刻
- 最新のプロセス異常終了のタイプ
- 以前の 10 回のプロセス終了時刻とタイプ
- 累積プロセス使用可能時間
- 累積プロセス使用不能時間
- 累積プロセス実行時間（プロセスが実際に CPU で動作した時間）
- 起動回数
- 正常終了回数
- 異常終了回数
- 過去 60 分間の異常障害回数
- 過去 24 時間の異常障害回数
- 過去 30 日間の異常障害回数

Embedded Event Manager ポリシーの設定および管理方法

環境変数の設定

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 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。

通例として、シスコで定義されているすべての環境変数の名前は、他の変数と区別するためにアンダースコア文字で始まります（_show_cmd など）。

event manager environment*var-name var-value* コマンドを使用して、環境変数と値を設定できます。

event manager environment コマンドを使用して設定される前または後に、すべての EEM 環境変数の名前および値を表示するには、**show event manager environment** コマンドを使用します。

設定例

次の例では、一連の EEM 環境変数を定義する方法を示します。

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# event manager environment _cron_entry 0-59/2 0-23/1 * *
0-7
RP/0/RP0/CPU0:Router(config)# event manager environment _email_from beta@cisco.com
RP/0/RP0/CPU0:Router(config)# event manager environment _email_to beta@cisco.com
RP/0/RP0/CPU0:Router(config)# commit
RP/0/RP0/CPU0:Router(config)# end
RP/0/RP0/CPU0:Router# show event manager environment

No. Name                            Value
1   _email_to                       beta@cisco.com
2   _cron_entry                     0-59/2 0-23/1 * * 0-7
3   _email_from                     beta@cisco.com
RP/0/RP0/CPU0:Router#
```

Embedded Event Manager ポリシーの登録

イベントがトリガーされたときにポリシーを実行するため、EEM ポリシーを登録する必要があります。EEM ポリシーの登録は、**event manager policy** コマンドを使用して実行されます。EEM スクリプトは、このコマンドの **no** 形式が入力されない限り、EEM によるスケジューリングが可能です。ポリシーを登録する前に、**show event manager policy available** コマンドを使用して、登録できる EEM ポリシーを表示します。

EEMは、ポリシー自体に含まれているイベントの指定内容に基づいて、ポリシーをスケジューリングおよび実行します。**event manager policy** コマンドが呼び出されると、EEM はポリシーを確認し、指定されたイベントが発生した場合に実行されるように登録します。

EEM ポリシーの登録時に以下を指定する必要があります。

- **username**：スクリプトを実行するユーザ名を指定します。
- **persist-time**：ユーザ名の認証が有効な秒数を定義します。このキーワードは任意です。デフォルト **persist-time** は 3600 秒（1 時間）です。
- **system** または **user**：ポリシーをシステム定義またはユーザ定義のポリシーとして指定します。このキーワードは任意です。

（注） EEM ポリシーを登録する前に、AAA 認可（**aaa authorization eventmanager** コマンドなど）を設定する必要があります。AAA 認可の設定の詳細については、『*Configuring AAA Services on Cisco IOS XR ソフトウェア* 』の「*Configuring AAA Services*」モジュールを参照してください。

ポリシーが登録されると、それらの登録は **show event manager policy registered** コマンドによって確認できます。

設定例

次に、ユーザ定義の EEM ポリシーを登録する例を示します。

```
RP/0/RP0/CPU0:Router# show event manager policy available
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# event manager policy cron.tcl username tom type user
RP/0/RP0/CPU0:Router# show event manager policy registered
```

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

ここでは、Tool Command Language（Tcl）スクリプトを使用して、Cisco IOS XR ソフトウェアの障害とイベントを処理するための、カスタマイズした Embedded Event Manager（EEM）ポリシーを作成する方法について説明します。

この項では、次の作業について説明します。

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。

（注） Tcl スクリプト言語で作成されたポリシーが使用できる必要があります。サンプル ポリシーがシステム ポリシー ディレクトリに格納されています。

設定例

次に、EEM ポリシーを登録して定義する例を示します。

```
RP/0/RP0/CPU0:Router# show event manager environment all
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# event manager environment _cron_entry 0-59/2 0-23/1 * *
0-7
RP/0/RP0/CPU0:Router(config)# event manager policy tm_cli_cmd.tcl username user_a type
system
RP/0/RP0/CPU0:Router(config)# commit
RP/0/RP0/CPU0:Router# show event manager policy registered system
```

（注） EEM ポリシーの登録を解除するには、**no event manager policy** コマンドを使用します。このコマンドを使用すると、EEM ポリシーが実行コンフィギュレーション ファイルから削除されます。

EEM ポリシー実行の一時停止

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。必要に応じて、**event manager scheduler suspend** コマンドを使用してすべての EEM ポリシーの実行をただちに一時停止することができます。

設定例

次に、すべての EEM ポリシーの実行を一時停止する例を示します。

