Cisco IOS CLI を使用した EEM ポリシーの記述について

Cisco IOS CLI を使用した EEM 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前に、「Embedded Event Manager の概要」の章で説明されている概念を十分に理解しておく必要があります。
- **action cns-event** コマンドを使用する場合は、Cisco Networking Services（CNS）イベントゲートウェイへのアクセスを設定する必要があります。
- **action force-switchover** コマンドを使用する場合は、デバイスでセカンダリプロセッサを設定する必要があります。
- **action snmp-trap** コマンドを使用した場合、**snmp-server enable traps event-manager** コマンドを有効にして、SNMP トラップが Cisco IOS デバイスから SNMP サーバーに送信されることを許可する必要があります。その他の関連する **snmp-server** コマンドを設定する必要もあります。詳細については、**action snmp-trap** コマンドのページを参照してください。

Cisco IOS CLI を使用した EEM ポリシーの記述について

Cisco IOS CLI を使用した EEM ポリシーの記述について

Embedded Event Manager ポリシー

EEM では、イベントを監視し、監視対象のイベントが発生したときやしきい値を超えたときに情報通知や是正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、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。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 ステートメントがサポートされています。**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内では、**event** コンフィギュレーション コマンドを 1 つだけが使用できます。アプレット コンフィギュレーション モードが終了し、**event** コマンドが存在しない場合は、このアプレットにイベントが関連付け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、このアプレットは登録されたと見なされません。この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。アプレットを登録解除することなく修正することが安全な方法です。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。

action コンフィギュレーション コマンドは、*label* 引数を使用して一意に識別できます。この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* 引数を使用してソート キーとして、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。

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。

EEM スクリプト

スクリプトは、ネットワーキング デバイスの外部で ASCII エディタを使用して定義します。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。

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 ポリシーの記述には、次の作業が含まれます。

- ポリシーが実行されるイベントの選択。
- イベントの記録およびイベントへの対応に関連付けられたイベント ディテクタ オプションの定義。
- イベント発生後に実行されるアクションの選択。

シスコは、Tcl に EEM ポリシー開発を促進するキーワード拡張機能の形式を加えました。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。Tcl を使用して EEM ポリシーを記述する方法については、「Tcl を使用した Embedded Event Manager ポリシーの記述」の章を参照してください。

EEM アプレットに使用される Embedded Event Manager 組み込み環境変数

EEM 組み込み環境変数は、シスコ定義の環境変数のサブセットです。組み込み変数は、EEM アプレットでだけ利用できます。組み込み変数は、読み込み専用であるか、または読み込みおよび書き込み用のいずれかです。これらの変数は、1 個の特定のイベント ディテクタまたはすべてのイベント ディテクタに適用されます。次の表に、イベント ディテクタおよびサブイベントごとの読み込み専用のシスコ組み込み環境変数の一覧をアルファベット順に示します。

表 *1 : EEM* 組み込み環境変数（読み取り専用）

環境変数	説明
すべてのイベント	
_event_id	パブリッシュされた該当イベントの ID を示す一意の番号。同一のイベントで複数のポリシーを実行可能であり、その場合、各ポリシーは同一の event_id を保持します。
_event_type	イベントのタイプ。
_event_type_string	イベントをトリガーしたイベントの種類を識別する ASCII 文字列。

環境変数	説明
_event_pub_sec _event_pub_msec	EEM に対してイベントがパブリッシュされた、秒単位およびミリ秒単位の時間。
_event_severity	イベントの重大度。
Application-Specific イベント ディテクタ	
_application_component_id	イベント アプリケーション コンポーネント ID。
_application_data1	イベントがパブリッシュされたときにアプリケーション固有のイベントに渡される、環境変数の値、文字テキスト、またはその両方の組み合わせ。
_application_data2	イベントがパブリッシュされたときにアプリケーション固有のイベントに渡される、環境変数の値、文字テキスト、またはその両方の組み合わせ。
_application_data3	イベントがパブリッシュされたときにアプリケーション固有のイベントに渡される、環境変数の値、文字テキスト、またはその両方の組み合わせ。
_application_data4	イベントがパブリッシュされたときにアプリケーション固有のイベントに渡される、環境変数の値、文字テキスト、またはその両方の組み合わせ。
_application_sub_system	イベント アプリケーション サブシステム番号。
_application_type	アプリケーションのタイプ。
CLI イベント ディテクタ	
_cli_msg	CLIイベントをトリガーした、完全に展開されたメッセージ。
_cli_msg_count	イベントがパブリッシュされる前にメッセージ一致が発生した回数。
Counter イベント ディテクタ	
_counter_name	カウンタの名前。
_counter_value	カウンタの値。
Enhanced Object Tracking イベント ディテクタ	
_track_number	トラッキング対象オブジェクトの数。

環境変数	説明
_track_state	トラッキング対象オブジェクトの状態（ダウン、またはアップ）。
Generic Online Diagnostics（GOLD）イベント ディテクタ	
_action_notify	GOLD イベント フラグのアクション通知情報（False または True）。
_event_severity	イベントの重大度（Normal、Minor、または Major）。
_gold_bl	起動診断レベル（次のいずれかの値）。 • 0：完全診断 • 1：最小診断 • 2：バイパス診断
_gold_card	GOLD 障害イベントが検出されたカード。
_gold_cf *testnum*	連続的な障害。*testnum* はテスト番号。たとえば、**_gold_cf3** は、テスト 3 の連続的な障害の EEM 組み込み環境変数です。
_gold_ci	カード インデックス。
_gold_cn	カードの名前。
_gold_ec *testnum*	テスト エラー コード。*testnum* はテスト番号。たとえば、**_gold_ec3** は、テスト 3 のエラー コードの EEM 組み込み環境変数です。
_gold_lf *testnum*	最終障害時間。*testnum* はテスト番号。たとえば、**_gold_lf3** は、テスト 3 の最終障害時間の EEM 組み込み環境変数です。 タイムスタンプの形式は *mmm dd yyyy hh*:*mm*:*ss* です。例：Mar 11 2005 08:47:00。
_gold_new_failure	GOLD イベント フラグの新しいテスト障害情報（False または True）。

環境変数	説明
_gold_overall_result	総合診断結果、次のいずれかの値である。 • 0：OK • 3：マイナー エラー • 4：メジャー エラー • 14：結果不明
_gold_pc	ポート数。
_gold_rc *testnum*	テスト総実行回数。*testnum* はテスト番号。たとえば、**_gold_rc3** は、テスト 3 の総実行回数の EEM 組み込み変数です。
_gold_sn	カード シリアル番号。
_gold_sub_card	GOLD 障害イベントが検出されたサブカード。
_gold_ta *testnum*	テスト属性名。*testnum* はテスト番号。たとえば、**_gold_ta3** は、テスト 3 の属性の EEM 組み込み環境変数です。
_gold_tc	テスト数。
_gold_tf *testnum*	合計障害回数。*testnum* はテスト番号。たとえば、**_gold_tf3** は、テスト 3 の合計障害回数の EEM 組み込み変数です。
_gold_tn *testnum*	テストの名前。*testnum* はテスト番号。たとえば、**_gold_tn3** は、テスト 3 の名前の EEM 組み込み環境変数です。
_gold_tr *testnum*	テストの結果。*testnum* はテスト番号。たとえば、**_gold_tr6** はテスト 6 用の EEM 組み込み変数です。テスト 6 はポート単位のテストでも、デバイス単位のテストでもありません。 テスト結果は、次の値のうちのいずれかです。 • P：診断結果 Pass • F：診断結果 Fail • U：診断結果 Unknown

環境変数	説明
_gold_tr *testnum* **d** *devnum*	デバイスごとのテスト結果。*testnum* はテスト番号で、*devnum* はデバイス番号です。たとえば、**_gold_tr3d20** は、テスト 3、デバイス 20 のテスト結果の EEM 組み込み環境変数です。 テスト結果は、次の値のうちのいずれかです。 • P：診断結果 Pass • F：診断結果 Fail • U：診断結果 Unknown
_gold_tr *testnum* **p** *portnum*	ポートごとのテスト結果。*testnum* はテスト番号で、*portnum* はポート番号です。たとえば、**_gold_tr5p20** は、テスト 5、ポート 20 のテスト結果の EEM 組み込み環境変数です。 テスト結果は、次の値のうちのいずれかです。 • P：診断結果 Pass • F：診断結果 Fail • U：診断結果 Unknown
_gold_tt	テストのタイプ。次のうちのいずれかです。 • 1：起動診断 • 2：オンデマンド診断 • 3：スケジュール診断 • 4：モニターリング診断
Interface Counter イベント ディテクタ	
_interface_is_increment	現在のインターフェイスカウンタ値が、絶対値（0）か増分値（1）かを示す値。
_interface_name	モニターされるインターフェイスの名前。
_interface_parameter	モニターされるインターフェイス カウンタの名前。
_interface_value	現在のインターフェイスカウンタ値と比較される値。
None イベント ディテクタ	

環境変数	説明
_event_id	1 であれば挿入イベントを示し、2 であれば削除イベントを示す値。
_none_argc **_none_arg1** **_none_arg2** **_none_arg3** **_none_arg4** **_none_arg5** **_none_arg6** **_none_arg7** **_none_arg8** **_none_arg9** **_none_arg10** **_none_arg11** **_none_arg12** **_none_arg13** **_none_arg14** **_none_arg15**	Extensible Markup Language（XML）Simple Object Access Protocol（SOAP）コマンドからスクリプトに渡されるパラメータ。
OIR イベント ディテクタ	
_oir_event	1 であれば挿入イベントを示し、2 であれば削除イベントを示す値。
_oir_slot	OIR イベントのスロット番号。
Resource イベント ディテクタ	
_resource_configured_threshold	設定されている ERM しきい値。
_resource_current_value	ERM によって報告された、現在の値。
_resource_dampen_time	ERM 減衰時間、ナノ秒単位。
_resource_direction	ERM イベント方向。イベント方向は、アップ、ダウン、または、変更なしのうちのいずれかです。
_resource_level	ERM イベント レベル。イベント レベルは、Normal、Minor、Major、および Critical の 4 つです。
_resource_notify_data_flag	ERM 通知データ フラグ。

環境変数	説明
_resource_owner_id	ERM リソース オーナー ID。
_resource_policy_id	ERM ポリシー ID。
_resource_policy_violation_flag	ERM ポリシー違反フラグ（False または True）。
_resource_time_sent	ERM イベント時間、ナノ秒単位。
_resource_user_id	ERM リソース ユーザー ID。
RF イベント ディテクタ	0 であれば RF イベントでないことを示し、1 であれば RF イベントであることを示す値。
_rf_event	
Remote Procedure Call（RPC）イベント ディテクタ	値 0 はエラーがないことを示し、値 1 ～ 83 はエラーを示します。
_rpc_event	
_rpc_argc **_rpc_arg0** **_rpc_arg1** **_rpc_arg2** **_rpc_arg3** **_rpc_arg4** **_rpc_arg5** **_rpc_arg6** **_rpc_arg7** **_rpc_arg8** **_rpc_arg9** **_rpc_arg10** **_rpc_arg11** **_rpc_arg12** **_rpc_arg13** **_rpc_arg14**	XML SOAP コマンドからアプレットに渡されるパラメータ。
SNMP イベント ディテクタ	0 であれば exit イベントでないことを示し、1 であれば exit イベントであることを示す値。
_snmp_exit_event	

環境変数	説明
_snmp_oid	パブリッシュされるイベントの原因となった SNMP オブジェクト ID。
_snmp_oid_delta_val	現在の SNMP オブジェクト ID の値と、イベントが最後にトリガーされたときの実際の増分差異。
_snmp_oid_val	イベントがパブリッシュされたときの SNMP オブジェクト ID 値。
SNMP 通知イベント ディテクタ	
_snmp_notif_oid	ユーザー指定オブジェクト ID。
_snmp_notif_oid_val	ユーザー指定オブジェクト ID 値。
_snmp_notif_src_ip_addr	SNMP プロトコル データ ユニット（PDU）の発信元 IP アドレス。
_snmp_notif_dest_ip_addr	SNMP PDU の宛先の IP アドレス。
_x_x_x_x_x_x_x(varbinds)	SNMP PDU varbind 情報。
_snmp_notif_trunc_vb_buf	バッファの領域不足から varbind 情報が切り捨てられているかどうかを示します。
syslog イベント ディテクタ	
_syslog_msg	パブリッシュされるイベントの原因となる syslog メッセージ。
System Manager（Process）イベント ディテクタ	
_process_dump_count	Posix プロセスがダンプされた回数。
_process_exit_status	終了時の Posix プロセスの状態。
_process_fail_count	Posix プロセスが失敗した回数。
_process_instance	Posix プロセスのインスタンス数。
_process_last_respawn	最後に再生成された Posix プロセス。
_process_node_name	Posix プロセスのノード名。
_process_path	Posix プロセスのパス。
_process_process_name	Posix プロセスの名前。
_process_respawn_count	Posix プロセスが再生成された回数。

環境変数	説明
Timer イベント ディテクタ	
_timer_remain	タイマーの期限が切れるまでの使用可能時間。 （注） この環境変数は、CRON タイマーには使用できません。
_timer_time	最後のイベントがトリガーされた時刻。
_timer_type	タイマーのタイプ。
Watchdog System Monitor（IOSWDSysMon）イベント ディテクタ	
_ioswd_node	ルート プロセッサ（RP）レポーティング ノードのスロット番号。
_ioswd_num_subs	存在するサブイベントの数。
全 Watchdog System Monitor（IOSWDSysMon）サブイベント	
_ioswd_sub1_present **_ioswd_sub2_present**	サブイベント 1 またはサブイベント 2 の存在を示す値。値1は、サブイベントが存在することを示し、値 0 はサブイベントが存在しないことを示します。
_ioswd_sub1_type _ioswd_sub2_type	イベントのタイプ（cpu_proc、または mem_proc）。
Watchdog System Monitor（IOSWDSysMon）cpu_proc サブイベント	
_ioswd_sub1_path _ioswd_sub2_path	サブイベントのプロセス名。
_ioswd_sub1_period **_ioswd_sub2_period**	サブイベントの測定に使用される時間間隔（秒単位、オプションでミリ秒単位）。
_ioswd_sub1_pid _ioswd_sub2_pid	サブイベントのプロセス ID。
_ioswd_sub1_taskname **_ioswd_sub2_taskname**	サブイベントのタスク名。
_ioswd_sub1_value _ioswd_sub2_value	パーセンテージで測定されたサブイベントの CPU 使用率。
Watchdog System Monitor（IOSWDSysMon）mem_proc サブイベント	

環境変数	説明
_ioswd_sub1_diff _ioswd_sub2_diff	イベントをトリガーした差のパーセンテージの値。 （注） この変数は、**_ioswd_sub1_is_percent** 変数または **_ioswd_sub2_is_percent** 変数が 1 である場合に限って設定されます。
_ioswd_sub1_is_percent **_ioswd_sub2_is_percent**	値がパーセンテージであるかどうかを識別する番号。0 であれば値がパーセンテージではないことを意味し、1 であれば値がパーセンテージであることを意味します。
_ioswd_sub1_path _ioswd_sub2_path	サブイベントのプロセス名。
_ioswd_sub1_pid _ioswd_sub2_pid	サブイベントのプロセス ID。
_ioswd_sub1_taskname **_ioswd_sub2_taskname**	サブイベントのタスク名。
_ioswd_sub1_value _ioswd_sub2_value	パーセンテージで測定されたサブイベントの CPU 使用率。
Watchdog System Monitor（WDSysMon）イベント ディテクタ	
_wd_sub1_present _wd_sub2_present	サブイベント 1 またはサブイベント 2 の存在を示す値。値1は、サブイベントが存在することを示し、値 0 はサブイベントが存在しないことを示します。
_wd_num_subs	存在するサブイベントの数。
_wd_sub1_type _wd_sub2_type	イベントのタイプ（cpu_proc、cpu_tot、deadlock、dispatch_mgr、mem_proc、mem_tot_avail、または mem_tot_used）。
Watchdog System Monitor（WDSysMon）cpu_proc サブイベント	
_wd_sub1_node _wd_sub2_node	サブイベント RP レポーティング ノードのスロット番号。
_wd_sub1_period _wd_sub2_period	サブイベントの測定に使用される時間間隔（秒単位、オプションでミリ秒単位）。
_wd_sub1_procname **_wd_sub2_procname**	サブイベントのプロセス名。
_wd_sub1_value _wd_sub2_value	パーセンテージで測定されたサブイベントの CPU 使用率。

環境変数	説明
Watchdog System Monitor（WDSysMon）cpu_tot サブイベント	
_wd_sub1_node _wd_sub2_node	サブイベントRP レポーティング ノードのスロット番号。
_wd_sub1_period _wd_sub2_period	サブイベントの測定に使用される時間間隔（秒単位、オプションでミリ秒単位）。
_wd_sub1_value _wd_sub2_value	パーセンテージで測定されたサブイベントの CPU 使用率。
Watchdog System Monitor（WDSysMon）deadlock サブイベント	
_wd_sub1_entry_ [1-N]**_b_node** **_wd_sub2_entry_**[1-N]**_b_node**	サブイベントRP レポーティング ノードのスロット番号。
_wd_sub1_entry_ [1-N]**_b_pid** **_wd_sub2_entry_**[1-N]**_b_pid**	サブイベントのプロセス ID。
_wd_sub1_entry_ [1-N]**_b_procname** **_wd_sub2_entry_**[1-N]**_b_procname**	サブイベントのプロセス名。
_wd_sub1_entry_ [1-N]**_b_tid** **_wd_sub2_entry_**[1-N]**_b_tid**	サブイベントの時間 ID。
_wd_sub1_entry_ [1-N]**_node** **_wd_sub2_entry_**[1-N]**_node**	サブイベントRP レポーティング ノードのスロット番号。
_wd_sub1_entry_ [1-N]**_pid** **_wd_sub2_entry_**[1-N]**_pid**	サブイベントのプロセス ID。
_wd_sub1_entry_ [1-N]**_procname** **_wd_sub2_entry_**[1-N]**_procname**	サブイベントのプロセス名。
_wd_sub1_entry_ [1-N]**_state** **_wd_sub2_entry_**[1-N]**_state**	サブイベントの時間 ID。
_wd_sub1_entry_ [1-N]**_tid** **_wd_sub2_entry_**[1-N]**_tid**	サブイベントの時間 ID。
_wd_sub1_num_entries **_wd_sub2_num_entries**	サブイベントの数。
Watchdog System Monitor（WDSysMon）dispatch manager サブイベント	
_wd_sub1_node _wd_sub2_node	サブイベントRP レポーティング ノードのスロット番号。

環境変数	説明
_wd_sub1_period _wd_sub2_period	サブイベントの測定に使用される時間間隔（秒単位、オプションでミリ秒単位）。
_wd_sub1_procname **_wd_sub2_procname**	サブイベントのプロセス名。
_wd_sub1_value _wd_sub2_value	パーセンテージで測定されたサブイベントの CPU 使用率。
Watchdog System Monitor（WDSysMon）mem_proc サブイベント	
_wd_sub1_diff _wd_sub2_diff	イベントをトリガーした差のパーセンテージの値。 （注） この変数は、**_wd_sub1_is_percent** 変数または **_wd_sub2_is_percent** 変数が 1 である場合に限って設定されます。
_wd_sub1_is_percent **_wd_sub2_is_percent**	値がパーセンテージであるかどうかを識別する番号。0 であれば値がパーセンテージではないことを意味し、1 であれば値がパーセンテージであることを意味します。
_wd_sub1_node _wd_sub2_node	サブイベント RP レポーティング ノードのスロット番号。
_wd_sub1_period _wd_sub2_period	サブイベントの測定に使用される時間間隔（秒単位、オプションでミリ秒単位）。
_wd_sub1_pid _wd_sub2_pid	サブイベントのプロセス ID。
_wd_sub1_procname **_wd_sub2_procname**	サブイベントのプロセス名。
_wd_sub1_value _wd_sub2_value	パーセンテージで測定されたサブイベントの CPU 使用率。
Watchdog System Monitor（WDSysMon）mem_tot_avail and mem_tot_used サブイベント	
_wd_sub1_avail _wd_sub2_avail	サブイベントに使用可能なメモリ。
_wd_sub1_diff _wd_sub2_diff	イベントをトリガーした差のパーセンテージの値。 （注） この変数は、**_wd_sub1_is_percent** 変数または **_wd_sub2_is_percent** 変数が 1 である場合に限って設定されます。

環境変数	説明
_wd_sub1_is_percent **_wd_sub2_is_percent**	値がパーセンテージであるかどうかを識別する番号。0 であれば値がパーセンテージではないことを意味し、1 であれば値がパーセンテージであることを意味します。
_wd_sub1_node　_wd_sub2_node	サブイベント RP レポーティング ノードのスロット番号。
_wd_sub1_period　_wd_sub2_period	サブイベントの測定に使用される時間間隔（秒単位、オプションでミリ秒単位）。
_wd_sub1_value　_wd_sub2_value	パーセンテージで測定されたサブイベントの CPU 使用率。
_wd_sub1_used　_wd_sub2_used	サブイベントが使用したメモリ。

Cisco IOS CLI を使用した EEM ポリシーの記述方法

Embedded Event Manager アプレットの登録と定義

アプレットを Embedded Event Manager に登録し、Cisco IOS CLI **event** コマンドと **action** コマンドを使用して定義するには、次の作業を実行します。EEM アプレットでは、**event** コマンドが 1 つだけ許可されます。**action** コマンドは複数許可されます。**event** コマンドと **action** コマンドが指定されていない場合、コンフィギュレーション モードの終了時にアプレットが削除されます。

この作業で使用する SNMP イベントディテクタと syslog **action** コマンドは、任意のイベントディテクタと **action** コマンドを表しています。他のイベントディテクタや **action** コマンドの使用例については、Embedded Event Manager アプレットの設定例 （63 ページ） を参照してください。

EEM 環境変数

EEM ポリシーの EEM 環境変数は、EEM **event manager environment** コンフィギュレーション コマンドを使用して定義されます。慣例として、すべてのシスコ EEM 環境変数は、「_」で始まります。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。

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

たとえば、イベント発生時に E メールを送信する EEM ポリシーを作成できます。次の表に、EEM ポリシーで使用できる電子メール特有の環境変数の説明を示します。

表 *2 : EEM* 電子メール固有の環境変数

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 サーバー。	電子メール サーバー名（Mailservername）は、次のテンプレート形式のいずれかを使用できます。 • 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