```
RP/0/RP0/CPU0:Router# show event manager policy registered system
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# event manager scheduler suspend
RP/0/RP0/CPU0:Router(config)# commit
```

EEM ポリシーを格納するディレクトリの指定

ディレクトリは、ユーザ定義のポリシー ファイルまたはユーザ ライブラリ ファイルを格納するために不可欠です。EEM ポリシーを記述する予定がない場合は、ディレクトリを作成する必要はありません。EEM は、**event manager policy** *policy-name* **user** コマンドを入力すると、ユーザ ポリシー ディレクトリを検索します。EEM に対して識別する前にユーザ ポリシー ディレクトリを作成するには、**mkdir** コマンドを使用します。ユーザ ポリシー ディレクトリを作成した後で、ユーザ ポリシー ディレクトリにポリシー ファイルをコピーするには、copy コマンドを使用します。**show event manager directory user [library | policy]** コマンドを使用して、EEM ユーザ ライブラリ ファイルまたはユーザ定義のポリシー ファイルに使用するディレクトリを表示できます。

設定例

次に、ユーザ ライブラリ ファイルの格納に使用するディレクトリを指定する例を示します。

```
RP/0/RP0/CPU0:Router# show event manager directory user library
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# event manager directory user library disk0:/usr/lib/tcl
RP/0/RP0/CPU0:Router(config)# commit
```

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 ポリシー構造と要件の詳細については、を参照してください。 TCL を使用した EEM ポリシー：詳細 （27 ページ）

手順

	コマンドまたはアクション	目的
ステップ 1	**show event manager policy available** [**system** | **user**] **例**： `RP/0/RP0/CPU0:Router# show event manager policy available`	登録可能な EEM ポリシーを表示します。
ステップ 2	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。	—
ステップ 3	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 コマンド拡張を選択し、ポリシーに追加します。有効なイベント登録 Tcl コマンド拡張を次に示します。 • event_register_appl • event_register_counter • event_register_stat • event_register_wdsysmon • event_register_oir • event_register_process • event_register_syslog • event_register_timer • event_register_timer_subscriber • event_register_hardware • event_register_none
ステップ 4	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 XR 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

	コマンドまたはアクション	目的
		す。::cisco 階層の下には、2 つの名前空間があります。各名前空間に属する名前空間と EEM Tcl コマンド拡張カテゴリは次のとおりです。 • ::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） 適切な名前空間がインポートされていることを確認するか、前述のコマンドを使用する場合は修飾されたコマンド名を使用します。
ステップ 5	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、EEM コンフィギュレーション コマンド **event manager environment** を使用します。規則とし

	コマンドまたはアクション	目的
		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** 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 EEM 環境変数には、電子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、次に示す変数が設定されている必要があります。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。 • **_email_server**：電子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（たとえば mailserver.example.com） • **_email_to**：電子メールの送信先アドレス（たとえば engineering@example.com） • **_email_from**：電子メールの送信元アドレス（たとえば devtest@example.com） • **_email_cc**：電子メールのコピーの送信先アドレス（たとえば manager@example.com）
ステップ 6	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。

	コマンドまたはアクション	目的
		• 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 7	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 8	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。	値 0 は、デフォルトのアクションを実行しないことを意味します。0 以外の値は、デフォルトのアクションを実行する必要があることを意味します。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 9	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、および _cerr_str）。

	コマンドまたはアクション	目的
ステップ 10	新しいファイル名で Tcl スクリプトを保存し、Tcl スクリプトをルータにコピーします。	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（「表 2 : 2 文字の省略形の指定 （5 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、「Embedded Event Manager 用のシスコ ファイル命名規則（4 ページ）」を参照してください。 ルータ上のフラッシュ ファイル システム（通常は disk0:）に、ファイルをコピーします。
ステップ 11	**configure**	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 12	**event manager directory user** { **library** *path* | **policy** *path*} 例： `RP/0/RP0/CPU0:Router(config)# event manager directory user library disk0:/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。
ステップ 13	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]] 例： `RP/0/RP0/CPU0:Router(config)# event`	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
	`manager policy test.tcl username user_a type user`	
ステップ 14	**commit**	
ステップ 15	ポリシーを実行し、ポリシーを観察します。	—
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。	—

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

	コマンドまたはアクション	目的

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
		`proc test1 {} {`
		`  puts "In procedure test1"`
		`}`
		`proc test2 {} {`
		`  puts "In procedure test2"`
		`}`
		lib2.tcl
		`proc test3 {} {`
		`  puts "In procedure test3"`
		`}`

ステップ 2	**tclsh** 例： `workstation% tclsh`	Tcl シェルを開始します。

	コマンドまたはアクション	目的

ステップ 3	**auto_mkindex** *directory_name* ***.tcl**
	例：
	`workstation% auto_mkindex eem_library *.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。
		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
		`# 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 ルータ上のユーザ 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、ステップ 1 から Tcl ライブラリ ファイルをコピーし、ステップ 3 から tclIndex ファイルをコピーします。	—