EEM アクション ラベルのアルファベット順

EEM アクション ラベルは一意の ID で、任意の文字列値が可能です。アクションは、ラベルをソート キーとして使用して、英数字のキーの昇順（辞書順）にソートされ、実行されます。ラベルとして数字を使用している場合は、英数字ソートは、10.0 は 1.0 よりも後ですが、2.0 よりも前になることに注意してください。このような場合、01.0、02.0 のような数字を使用する、または頭文字の後に同様の数字を続けることを推奨します。

手順の概要

1. **enable**
2. **show event manager environment** [**all**| *variable-name*]
3. **configure terminal**
4. **event manager environment** *variable-name string*
5. EEM アクション ラベルのアルファベット順 を、必要なすべての環境変数に繰り返します。
6. **event manager applet** *applet-name*
7. 次のいずれかを実行します。
   - **event snmp oid** *oid-value* **get-type** {**exact**| **next**} **entry-op** *operator* **entry-val** *entry-value*[**exit-comb**| **and**}] [**exit-op** *operator*] [**exit-val** *exit-value*] [**exit-time** *exit-time-value*] **poll-interval** *poll-int-value*
8. **action** *label* **cli command** *cli-string* [**pattern** *pattern-string*]
9. **action** *label* **syslog** [**priority** *priority-level*] **msg** *msg-text* **facility** *string*
10. **action** *label* **mail server** *server-address* **to** *to-address* **from** *from-address* [**cc** *cc-address*] **subject** *subject* **body** *body-text*

11. 必要に応じて action コマンドを追加します。
12. **end**

手順の詳細

	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 _email_to engineering@example.com`	指定された EEM 環境変数の値を設定します。 • この例では、E メール送信先の E メール アドレスを保持する環境変数は、engineering@example.com に設定されます。
ステップ 5	EEM アクション ラベルのアルファベット順 を、必要なすべての環境変数に繰り返します。	EEM アクション ラベルのアルファベット順 を繰り返して、EEM アクション ラベルのアルファベット順 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 applet** *applet-name* 例： `Device(config)# event manager applet memory-fail`	Embedded Event Manager（EEM）にアプレットを登録し、アプレット コンフィギュレーション モードを開始します。
ステップ 7	次のいずれかを実行します。 • **event snmp oid** *oid-value* **get-type** {**exact**| **next**} **entry-op** *operator* **entry-val** *entry-value*[**exit-comb**| **and**}] [**exit-op** *operator*] [**exit-val** *exit-value*] [**exit-time** *exit-time-value*] **poll-interval** *poll-int-value*	EEM アプレットの実行の原因となる、イベント基準を指定します。 • この例では、空きメモリの値が 5120000 を下回ったときに EEM イベントがトリガーされます。

	コマンドまたはアクション	目的
	例： `Device(config-applet)# event snmp oid 1.3.6.1.4.1.9.9.48.1.1.1.6.1 get-type exact entry-op lt entry-val 5120000 poll-interval 90`	• 終了基準はオプションです。指定されない場合、イベントのモニターリングは、すぐに再び有効になります。

ステップ 8	**action** *label* **cli command** *cli-string* [**pattern** *pattern-string*]
	例：
	`Device(config-applet)# action 1.0 cli command "enable"`
	例：
	`Device(config-applet)# action 2.0 cli command "clear counters Ethernet0/1" pattern "confirm"`
	例：
	`Device(config-applet)# action 3.0 cli command "y"`
		EEM アプレットがトリガーされたときに Cisco IOS CLI コマンドを実行するアクションを指定します。
		pattern キーワードはオプションで、コマンド文字列が入力を求める場合にだけ使用します。**action cli** コマンドは、オプションの **pattern** キーワードで指定されているとおりの応答プロンプトを受信した時点で終了します。次の応答プロンプトに一致する正規表現パターンを指定する必要があります。正しくないパターンを指定すると、**action cli** コマンドが、maxrun タイマー期限切れによるアプレット実行タイムアウトまで、待ち続けることになります。
		• 実行されるアクションは、**pattern** キーワードが **clear counters Ethernet0/1** コマンドの *confirm* 引数を指定するときに実行される EEM アプレットを指定するためのものです。この場合、コマンド文字列は「confirm」という入力を要求します。その入力は、「yes」または「no」で完了する必要があります。

ステップ 9	**action** *label* **syslog** [**priority** *priority-level*] **msg** *msg-text* **facility** *string*
	例：
	`Device(config-applet)# action 1.0 syslog priority critical msg "Memory exhausted; current available memory is $_snmp_oid_val bytes"`
	例：
	`Device(config-applet)# action 1.0 syslog priority errors facility EEM-FAC message "TEST MSG"`
		EEM アプレットがトリガーされたときに実行されるアクションを指定します。
		この例では、実行されるアクションは syslog にメッセージを書き込むことです。
		• オプションの **priority** キーワードは syslog メッセージの優先度レベルを指定します。選択した場合は、*priority-level* 引数を定義する必要があります。
		• *msg-text* 引数は、文字テキスト、環境変数、またはその両方の組み合わせが可能です。
		• **facility** キーワードは生成したメッセージの場所を指定します。
		• *string* 引数は、キャラクタ テキスト、環境変数、またはその両方の組み合わせが可能です。

	コマンドまたはアクション	目的
ステップ 10	**action** *label* **mail server** *server-address* **to** *to-address* **from** *from-address* [**cc** *cc-address*] **subject** *subject* **body** *body-text* 例： `Device(config-applet)# action 2.0 mail server 192.168.1.10 to engineering@example.com from devtest@example.com subject "Memory failure" body "Memory exhausted; current available memory is $_snmp_oid_val bytes"`	EEM アプレットがトリガーされたときにショートメールを送信するアクションを指定します。 • *server-address* 引数は、電子メールの転送に使用する電子メール サーバーの完全修飾ドメイン名を指定します。 • *to-address* 引数は、電子メールの送信先の電子メール アドレスを指定します。 • *from-address* 引数は、電子メール送信元の電子メール アドレスを指定します。 • *subject* 引数は、英数字の文字列で、電子メールのサブジェクトラインの内容を指定します。 • *body-text* 引数は、英数字の文字列で、電子メールのテキストの内容を指定します。
ステップ 11	必要に応じて action コマンドを追加します。	--
ステップ 12	**end** 例： `Device(config-applet)# end`	アプレット コンフィギュレーション モードを終了し、特権 EXEC モードに戻ります。

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。debugging 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 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。

手順の概要

1. **enable**
2. **show event manager environment** [**all**| *variable-name*]
3. **configure terminal**
4. **event manager environment** *variable-name string*
5. EEM Tcl スクリプトの登録と定義 を繰り返して、EEM Tcl スクリプトの登録と定義 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。
6. **event manager policy** *policy-filename* [**type** {**system**| **user**}] [**trap**]
7. **exit**

手順の詳細

	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	EEM Tcl スクリプトの登録と定義 を繰り返して、EEM Tcl スクリプトの登録と定義 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： `Device(config)# event manager policy tm_cli_cmd.tcl type system`	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。 • **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。

	コマンドまたはアクション	目的
		• **user** キーワードを使用して、ユーザー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
```

Embedded Event Manager 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。

手順の概要

1. **enable**
2. **show event manager policy registered** [**description** [*policy-name*] | **detailed** *policy-filename* [**system** | **user**] | [**event-type** *event-name*] [**system** | **user**] [**time-ordered** | **name-ordered**]]
3. **configure terminal**
4. **no event manager policy** *policy-filename*
5. **exit**
6. ステップ 2 を繰り返して、ポリシーが削除されたことを確認します。

手順の詳細

	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** [**description** [*policy-name*] | **detailed** *policy-filename* [**system** | **user**] | [**event-type** *event-name*] [**system** | **user**] [**time-ordered** | **name-ordered**]] 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** キーワードおよび **user** キーワードは登録されているシステム ポリシーおよびユーザー ポリシーを表示します。 • 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 IPSLAping1`	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。
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	ステップ 2 を繰り返して、ポリシーが削除されたことを確認します。 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 2 個の EEM アプレットを表示する例を示します。

```
Device# show event manager policy registered
No.  Class   Type    Event Type          Trap  Time Registered          Name
```

```
1    applet  system  snmp                Off   Fri Aug 12 17:42:52 2005  IPSLAping1
 oid {1.3.6.1.4.1.9.9.42.1.2.9.1.6.4} get-type exact entry-op eq entry-val {1}
 exit-op eq exit-val {2} poll-interval 90.000
 action 1.0 syslog priority critical msg "Server IPecho Failed: OID=$_snmp_oid_val"
 action 1.1 snmp-trap strdata "EEM detected server reachability failure to 10.1.88.9"
 action 1.2 publish-event sub-system 88000101 type 1 arg1 "10.1.88.9" arg2 "IPSLAEcho"
arg3 "fail"
 action 1.3 counter name _IPSLA1F op inc value 1
2    applet  system  snmp                Off   Thu Sep 15 05:57:16 2005  memory-fail
 oid {1.3.6.1.4.1.9.9.48.1.1.1.6.1} get-type exact entry-op lt entry-val {5120000}
poll-interval 90
 action 1.0 syslog priority critical msg Memory exhausted; current available memory is
$_snmp_oid_val bytes
 action 2.0 force-switchover
```

次の例では、**show event manager policy registered** 特権 EXEC コマンドを使用して、アプレット IPSLAping1 が **no event manager policy** コマンドの入力後に削除されていることを示します。

```
Device# show event manager policy registered
No.  Class   Type    Event Type          Trap  Time Registered           Name
1    applet  system  snmp                Off   Thu Sep 15 05:57:16 2005  memory-fail
 oid {1.3.6.1.4.1.9.9.48.1.1.1.6.1} get-type exact entry-op lt entry-val {5120000}
poll-interval 90
 action 1.0 syslog priority critical msg Memory exhausted; current available memory is
$_snmp_oid_val bytes
 action 2.0 force-switchover
```

すべての Embedded Event Manager ポリシーの実行の一時停止

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。

手順の概要

1. **enable**
2. **show event manager policy registered** [**description** [*policy-name*] | **detailed** *policy-filename* [**system** | **user**] | [**event-type** *event-name*] [**system** | **user**] [**time-ordered** | **name-ordered**]]
3. **configure terminal**
4. **event manager scheduler suspend**
5. **exit**

手順の詳細

	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）。

	コマンドまたはアクション	目的
ステップ 2	**show event manager policy registered** [**description** [*policy-name*] | **detailed** *policy-filename* [**system** | **user**] | [**event-type** *event-name*] [**system** | **user**] [**time-ordered** | **name-ordered**]] 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** キーワードおよび **user** キーワードは登録されているシステム ポリシーおよびユーザー ポリシーを表示します。 • 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。

Embedded Event Manager 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。

手順の概要

1. **enable**
2. **configure terminal**
3. **event manager history size** {**events** | **traps**} [*size*]
4. **exit**
5. **show event manager history events** [**detailed**] [**maximum** *number*]
6. **show event manager history traps** {**server** | **policy**}

手順の詳細

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、各 EEM イベントの詳細情報を表示します。次に例を示します。

例：

```
Device# show event manager history events
No.  Time of Event             Event Type          Name
1    Fri Aug13  21:42:57 2004  snmp                applet: SAAping1
2    Fri Aug13  22:20:29 2004  snmp                applet: SAAping1
3    Wed Aug18  21:54:48 2004  snmp                applet: SAAping1
4    Wed Aug18  22:06:38 2004  snmp                applet: SAAping1
5    Wed Aug18  22:30:58 2004  snmp                applet: SAAping1
6    Wed Aug18  22:34:58 2004  snmp                applet: SAAping1
7    Wed Aug18  22:51:18 2004  snmp                applet: SAAping1
8    Wed Aug18  22:51:18 2004  application         applet: CustApp1
```

ステップ 6 show event manager history traps {**server** | **policy**}

このコマンドを使用して、EEM サーバーまたは 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。次に、EEM ポリシー内からトリガーされた EEM SNMP トラップが表示される例を示します。

例：

```
Device# show event manager history traps policy
```

```
No.  Time                      Trap Type           Name
1    Wed Aug18  22:30:58 2004  policy              EEM Policy Director
2    Wed Aug18  22:34:58 2004  policy              EEM Policy Director
3    Wed Aug18  22:51:18 2004  policy              EEM Policy Director
```

Embedded Event Manager 登録済みポリシーの表示

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。

手順の概要

1. **enable**
2. **show event manager policy registered** [**event-type** *event-name*] [**time-ordered**| **name-ordered**]

手順の詳細

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]

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          Time                    Registered Name
1    applet  snmp                Thu May30 05:57:16 2004 memory-fail
 oid {1.3.6.1.4.1.9.9.48.1.1.1.6.1} get-type exact entry-op lt entry-val
{5120000} poll-interval 90
action 1.0 syslog priority critical msg "Memory exhausted; current available memory
is $_snmp_oid_val bytes"
 action 2.0 force-switchover
2    applet  syslog              Wed Jul16 00:05:17 2004 intf-down
 pattern {.*UPDOWN.*Ethernet1/0.*}
 action 1.0 cns-event msg "Interface state change: $_syslog_msg"
```

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          Time Registered          Name
1    applet  syslog              Wed Jul16  00:05:17 2004 intf-down
 pattern {.*UPDOWN.*Ethernet1/0.*}
 action 1.0 cns-event msg "Interface state change: $_syslog_msg"
```

```
2    applet  snmp                Thu May30 05:57:16 2004   memory-fail
 oid {1.3.6.1.4.1.9.9.48.1.1.1.6.1} get-type exact entry-op lt entry-val
{5120000} poll-interval 90
 action 1.0 syslog priority critical msg "Memory exhausted; current available memory
is $_snmp_oid_val bytes"
 action 2.0 force-switchover
```

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    applet  syslog              Wed Jul16  00:05:17 2004 intf-down
 pattern {.*UPDOWN.*Ethernet1/0.*}
 action 1.0 cns-event msg "Interface state change: $_syslog_msg"
```

イベント SNMP 通知の設定

SNMP 通知を設定するには、次の作業を実行します。

始める前に

- SNMP イベントマネージャは、**snmp-server manager** コマンドを使用して設定する必要があります。

手順の概要

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **event** [**tag** *event-tag*] **snmp-notification oid** *oid-string* **oid-val** *comparison-value* **op** *operator* [**maxrun** *maxruntime-number*] [**src-ip-address** *ip-address*] [**dest-ip-address** *ip-address*] [**default** *seconds*] [**direction** {**incoming** | **outgoing**}] [**msg-op** {**drop** | **send**}]
5. **end**

手順の詳細

	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	**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 3	**event manager applet** *applet-name* 例： `Device(config)# event manager applet snmp`	Event Manager にアプレットを登録し、アプレット コンフィギュレーション モードを開始します。
ステップ 4	**event** [**tag** *event-tag*] **snmp-notification oid** *oid-string* **oid-val** *comparison-value* **op** *operator* [**maxrun** *maxruntime-number*] [**src-ip-address** *ip-address*] [**dest-ip-address** *ip-address*] [**default** *seconds*] [**direction** {**incoming** | **outgoing**}] [**msg-op** {**drop** | **send**}] 例： `Device(config-applet)# event snmp-notification dest-ip-address 192.168.1.1 oid 1 op eq oid-val 10`	簡易ネットワーク管理プロトコル（SNMP）通知のサンプリングによって実行される Embedded Event Manager（EEM）アプレットのイベント基準を指定します。
ステップ 5	**end** 例： `Device(config-applet)# end`	アプレット コンフィギュレーション モードを終了し、特権 EXEC モードに戻ります。

複数イベント サポートの設定

複数イベントサポート機能は、EEMサーバーに複数のイベントを登録する機能を追加します。複数イベントサポートには、1個以上のイベントの発生、1個以上のトラッキング対象オブジェクトの状態、および、発生するイベントの時間間隔が含まれます。イベント パラメータは、CLIコマンドで指定されます。複数イベントを扱うためのデータ構造には、複数のイベントIDと相関関係ロジックが含まれます。このデータは、EEM サーバーに複数のイベントを登録するために使用されます。

イベント設定パラメータの設定

trigger コマンドは、トリガー アプレット コンフィギュレーション モードを開始し、EEM アプレットの複数イベント設定ステートメントを指定します。トリガー ステートメントは、各イベント文に指定される *tag* 引数を使用して複数イベント ステートメントを関連付けます。イベントは指定されたパラメータに基づいて発生します。

手順の概要

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*

4. **event** [**tag** *event-tag*] **cli pattern** *regular-expression* **sync** {**yes** | **no skip** {**yes** | **no**}} [**occurs** *num-occurrences*] [**period** *period-value*] [**maxrun** *maxruntime-number*]
5. **trigger** [**occurs** *occurs-value*] [**period** *period-value*] [**period-start** *period-start-value*] [**delay** *delay-value*]
6. **correlate** {**event** *event-tag* | **track** *object-number*} [*boolean-operator* **event** *event-tag*]
7. **attribute tag** *event-tag* [**occurs** *occurs-value*]
8. **action** *label* **cli command** *cli-string*

手順の詳細

	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	**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 3	**event manager applet** *applet-name* 例： `Device(config)# event manager applet EventInterface`	EEM にアプレットを登録し、アプレット コンフィギュレーション モードを開始します。
ステップ 4	**event** [**tag** *event-tag*] **cli pattern** *regular-expression* **sync** {**yes**	**no skip** {**yes**	**no**}} [**occurs** *num-occurrences*] [**period** *period-value*] [**maxrun** *maxruntime-number*] 例： `Device(config-applet)# event tag 1.0 cli pattern "show bgp all" sync yes occurs 32 period 60 maxrun 60`	Cisco IOS コマンドライン インターフェイス（CLI）コマンドの一致によって実行される EEM アプレットのイベント基準を指定します。
ステップ 5	**trigger** [**occurs** *occurs-value*] [**period** *period-value*] [**period-start** *period-start-value*] [**delay** *delay-value*] 例： `Device(config-applet)# trigger occurs 1 period-start "0 8 * * 1-5" period 60`	EEM アプレットの複雑なイベント設定パラメータを指定します。
ステップ 6	**correlate** {**event** *event-tag*	**track** *object-number*} [*boolean-operator* **event** *event-tag*] 例：	EEM アプレットのトリガー モードで複雑なイベント関連付けを指定します。

	コマンドまたはアクション	目的
	`Device(config-applet)# correlate event 1.0 or event 2.0`	（注） 「and」を使用して、トラップやsyslogメッセージなどのイベントをグループ化した場合、デフォルトのトリガー発生時間枠は3分です。
ステップ 7	**attribute tag** *event-tag* [**occurs** *occurs-value*] 例： `Device(config-applet)# attribute tag 1.0 occurs 1`	EEMアプレットの複雑なイベントをビルドする最大8個の属性文を指定します。
ステップ 8	**action** *label* **cli command** *cli-string* 例： `Device(config-applet)# action 1.0 cli command "show pattern"`	EEM アプレットがトリガーされたときに CLI コマンドを実行するアクションを指定します。

例

次に、**show bgp all** CLI コマンドと「COUNT」文字列を含む syslog メッセージが 60 秒以内に発生した場合にアプレットが実行される例を示します。

```
event manager applet delay_50
 event tag 1.0 cli pattern "show bgp all" sync yes occurs 32 period 60 maxrun 60
 event tag 2.0 syslog pattern "COUNT"
 trigger occurs 1 delay 50
  correlate event 1.0 or event 2.0
  attribute tag 1.0 occurs 1
  attribute tag 2.0 occurs 1
 action 1.0 cli command "show pattern"
 action 2.0 cli command "enable"
 action 3.0 cli command "config terminal"
 action 4.0 cli command " ip route 192.0.2.0 255.255.255.224 192.0.2.12"
 action 91.0 cli command "exit"
 action 99.0 cli command "show ip route | incl 192.0.2.5"
```

EEM クラスベース スケジューリングの設定

Embedded Event Manager（EEM）ポリシーをスケジュールし、ポリシー スケジュール オプションを設定するには、次の作業を実行します。このタスクでは、2 個の EEM 実行スレッドが作成され、デフォルト クラスに割り当てられたアプレットが実行されます。

EEM ポリシーは、登録時に **class** キーワードを使用して、クラスに割り当てられます。クラスなしで登録された EEM ポリシーは、デフォルト クラスに割り当てられます。デフォルト クラスを保持するスレッドは、スレッドが作業に利用可能であるとき、デフォルト クラスをサービスします。特定のクラス文字に割り当てられたスレッドは、スレッドが作業に利用可能であるとき、クラス文字が一致する任意のポリシーをサービスします。

EEM 実行スレッドが、指定されたクラスのポリシー実行に利用可能でない場合で、クラスのスケジューラルールが設定されている場合は、ポリシーは該当クラスのスレッドが実行可能になるまで待ちます。同じ入力イベントからトリガーされた同期ポリシーは、同一の実行スレッドにスケジュールされなければなりません。

手順の概要

1. **enable**
2. **configure terminal**
3. {| |} クラスオプションスレッド **event manager scheduler appletaxpcall-homethread class** *class-options* **number** 番号
4. **exit**

手順の詳細

	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	**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 3	{| |} クラスオプションスレッド **event manager scheduler appletaxpcall-homethread class** *class-options* **number** 番号 例： `Device(config)# event manager scheduler applet thread class default number 2`	EEM ポリシーをスケジュールし、ポリシー スケジューリング オプションを設定します。 • この例では、2 個の EEM 実行スレッドが作成され、デフォルト クラスに割り当てられたアプレットが実行されます。
ステップ 4	**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。

スケジュール済み EEM ポリシー イベントまたはイベント キューの保留

EEM スケジューラで、スケジュールされた EEM ポリシー イベントまたはイベント キューをホールドするには、次の作業を実行します。このタスクでは、すべての保留 EEM ポリシーが表示されます。ジョブ ID 2 を使用して特定されるポリシーは、EEM スケジューラでホールドされています。最初のステップは、ジョブ ID 2 のポリシーは、状態が Pending から Held に変更されていることを示しています。

手順の概要

1. **enable**
2. **show event manager policy pending** [**queue-type**{**applet** | **call-home** | **axp** | **script**} **class** *class-options* | **detailed**]
3. **event manager scheduler hold** {**all**| **policy** *job-id* | **queue-type** {**applet** | **call-home** | **axp** | **script**} **class** *class-options*} [**processor** {**rp_primary**| **rp_standby**}]
4. **show event manager policy pending** [**queue-type**{**applet** | **call-home** | **axp** | **script**} **class** *class-options* | **detailed**