	コマンドまたはアクション	目的

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、ターゲットルータ上でユーザ定義 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。	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
		`::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	**configure**	
ステップ 7	**event manager directory user library** *path* 例： `RP/0/RP0/CPU0:Router(config)# event manager directory user library disk0:/eem_library`	EEM ユーザ ライブラリのディレクトリを指定します。これは、ステップ 4 のファイルがコピーされたディレクトリです。
ステップ 8	**event manager directory user policy** *path* 例： `RP/0/RP0/CPU0:Router(config)# event manager directory user policy disk0:/eem_policies`	EEM ユーザ ポリシーのディレクトリを指定します。これは、ステップ 5 のファイルがコピーされたディレクトリです。
ステップ 9	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]] 例： `RP/0/RP0/CPU0:Router(config)# event manager policy libtest.tcl username user_a`	ユーザ定義の EEM ポリシーを登録します。

	コマンドまたはアクション	目的
ステップ 10	**event manager run** *policy* [*argument*] 例： `RP/0/RP0/CPU0:Router(config)# event manager run libtest.tcl`	手動で EEM ポリシーを実行します。
ステップ 11	**commit**	

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。Tcl パッケージは、Tcl **package** キーワードを使用してサポートされます。

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、ディレクトリに含まれるすべての 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

	コマンドまたはアクション	目的
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** 例： `workstation% tclsh`	Tcl シェルを開始します。

	コマンドまたはアクション	目的

ステップ 3	**pkg_mkindex** *directory_name* ***.tcl**
	例：
	`workstation% pkg_mkindex eem_library *.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** コマンドを実行することを推奨します。ディレクトリ内で**pkg_mkindex** コマンドを実行すると、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。
		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
		`# 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 ルータ上のユーザ 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、ステップ 1 から Tcl パッケージ ファイルをコピーし、ステップ 3 から pkgIndex ファイルをコピーします。	—

	コマンドまたはアクション	目的

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、ターゲットルータ上でユーザ定義 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。	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。
		packagetest.tcl
		`::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	**configure**	
ステップ 7	**event manager directory user library** *path* 例： `RP/0/RP0/CPU0:Router(config)# event manager directory user library disk0:/eem_library`	EEM ユーザ ライブラリのディレクトリを指定します。これは、ステップ 4 のファイルがコピーされたディレクトリです。
ステップ 8	**event manager directory user policy** *path* 例： `RP/0/RP0/CPU0:Router(config)# event manager directory user policy disk0:/eem_policies`	EEM ユーザ ポリシーのディレクトリを指定します。これは、ステップ 5 のファイルがコピーされたディレクトリです。
ステップ 9	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]] 例： `RP/0/RP0/CPU0:Router(config)# event manager policy packagetest.tcl username user_a`	ユーザ定義の EEM ポリシーを登録します。
ステップ 10	**event manager run** *policy* [*argument*] 例：	手動で EEM ポリシーを実行します。

	コマンドまたはアクション	目的
	`RP/0/RP0/CPU0:Router(config)# event manager run packagetest.tcl`	
ステップ 11	**commit**	

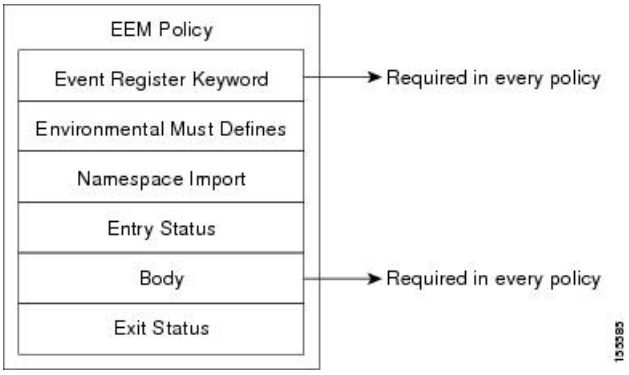
TCL を使用した EEM ポリシー：詳細

この項では、TCL を使用した EEM ポリシーのプログラミングに関する詳細な概念情報を示します。

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の、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 *1 : Tcl* ポリシーの構造と要件



各ポリシーの開始は、**event_register** Tcl コマンド拡張を使用して検出するために、イベントを示し、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 コード例を示します。

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

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、次のものを含めることができます。

- 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。
- 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。

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。

開始ステータスの指定には、次の 3 種類の値のいずれかを使用できます。

- 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）
- Undefined（実行された前のポリシーがない）

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、EEM 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。値 0 は、デフォルトのアクションを実行しないことを意味します。0 以外の値は、デフォルトのアクションを実行する必要があることを意味します。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 変数が設定されると、他の 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、および _cerr_str）。

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

```
XYSSSSSSSSSSSSSEEEEEEEEPPPPPPPP
```

次の表に示されているように、この 32 ビットの整数は変数に分けられます。

表 *7*：*_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。 この変数特有の 4 つのエラー クラス エンコーディングについては、「表 8：エラー クラス エンコーディング （29 ページ）」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 $_cerr_sub_num に含まれています。
EEEEEEEE	サブシステム固有のエラー番号（8 ビット＝値 256）。このセグメントは、32 ビット シーケンスの次の 8 ビットで、このエラー番号に対応する文字列は、$_cerr_sub_err に含まれています。

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
```

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 *8*：エラー クラス エンコーディング

エラー戻り値	エラー クラス
00	CERR_CLASS_SUCCESS

エラー戻り値	エラー クラス
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。