手順の詳細

	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 pending** [**queue-type**{**applet**	**call-home**	**axp**	**script**} **class** *class-options*	**detailed**] 例： `Device# show event manager policy pending`	保留 EEM ポリシーを表示します。
ステップ 3	**event manager scheduler hold** {**all**| **policy** *job-id*	**queue-type** {**applet**	**call-home**	**axp**	**script**} **class** *class-options*} [**processor** {**rp_primary**| **rp_standby**}] 例： `Device# event manager scheduler hold policy 2`	EEM スケジューラで、スケジュールされた EEM ポリシー イベントまたはイベント キューをホールドします。 • この例では、ジョブ ID 2 のポリシーがホールドされます。
ステップ 4	**show event manager policy pending** [**queue-type**{**applet**	**call-home**	**axp**	**script**} **class** *class-options*	**detailed** 例： `Device# show event manager policy pending`	他の保留ポリシーとともに、手順 3 でホールドされた EEM ポリシーのステータスが Held と表示されます。

例

次に、すべての保留 EEM ポリシーの表示方法とジョブ ID 2 の EEM ポリシーをホールドする例を示します。

```
Device# show event manager policy pending
no. job id status time of event             event type     name
1   1      pend   Thu Sep 7  02:54:04 2006  syslog         applet: one
2   2      pend   Thu Sep 7  02:54:04 2006  syslog         applet: two
3   3      pend   Thu Sep 7  02:54:04 2006  syslog         applet: three
Device# event manager scheduler hold policy 2
```

```
Device# show event manager policy pending

no. job id status time of event             event type     name
1   1      pend   Thu Sep 7  02:54:04 2006  syslog         applet: one
2   2      held   Thu Sep 7  02:54:04 2006  syslog         applet: two
3   3      pend   Thu Sep 7  02:54:04 2006  syslog         applet: three
```

EEM ポリシー イベントまたはイベント キューの実行の再開

EEM ポリシー イベントまたはイベント キューの実行を再開するには、次の作業を実行します。このタスクでは、スケジュール済み EEM ポリシー イベントまたはイベント キューの保留で保留状態となっていたポリシーは、実行を再開できるようになっています。

手順の概要

1. **enable**
2. **show event manager policy pending**
3. **event manager scheduler release** {**all** | **policy** *policy-id* | **queue-type** {**applet** | **call-home** | **axp** | **script**}} **class** *class-options* [**processor** {*rp_primary* | *rp_standby*}]
4. **show event manager policy pending**

手順の詳細

	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 pending** 例： `Device# show event manager policy pending`	保留およびホールドされた EEM ポリシーを表示します。 （注） この例では、このタスクに適用可能な構文だけが使用されています。詳細については、『Cisco IOS Network Management Command Reference』を参照してください。
ステップ 3	**event manager scheduler release** {**all** | **policy** *policy-id* | **queue-type** {**applet** | **call-home** | **axp** | **script**}} **class** *class-options* [**processor** {*rp_primary* | *rp_standby*}] 例： `Device# event manager scheduler release policy 2`	指定された EEM ポリシーの実行を再開します。 • 例では、ジョブ ID 2 のポリシーの実行を再開する方法を示しています。
ステップ 4	**show event manager policy pending** 例：	他の保留ポリシーとともに、手順 3 で再開された EEM ポリシーの状態が pending と表示されます。

	コマンドまたはアクション	目的
	`Device# show event manager policy pending`	（注） この例では、このタスクに適用可能な構文だけが使用されています。詳細については、『Cisco IOS Network Management Command Reference』を参照してください。

例

次に、すべての保留EEMポリシーの表示方法、および実行を再開するポリシーを指定する方法、ポリシーが保留状態に戻っていることを確認する例を示します。

```
Device# show event manager policy pending

no. job id status time of event             event type     name
1   1      pend   Thu Sep 7  02:54:04 2006  syslog         applet: one
2  2      held   Thu Sep 7  02:54:04 2006  syslog         applet: two
3   3      pend   Thu Sep 7  02:54:04 2006  syslog         applet: three
Rotuer# event manager scheduler release policy 2
Rotuer# show event manager policy pending
no. job id status time of event             event type     name
1   1      pend   Thu Sep 7  02:54:04 2006  syslog         applet: one
2   2      pend   Thu Sep 7  02:54:04 2006  syslog         applet: two
3   3      pend   Thu Sep 7  02:54:04 2006  syslog         applet: three
```

保留 EEM ポリシー イベントまたはイベント キューのクリア

実行中または実行を保留中の EEM ポリシー イベントをクリアするには、次の作業を実行します。このタスクでは、ジョブ ID 2 のポリシーが保留キューからクリアされます。ポリシーがクリアされる前後に保留中のポリシーを表示するには、**show event manager policy pending** コマンドを使用します。

手順の概要

1. **enable**
2. **show event manager policy pending**
3. **event manager scheduler clear** {**all** | **policy** *job-id* | **queue-type** {**applet** | **call-home** | **axp** | **script**} **class** *class-options*} [**processor** {**rp_primary** | **rp_standby**}]
4. **show event manager policy pending**

手順の詳細

	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）。

	コマンドまたはアクション	目的
ステップ 2	**show event manager policy pending** 例： `Device# show event manager policy pending`	保留 EEM ポリシーを表示します。 （注） この例では、このタスクに適用可能な構文だけが使用されています。詳細については、『Cisco IOS Network Management Command Reference』を参照してください。
ステップ 3	**event manager scheduler clear** {**all** | **policy** *job-id* | **queue-type** {**applet** | **call-home** | **axp** | **script**} **class** *class-options*} [**processor** {**rp_primary** | **rp_standby**}] 例： `Device# event manager scheduler clear policy 2`	実行中または実行を保留中の EEM ポリシーをクリアします。 • この例では、ジョブ ID 2 のポリシーが保留キューからクリアされます。
ステップ 4	**show event manager policy pending** 例： `Device# show event manager policy pending`	手順 3 でクリアされたポリシーを除く、保留中のすべての EEM ポリシーを表示します。 （注） この例では、このタスクに適用可能な構文だけが使用されています。詳細については、『Cisco IOS Network Management Command Reference』を参照してください。

例

次に、実行を保留されたジョブ ID 2 のポリシーをクリアする例を示します。ポリシーがクリアされる前後に保留中のポリシーを表示するには、**show** コマンドを使用します。

```
Device# show event manager policy pending
no. job id status time of event             event type     name
1   1      pend   Thu Sep 7  02:54:04 2006  syslog         applet: one
2   2      pend   Thu Sep 7  02:54:04 2006  syslog         applet: two
3   3      pend   Thu Sep 7  02:54:04 2006  syslog         applet: three

Device# event manager scheduler clear policy 2
Device# show event manager policy pending

no. job id status time of event             event type     name
1   1      pend   Thu Sep 7  02:54:04 2006  syslog         applet: one
3   3      pend   Thu Sep 7  02:54:04 2006  syslog         applet: three
```

EEM ポリシー イベントまたはイベント キューのスケジューリング パラメータの変更

EEM ポリシー イベントのスケジューリング パラメータを変更するには、次の作業を実行します。**show event manager policy pending** コマンドは、B またはデフォルトクラスに割り当てられているポリシーを表示します。現在保留されているすべてのポリシーがクラス A に変更され

ます。設定変更後、**show event manager policy pending** コマンドはクラス A として割り当てられているすべてのポリシーを表示します。

手順の概要

1. **enable**
2. **show event manager policy pending**
3. **event manager scheduler modify** {**all** | **policy** *job-id* | **queue-type** {**applet** | **call-home** | **axp** | **script**} | **class** *class-options*} [**queue-priority** {**high** | **last** | **low** | **normal**}][**processor** {**rp_primary** | **rp_standby**}]
4. **show event manager policy pending**

手順の詳細

	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 pending** 例： `Device# show event manager policy pending`	保留 EEM ポリシーを表示します。 （注） この例では、このタスクに適用可能な構文だけが使用されています。詳細については、『Cisco IOS Network Management Command Reference』を参照してください。
ステップ 3	**event manager scheduler modify** {**all** | **policy** *job-id* | **queue-type** {**applet** | **call-home** | **axp** | **script**} | **class** *class-options*} [**queue-priority** {**high** | **last** | **low** | **normal**}][**processor** {**rp_primary** | **rp_standby**}] 例： `Device# event manager scheduler modify all class A`	EEM ポリシーのスケジューリング パラメータを変更します。 • この例では、現時点での保留 EEM ポリシーはすべてクラス A に割り当てられています。
ステップ 4	**show event manager policy pending** 例： `Device# show event manager policy pending`	他の保留ポリシーとともに、手順 3 で変更された EEM ポリシーが表示されます。 （注） この例では、このタスクに適用可能な構文だけが使用されています。詳細については、『Cisco IOS Network Management Command Reference』を参照してください。

例

次に、EEMポリシーのスケジューリングパラメータを変更する例を示します。この例では、**show event manager policy pending** コマンドは、B またはデフォルトクラスに割り当てられているポリシーを表示します。現在保留されているすべてのポリシーがクラス A に変更されます。設定変更後、**show event manager policy pending** コマンドはクラス A として現在割り当てられているすべてのポリシーを確認します。

```
Device# show event manager policy pending
no. class   status time of event             event type     name
1   default pend   Thu Sep 7  02:54:04 2006  syslog         applet: one
2   default pend   Thu Sep 7  02:54:04 2006  syslog         applet: two
3   B       pend   Thu Sep 7  02:54:04 2006  syslog         applet: three

Device# event manager scheduler modify all class A
Device# show event manager policy pending

no. class status time of event             event type     name
1   A     pend   Thu Sep 7  02:54:04 2006  syslog         applet: one
2   A     pend   Thu Sep 7  02:54:04 2006  syslog         applet: two
3   A     pend   Thu Sep 7  02:54:04 2006  syslog         applet: three
```

クラスベースのアクティブ EEM ポリシーの確認

アクティブな EEM ポリシーか、または実行中の EEM ポリシーを確認するには、**show event manager policy active** コマンドを使用します。

手順の概要

1. **show event manager policy active** [**queue-type** {**applet**| **call-home** | **axp** | **script**} **class** *class-options* | **detailed**]

手順の詳細

show event manager policy active [**queue-type** {**applet**| **call-home** | **axp** | **script**} **class** *class-options* | **detailed**]

このコマンドは、実行中の EEM ポリシーだけを表示します。このコマンドには、オプションの **class** キーワード、**detailed** キーワード、および **queue-type** キーワードが含まれています。次に、このコマンドの出力例を示します。

例：

```
Device# show event manager policy active
no. job id p s status time of event event type name
1 12598 N A running Mon Oct29 20:49:37 2007 timer watchdog loop.tcl
2 12609 N A running Mon Oct29 20:49:42 2007 timer watchdog loop.tcl
3 12620 N A running Mon Oct29 20:49:46 2007 timer watchdog loop.tcl
4 12650 N A running Mon Oct29 20:49:59 2007 timer watchdog loop.tcl
5 12842 N A running Mon Oct29 20:51:13 2007 timer watchdog loop.tcl
default class - 6 applet events
no. job id p s status time of event event type name
1 15852 N A running Mon Oct29 21:11:09 2007 counter WDOG_SYSLG_CNTR_TRACK_INTF_APPL
```

```
2 15853 N A running Mon Oct29 21:11:09 2007 counter WDOG_SYSLG_CNTR_TRACK_INTF_APPL
3 15854 N A running Mon Oct29 21:11:10 2007 counter WDOG_SYSLG_CNTR_TRACK_INTF_APPL
4 15855 N A running Mon Oct29 21:11:10 2007 timer watchdog WDOG_SYSLG_CNTR_TRACK_INTF_APPL
5 15856 N A running Mon Oct29 21:11:11 2007 counter WDOG_SYSLG_CNTR_TRACK_INTF_APPL
6 15858 N A running Mon Oct29 21:11:11 2007 counter WDOG_SYSLG_CNTR_TRACK_INTF_APPL
```

クラスベースのアクティブ EEM ポリシーの確認

アクティブな EEM ポリシーか、または実行中の EEM ポリシーを確認するには、**show event manager policy active** コマンドを使用します。

手順の概要

1. **show event manager policy active** [**queue-type** {**applet**| **call-home** | **axp** | **script**} **class** *class-options* | **detailed**]

手順の詳細

show event manager policy active [**queue-type** {**applet**| **call-home** | **axp** | **script**} **class** *class-options* | **detailed**]

このコマンドは、実行中の EEM ポリシーだけを表示します。このコマンドには、オプションの **class** キーワード、**detailed** キーワード、および **queue-type** キーワードが含まれています。次に、このコマンドの出力例を示します。

例：

```
Device# show event manager policy active
no. job id p s status time of event event type name
1 12598 N A running Mon Oct29 20:49:37 2007 timer watchdog loop.tcl
2 12609 N A running Mon Oct29 20:49:42 2007 timer watchdog loop.tcl
3 12620 N A running Mon Oct29 20:49:46 2007 timer watchdog loop.tcl
4 12650 N A running Mon Oct29 20:49:59 2007 timer watchdog loop.tcl
5 12842 N A running Mon Oct29 20:51:13 2007 timer watchdog loop.tcl
default class - 6 applet events
no. job id p s status time of event event type name
1 15852 N A running Mon Oct29 21:11:09 2007 counter WDOG_SYSLG_CNTR_TRACK_INTF_APPL
2 15853 N A running Mon Oct29 21:11:09 2007 counter WDOG_SYSLG_CNTR_TRACK_INTF_APPL
3 15854 N A running Mon Oct29 21:11:10 2007 counter WDOG_SYSLG_CNTR_TRACK_INTF_APPL
4 15855 N A running Mon Oct29 21:11:10 2007 timer watchdog WDOG_SYSLG_CNTR_TRACK_INTF_APPL
5 15856 N A running Mon Oct29 21:11:11 2007 counter WDOG_SYSLG_CNTR_TRACK_INTF_APPL
6 15858 N A running Mon Oct29 21:11:11 2007 counter WDOG_SYSLG_CNTR_TRACK_INTF_APPL
```

保留 EEM ポリシーの確認

実行が保留中の EEM ポリシーを確認するには、**show event manager policy pending** コマンドを使用します。EEM クラス ベースのスケジュール オプションを指定するには、オプションのキーワードを使用します。

手順の概要

1. **show event manager policy pending** [**queue-type** {**applet**| **call-home** | **axp** | **script**} **class** *class-options* | **detailed**]

手順の詳細

show event manager policy pending [**queue-type** {**applet**| **call-home** | **axp** | **script**} **class** *class-options* | **detailed**]

このコマンドは、保留中の EEM ポリシーのみを表示します。このコマンドには、オプションの **class** キーワード、**detailed** キーワード、および **queue-type** キーワードが含まれています。次に、このコマンドの出力例を示します。

例：

```
Device# show event manager policy pending
no. job id p s status time of event event type name
1    12851   N A    pend    Mon Oct29   20:51:18 2007   timer watchdog   loop.tcl
2    12868   N A    pend    Mon Oct29   20:51:24 2007   timer watchdog   loop.tcl
3    12873   N A    pend    Mon Oct29   20:51:27 2007   timer watchdog   loop.tcl
4    12907   N A    pend    Mon Oct29   20:51:41 2007   timer watchdog   loop.tcl
5    13100   N A    pend    Mon Oct29   20:52:55 2007   timer watchdog   loop.tcl
```

EEM アプレット（インタラクティブ CLI）サポートの設定

同期アプレットは、2 つのコマンド、**action gets** および **action puts** を使用してローカルコンソール（tty）との連携をサポートするように拡張されました。これらのコマンドによってコンソールへの直接入力と表示が可能です。同期アプレットの出力は、System Logger をバイパスします。ローカル コンソールは、アプレットによって開かれ、対応する同期イベント ディテクタ pty によってサービスされます。同期出力は、開かれたコンソールに向けられます。

同期 EEM アプレットのアクティブ コンソールからの入力の読み取りと書き込み

次のタスクを使用して、EEM アプレットのインタラクティブ CLI サポートを実装します。

アクティブなコンソールからの入力の読み取り

同期ポリシーがトリガーされたとき、関連するコンソールがパブリッシュ情報仕様に格納されます。ポリシー ディテクタは、この情報を event_reqinfo コール内で問い合わせ、**action gets** コマンドで使用するために与えられたコンソール情報を格納します。

action gets コマンドは、アクティブコンソールからの入力の 1 行を読み、入力を変数に格納します。後続の改行文字は戻されません。

手順の概要

1. **enable**
2. **configure terminal**

3. **event manager applet** *applet-name*
4. **event none**
5. **action** *label* **gets** *variable*
6. **action** *label* **syslog** [**priority** *priority-level*] **msg** *msg-text*
7. **exit**

手順の詳細

	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	**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 3	**event manager applet** *applet-name* 例： `Device(config)# event manager applet action`	EEM にアプレットを登録し、アプレット コンフィギュレーション モードを開始します。
ステップ 4	**event none** 例： `Device(config-applet)# event none`	EEM に登録して手動で起動される EEM ポリシーを指定します。
ステップ 5	**action** *label* **gets** *variable* 例： `Device(config-applet)# action label2 gets input`	EEM アプレットがトリガーされたときに、同期アプレットのローカル コンソールから入力を取得し、与えられた変数に値を格納します。
ステップ 6	**action** *label* **syslog** [**priority** *priority-level*] **msg** *msg-text* 例： `Device(config-applet)# action label3 syslog msg "Input entered was \"$input\""`	EEM アプレットがトリガーされたときに実行されるアクションを指定します。 • この例では、実行されるアクションは手順 5 で指定された変数の値を syslog に書き込むことです。
ステップ 7	**exit** 例： `Device(config-applet)# exit`	アプレット コンフィギュレーション モードを終了し、特権 EXEC モードに戻ります。

例

次に、同期アプレットのローカルttyから入力を取得して値を格納する例を示します。

```
Device(config)# event manager applet action
Device(config-applet)# event none
Device(config-applet)# action label2 gets input
Device(config-applet)#  action label3 syslog msg "Input entered was \"$input\""
```

アクティブなコンソールへの入力の書き込み

同期ポリシーがトリガーされたとき、関連するコンソールがパブリッシュ情報仕様に格納されます。ポリシーディテクタは、この情報をevent_reqinfoコール内で問い合わせ、**action puts** コマンドで使用するために与えられたコンソール情報を格納します。

action puts コマンドは、アクティブコンソールに文字列を書き込みます。**nonewline** キーワードが指定されない限り、改行文字が表示されます。同期アプレットの **action puts** コマンドからの出力は、直接コンソールに表示され、System Logger をバイパスします。非同期アプレットの **action puts** コマンドの出力は、System Logger に向けられます。

手順の概要

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **event none**
5. **action** *label* **regexp** *string-pattern string-input* [*string-match* [*string-submatch1*] [*string-submatch2*] [*string-submatch3*]]
6. **action** *label* **puts** [**nonewline**] *string*
7. **exit**
8. **event manager run** *applet-name*

手順の詳細

	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	**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 3	**event manager applet** *applet-name* 例：	EEM にアプレットを登録し、アプレット コンフィギュレーション モードを開始します。

	コマンドまたはアクション	目的
	`Device(config)# event manager applet action`	
ステップ 4	**event none** 例： `Device(config-applet)# event none`	EEM に登録して手動で起動される EEM ポリシーを指定します。
ステップ 5	**action** *label* **regexp** *string-pattern string-input* [*string-match* [*string-submatch1*] [*string-submatch2*] [*string-submatch3*]] 例： `Device(config-applet)# action 1 regexp "(.*) (.*) (.*)" "one two three" _match _sub1`	EEM アプレットがトリガーされたときに入力文字列の正規表現パターンと比較するアクションを指定します。
ステップ 6	**action** *label* **puts** [**nonewline**] *string* 例： `Device(config-applet)# action 2 puts "match is $_match"`	EEM アプレットがトリガーされたときにデータを直接ローカルコンソールに出力するアクションを指定します。 • **nonewline** キーワードはオプションであり、改行文字を表示しないために使用します。
ステップ 7	**exit** 例： `Device(config-applet)# exit`	アプレット コンフィギュレーション モードを終了し、特権 EXEC モードに戻ります。
ステップ 8	**event manager run** *applet-name* 例： `Device# event manager run action`	登録された EEM ポリシーを手動で実行します。 • この例では、手順 3 で登録されたポリシーがトリガーされ、手順 5 および手順 6 で指定された、関連付けられたアクションが実行されます。

例

次に、**action puts** コマンドがデータを直接ローカルコンソールに出力する例を示します。

```
Device(config-applet)# event manager applet puts
Device(config-applet)# event none
Device(config-applet)# action 1 regexp "(.*) (.*) (.*)" "one two three" _match _sub1
Device(config-applet)# action 2 puts "match is $_match"
Device(config-applet)# action 3 puts "submatch 1 is $_sub1"
Device# event manager run puts
match is one two three
submatch 1 is one
```

SNMP ライブラリ拡張の設定

リリースに応じて、SNMP ライブラリ拡張機能で次の設定を実行できます。

前提条件

この機能を使用するには、Cisco IOS Release12.4(22)T 以降のリリースを実行している必要があります

SNMP Get および Set オペレーション

SNMP ライブラリ拡張機能により、EEM アプレットの **action info** コマンドと Tcl の **sys_reqinfo_snmp** コマンドが拡張され、SNMP の get-one、get-next、getid および set-any オペレーションのための機能が追加されます。

SNMP Get オペレーション

SNMP イベント マネージャは SNMP get オペレーションを実行して、管理対象オブジェクトの 1 つ以上の変数を取得します。**action info type snmp oid get-type** コマンドと **action info type snmp getid** コマンドを使用すると、取得する変数とエージェントの IP アドレスを指定して SNMP get 要求を送信するように SNMP イベントマネージャを設定できます。

たとえば、OID の値が 1.3.6.1.2.1.1.1 である変数を取得する場合、変数値、1.3.6.1.2.1.1.1 を指定する必要があります。指定された値が一致しない場合、トラップが生成され、エラーメッセージが syslog 履歴に書き込まれます。

action info type snmp oid get-type コマンドは、実行する get オペレーションのタイプを指定します。正確な変数を取得するには、get オペレーションのタイプを **exact** に指定する必要があります。指定された OID 値の辞書順での後続値を取得するには、get オペレーションのタイプを **next** に設定する必要があります。

次の表に、SNMP get オペレーションから取得された値が保存される組み込み変数を示します。

表 *3 : action info type snmp oid* コマンドの組み込み変数

組み込み変数	説明
_info_snmp_oid	SNMP オブジェクト ID。
_info_snmp_value	割り当てられた SNMP データ エレメントの値文字列。

GetID の動作

action info type snmp getid コマンドは SNMP エンティティから次の変数を取得します。

- sysDescr.0
- sysObjectID.0
- sysUpTime.0
- sysContact.0

- sysName.0
- sysLocation.0

次の表に、SNMP getID オペレーションから取得された値が保存される組み込み変数を示します。

表 *4 : action info type snmp getid* コマンドの組み込み変数

組み込み変数	説明
_info_snmp_syslocation_oid	sysLocation 変数の OID 値。
_info_snmp_syslocation_value	sysLocation 変数の値文字列。
_info_snmp_sysdescr_oid	sysDescr 変数の OID 値。
_info_snmp_sysdescr_value	sysDescr 変数の値文字列。
_info_snmp_sysobjectid_oid	sysObjectID 変数の OID 値。
_info_snmp_sysobjectid_value	sysObjectID 変数の値文字列。
_info_snmp_sysuptime_oid	sysUptime 変数の OID 値。
_info_snmp_sysuptime_value	sysUptime 変数の値文字列。
_info_snmp_syscontact_oid	sysContact 変数の OID 値。
_info_snmp_syscontact_value	sysContact 変数の値文字列。

get オペレーション要求は、ローカル ホストとリモート ホストの両方に送信できます。

SNMP Set オペレーション

MIB ビューでは、すべての SNMP 変数にデフォルト値が割り当てられています。SNMP イベント マネージャは、set オペレーションによってこれらの MIB 変数の値を変更できます。set オペレーションは、読み取りと書き込みアクセスが許可されたシステムでだけ実行できます。

set オペレーションを実行するには、変数のタイプと変数に割り当てられる値を指定する必要があります。

次の表に、有効な OID タイプと各 OID タイプの値を示します。

表 *5 : set* オペレーションの *OID* タイプおよび値

OID タイプ	説明
counter32	最小値が 0 の 32 ビットの数値。最大値に到達すると、カウンタが 0 にリセットされます。0 ～ 4294967295 の範囲の整数値が有効です。

OID タイプ	説明
gauge	最小値が 0 の 32 ビットの数値。たとえば、gauge オブジェクト タイプを使用して、デバイス上のインターフェイスの速度を測定できます。0 ～ 4294967295 の範囲の整数値が有効です。
integer	管理対象オブジェクトのコンテキスト内の番号が付けられたタイプを指定する場合は、32 ビットの数字が使用されます。たとえば、デバイス インターフェイスの動作ステータスを 1 に設定した場合はアップ、2 に設定した場合はダウンを示します。0 ～ 4294967295 の範囲の整数値が有効です。
ipv4	IP バージョン 4 アドレス。ドット付き 10 進表記の IPv4 アドレスが有効です。
octet string	物理アドレスを表すために使用される、16 進表記のオクテット文字列。テキスト文字列が有効です。
string	テキスト文字列を表すために使用される、テキスト表記のオクテット文字列。テキスト文字列が有効です。
unsigned32	10 進の値を表すために使用される、32 ビットの数値。0 ～ 4294967295 の範囲の符号なし整数値が有効です。

set オペレーションは、ローカル ホストとリモート ホストの両方で実行できます。

SNMP トラップ要求および通知要求

トラップは、SNMP マネージャまたは NMS にネットワーク状態を警告する SNMP 通知です。

SNMP インフォーム要求は、SNMP マネージャにネットワーク状態を警告する SNMP 通知を参照し、SNMP マネージャからの受信の確認を要求します。

SNMP イベントは、SNMP MIB オブジェクト ID 値がサンプリングされたとき、または、SNMP カウンタが定義されたしきい値を超えたときに発生します。通知がイネーブルであり、該当するイベントが設定されている場合、SNMP トラップまたはインフォーム メッセージが生成されます。イベント マネージャ サーバーによって SNMP トラップまたはインフォーム メッセージが受信されたとき、SNMP 通知イベントがトリガーされます。

Embedded Event Manager（EEM）アプレットがトリガーされたときに SNMP トラップまたは通知メッセージを送信するには、 **action info type snmp trap** コマンドと **action info type snmp**

inform コマンドを使用します。CISCO-EMBEDDED-EVENT-MGR-MIB.my を使用して、トラップおよびインフォーム メッセージが定義されます。

SNMP Get および Set オペレーションの EEM Applet 設定

ポリシーをイベント マネージャ サーバーに登録する一方で、SNMP イベントに関連付けられたアクションを設定できます。

SNMP set および get オペレーションの EEM アプレットを設定するには、次の作業を実行します。

始める前に

- SNMP イベントマネージャは、**snmp-server manager** コマンドを使用して設定する必要があります。
- SNMP エンティティへのアクセスを有効にするためには、**snmp-server community** コマンドを使用して、SNMP コミュニティストリングを設定する必要があります。

手順の概要

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. 次のいずれかを実行します。
   - **event snmp oid** *oid-value* **get-type** {**exact** | **next**} **entry-op** *operator* **entry-val** *entry-value*[**exit-comb** | **and**}] [**exit-op** *operator*] [**exit-val** *exit-value*] [**exit-time** *exit-time-value*] **poll-interval** *poll-int-value*
5. **action label info type snmp oid** *oid-value* **get-type** {**exact**| **next**} [**community** *community-string*] [**ipaddr** *ip-address*]
6. **action label info type snmp oid** *oid-value* **set-type** *oid-type* *oid-type-value* **community** *community-string* [**ipaddr** *ip-address*]
7. **action label info type snmp getid** *oid-value* [**community** *community-string*] [**ipaddr** *ip-address*]
8. **exit**

手順の詳細

	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	**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。

	コマンドまたはアクション	目的
ステップ 3	**event manager applet** *applet-name* 例： `Device(config)# event manager applet snmp`	Event Manager にアプレットを登録し、アプレット コンフィギュレーション モードを開始します。

ステップ 4	次のいずれかを実行します。
	• **event snmp oid** *oid-value* **get-type** {**exact** | **next**} **entry-op** *operator* **entry-val** *entry-value*[**exit-comb** | **and**}] [**exit-op** *operator*] [**exit-val** *exit-value*] [**exit-time** *exit-time-value*] **poll-interval** *poll-int-value*
	例：
	`Device(config-applet)# event snmp oid`
	例：
	`1.3.6.1.4.1.9.9.48.1.1.1.6.1 get-type exact`
	例：
	`entry-op lt entry-val 5120000 poll-interval 90`
		EEM アプレットの実行の原因となる、イベント基準を指定します。
		• この例では、空きメモリの値が 5120000 を下回ったときに EEM イベントがトリガーされます。
		• 終了基準はオプションです。指定されない場合、イベントのモニターリングは、すぐに再び有効になります。

ステップ 5	**action label info type snmp oid** *oid-value* **get-type** {**exact**| **next**} [**community** *community-string*] [**ipaddr** *ip-address*]
	例：
	`Device(config-applet)# action 1.3 info type`
	例：
	`snmp oid 1.3.6.1.4.1.9.9.48.1.1.1.6.1 get-type`
	例：
	`exact community public ipaddr 172.17.16.69`
		実行する get オペレーションのタイプを指定します。
		• この例では、get オペレーションのタイプが exact と指定され、コミュニティ ストリングが public と指定されます。

ステップ 6	**action label info type snmp oid** *oid-value* **set-type** *oid-type oid-type-value* **community** *community-string* [**ipaddr** *ip-address*]
	例：
	`Device(config-applet)# action 1.4 info type`
	例：
	`snmp oid 1.3.6.1.4.1.9.9.48.1.1.1.6.1 set-type`
	例：
		（任意）設定される変数を指定します。
		• この例では、sysName.0 変数が set オペレーションに指定され、コミュニティ ストリングに rw が指定されます。
		（注） set オペレーションの場合、SNMP コミュニティ ストリングを指定する必要があります。

	コマンドまたはアクション	目的

	`integer 42220 sysName.0 community rw ipaddr`
	例：
	`172.17.16.69`	

ステップ 7	**action label info type snmp getid** *oid-value* [**community** *community-string*] [**ipaddr** *ip-address*]
	例：
	`Device(config-applet)# action 1.3 info type`
	例：
	`snmp getid community public ipaddr 172.17.16.69`
		（任意）個々の変数が getid オペレーションによって取得される必要があるかどうかを指定します。

ステップ 8	**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。

SNMP OID 通知の EEM アプレットの設定

SNMP 通知を設定するには、次の作業を実行します。

始める前に

- SNMP イベントマネージャを、**snmp-server manager** コマンドを使用して設定し、SNMP エージェントが EEM ポリシーのために生成された SNMP トラップを送受信するように設定する必要があります。
- SNMP トラップとインフォームを **snmp-server enable traps event-manager** および **snmp-server enable traps** コマンドを使用して有効にして、トラップ要求とインフォーム要求をデバイスからイベントマネージャサーバーに送信できるようにする必要があります。

手順の概要

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. 次のいずれかを実行します。
   - **event snmp oid** *oid-value* **get-type** {**exact** | **next**} **entry-op** *operator* **entry-val** *entry-value*[**exit-comb** | **and**}] [**exit-op** *operator*] [**exit-val** *exit-value*] [**exit-time** *exit-time-value*] **poll-interval** *poll-int-value*
5. **action label info type snmp var** *variable-name* **oid** *oid-value* *oid-type oid-type-value*

6. **action label info type snmp trap enterprise-oid** *enterprise-oid-value* **generic-trapnum** *generic-trap-number* **specific-trapnum** *specific-trap-number* **trap-oid** *trap-oid-value* **trap-var** *trap-variable*
7. **action label info type snmp inform trap-oid** *trap-oid-value* **trap-var** *trap-variable* **community** *community-string* **ipaddr** *ip-address*
8. **exit**

手順の詳細

	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	**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 3	**event manager applet** *applet-name* 例： `Device(config)# event manager applet snmp`	Event Manager にアプレットを登録し、アプレット コンフィギュレーション モードを開始します。

ステップ 4	次のいずれかを実行します。
	• **event snmp oid** *oid-value* **get-type** {**exact** | **next**} **entry-op** *operator* **entry-val** *entry-value*[**exit-comb** | **and**}] [**exit-op** *operator*] [**exit-val** *exit-value*] [**exit-time** *exit-time-value*] **poll-interval** *poll-int-value*
	例：
	`Device(config-applet)# event snmp oid`
	例：
	`1.3.6.1.4.1.9.9.48.1.1.1.6.1 get-type exact`
	例：
	`entry-op lt entry-val 5120000 poll-interval 90`
		EEM アプレットの実行の原因となる、イベント基準を指定します。
		• この例では、空きメモリの値が 5120000 を下回ったときに EEM イベントがトリガーされます。
		• 終了基準はオプションです。指定されない場合、イベントのモニターリングは、すぐに再び有効になります。

ステップ 5	**action label info type snmp var** *variable-name* **oid** *oid-value* *oid-type oid-type-value* 例： `Device(config-applet)# action 1.3 info type` 例：	管理対象オブジェクトのインスタンスとその値を指定します。 • この例では、sysDescr.0 変数が使用されています。

	コマンドまたはアクション	目的

	`snmp var sysDescr.0 oid`
	例：
	`1.3.6.1.4.1.9.9.48.1.1.1.6.1 integer 4220`	

ステップ 6	**action label info type snmp trap enterprise-oid** *enterprise-oid-value* **generic-trapnum** *generic-trap-number* **specific-trapnum** *specific-trap-number* **trap-oid** *trap-oid-value* **trap-var** *trap-variable*
	例：
	`Device(config-applet)# action 1.4 info type`
	例：
	`snmp trap enterprise-oid 1.3.6.1.4.1.1`
	例：
	`generic-trapnum 4 specific-trapnum 7 trap-oid`
	例：
	`1.3.6.1.4.1.226.0.2.1 trap-var sysUpTime.0`
		EEM アプレットがトリガーされたときに SNMP トラップを生成します。
		• この例では、authenticationFailure トラップが生成されます。
		（注） 固有のトラップ番号は、enterprise イベントが発生したときに生成される enterprise-specific トラップを示します。標準トラップ番号が 6 に設定されていない場合、指定した固有のトラップ番号がトラップの生成に使用されます。

ステップ 7	**action label info type snmp inform trap-oid** *trap-oid-value* **trap-var** *trap-variable* **community** *community-string* **ipaddr** *ip-address*
	例：
	`Device(config-applet)# action 1.4 info type`
	例：
	`snmp inform trap-oid 1.3.6.1.4.1.1.226.0.2.1`
	例：
	`trap-var sysUpTime.0 community public ipaddr`
	例：
	`172.69.16.2`
		EEM アプレットがトリガーされたときに SNMP インフォーム要求を生成します。
		• この例では、sysUpTime.0 変数のインフォーム要求が生成されます。

ステップ 8	**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、特権モードに戻ります。

EEM アプレットの可変ロジックの設定

EEM アプレットの可変ロジック機能は、EEM アプレット内に条件付きロジックを適用する機能を追加します。アプレットには、可変ロジックが導入される前は、イベントがトリガーされたときに各アクションが設定された順に実行されるリニア構造だけがありました。条件付きロジックは、アプレット内のアクションのフローを条件式に従って変更する制御構造を追加します。各制御構造には、ループ アクションや、構造を実行するかどうかを決定する if/else アクションを含むアプレット アクションのリストが含まれます。

アプレットコンフィギュレーションモードの情報は、action コマンドの内容を設定するための背景として示されます。

Tool Command Language（Tcl）とアプレット（CLI）ベースの EEM ポリシーの間で一貫したユーザー インターフェイスを実現するには、次の基準に従います。

- Tcl ベースの実装では、イベント仕様基準は TCL で記述されます。
- アプレット ベースの実装では、イベント仕様データは CLI アプレット サブモード コンフィギュレーション文を使用して記述されます。

アプレット コンフィギュレーション モードは、event manager applet コマンドを使用して開始します。アプレットコンフィギュレーションモードでは、config プロンプトが、(config-applet)# に変わります。アプレット コンフィギュレーション モードでは、2 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：アプレットが実行される原因となるイベント基準を指定するために使用します。
- action：実行する組み込みアクションを指定するために使用します。

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** アプレットコンフィギュレーション コマンドを使用できます。**action** アプレット コンフィギュレーション コマンドが存在しない場合は、終了時に、このアプレットに文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。このアプレットに文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。アプレット コンフィギュレーション モードでコマンドが指定されない場合は、終了時にアプレットが削除されます。exit applet config コマンドは、アプレットコンフィギュレーション モードを終了するために使用されます。

リリースに応じて、EEM アプレットの可変ロジック機能は次の設定を実行できます。

前提条件

この機能を使用するには、Cisco IOS Release12.4(22)T 以降のリリースを実行している必要があります

EEM アプレットの可変ロジックの設定

EEM 3.0 は、アプレット内で単純な可変ロジックを可能にするための新しいアプレット action コマンドを追加しました。

action コマンドを使用して可変ロジックを設定するには、次の作業を実行します。

条件付きブロックのループの指定

EEM アプレットがトリガーされたときに、条件付きブロックのループを指定するには、次の作業を実行します。次のタスクでは、変数の値が 10 よりも小さいかどうかを確認するために、条件付きループが設定されます。変数の値が 10 よりも小さい場合は、メッセージ「i is $_i」が syslog に書き込まれます。

（注） リリースに応じて、**set**（EEM）コマンドは **action set** コマンドに置き換えられます。詳細については、**action** *label* **set** コマンドを参照してください。特定のリリースで set（EEM）コマンドを入力した場合、IOS パーサーは **set** コマンドを **action** *label* **set** コマンドに変換します。

手順の概要

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **action** *label* **set**
5. **action** *label* **while** *string_op1 operator string_op2*
6. 必要に応じてアクションを追加します。
7. **action** *label* **end**

手順の詳細

	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	**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 3	**event manager applet** *applet-name* 例： `Device(config)# event manager applet condition`	Embedded Event Manager（EEM）にアプレットを登録し、アプレット コンフィギュレーション モードを開始します。
ステップ 4	**action** *label* **set** 例： `Device(config-applet)#action 1.0 set i 2`	イベントに対するアクションを設定します。 • この例では、変数 i の値が 2 に設定されます。

	コマンドまたはアクション	目的
ステップ 5	**action** *label* **while** *string_op1 operator string_op2* 例： `Device(config-applet)# action 2 while $i lt 10`	条件付きブロックのループを指定します。 • この例では、変数 i の値が 10 よりも小さいかどうかを確認するために、ループが設定されます。
ステップ 6	必要に応じてアクションを追加します。 例： `Device(config-applet)# action 3 syslog msg "i is $i"`	action コマンドで指示されたアクションを実行します。 • この例では、メッセージ「i is $_i」が syslog に書き込まれます。
ステップ 7	**action** *label* **end** 例： `Device(config-applet)# action 3 end`	実行中のアクションを終了します。

if else 条件付きブロックの指定

if 条件付き文の開始とそれに続く else 条件付き文を指定するには、次の作業を実行します。if 条件付き文と else 条件付き文は、それぞれを結合して使用することも、別々に使用することもできます。このタスクでは、変数の値が 5 に設定されます。次に、変数の値が 10 よりも小さいかどうかを確認するために、if 条件付きブロックが指定されます。if 条件付きブロックが満たされる場合にメッセージ「x is less than 10」を出力する action コマンドが指定されます。

if 条件付きブロックに続いて、else 条件付き部位ロックが指定されます。if 条件付きブロックが満たされない場合にメッセージ「x is greater than 10」を出力する action コマンドが指定されます。

手順の概要

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **action** *label* **set** *variable-name variable-value*
5. **action** *label* **if** [*stringop1*] {**eq** | **gt** | **ge** | **lt** | **le** | **ne**} [*stringop2*]
6. 必要に応じてアクションを追加します。
7. **action** *label* **else**
8. 必要に応じてアクションを追加します。
9. **end**

手順の詳細

	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	**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 3	**event manager applet** *applet-name* 例： `Device(config)# event manager applet ifcondition`	Embedded Event Manager（EEM）にアプレットを登録し、アプレット コンフィギュレーション モードを開始します。
ステップ 4	**action** *label* **set** *variable-name variable-value* 例： `Device(config-applet)# action 1.0 set x 5`	イベントに対するアクションを設定します。 • この例では、変数 x の値が 5 に設定されます。
ステップ 5	**action** *label* **if** [*stringop1*] {**eq** | **gt** | **ge** | **lt** | **le** | **ne**} [*stringop2*] 例： `Device(config-applet)# action 2.0 if $x lt 10`	if 条件付き文を指定します。 • この例では、if条件付き文は変数の値が 10 よりも小さいかどうかを確認します。
ステップ 6	必要に応じてアクションを追加します。 例： `Device(config-applet)# action 3.0 puts "$x is less than 10"`	action コマンドで指示されたアクションを実行します。 • この例では、メッセージ「5 is less than 10」が画面に表示されます。
ステップ 7	**action** *label* **else** 例： `Device(config-applet)# action 4.0 else`	else 条件付きステートメントを指定します。
ステップ 8	必要に応じてアクションを追加します。 例： `Device(config-applet)# action 5.0`	action コマンドで指示されたアクションを実行します。 • この例では、メッセージ「5 is greater than 10」が画面に表示されます。

	コマンドまたはアクション	目的
ステップ 9	**end** 例： `Device(config-applet)# end`	実行中のアクションを終了します。

foreach 反復文の指定

デリミタをトークン化パターンとして使用して入力文字列上で繰り返す条件付き文を指定するには、次の作業を実行します。foreach反復文は目的の情報を取得するためにコレクションを使用して繰り返すために使用されます。デリミタは、正規表現パターン文字列です。各反復で見つかったトークンは、与えられた iterator 変数に割り当てられます。すべての算術演算は、長整数としてオーバーフローのチェックなしで実行されます。この例では、変数 x の値が 5 に設定されます。反復文は、入力文字列 red、blue、green、orange の間、実行するように設定されます。入力文字列の各エレメントに対して、対応するメッセージが画面に表示されます。

手順の概要

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **action** *label* **foreach** [*string-iterator*] [*string-input*] [*string-delimiter*]
5. 任意の action コマンドを指定します。
6. **action** *label* **end**

手順の詳細

	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	**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 3	**event manager applet** *applet-name* 例： `Device(config)# event manager applet iteration`	Embedded Event Manager（EEM）にアプレットを登録し、アプレット コンフィギュレーション モードを開始します。

	コマンドまたはアクション	目的
ステップ 4	**action** *label* **foreach** [*string-iterator*] [*string-input*] [*string-delimiter*] 例： `Device(config-applet)# action 2.0 foreach iterator "red blue green orange"`	デリミタをトークン化パターンとして使用して、入力文字列上で繰り返します。 • この例では、入力のエレメント、red、blue、green、および、orangeの間、反復が実行されます。
ステップ 5	任意の action コマンドを指定します。 例： `Device(config-applet)# action 3.0 puts "Iterator is $iterator"`	action コマンドで指示されたアクションを実行します。 • この例では、次のメッセージが画面に表示されます。 Iterator is red Iterator is blue Iterator is green Iterator is orange
ステップ 6	**action** *label* **end** 例： `Device(config-applet)# action 4.0 end`	実行中のアクションを終了します。

正規表現の使用

正規表現パターンを入力文字列と比較するには、次の作業を実行します。正規表現を使用すると、比較される文字列として可能性のある文字列のセットを表す規則を指定できます。

手順の概要

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **action** *label* **regexp** *string-pattern string-input* [*string-match* [*string-submatch1*] [*string-submatch2*] [*string-submatch3*]]

手順の詳細

	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）。

	コマンドまたはアクション	目的
ステップ 2	**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 3	**event manager applet** *applet-name* 例： `Device(config)# event manager applet regexp`	Embedded Event Manager（EEM）にアプレットを登録し、アプレット コンフィギュレーション モードを開始します。
ステップ 4	**action** *label* **regexp** *string-pattern string-input* [*string-match* [*string-submatch1*] [*string-submatch2*] [*string-submatch3*]] 例： `Device(config-applet)# action 2.0 regexp "(.*) (.*) (.*)" "red blue green" _match _sub1`	入力文字列と比較する表現パターンを指定します。 • この例では、「red blue green」の入力文字列が指定されます。表現パターンが入力文字列と一致すると、**red blue green** の結果全体が変数の **_match** に格納され、部分一致の **red** は変数の **_sub1** に格納されます。

変数の値の増加

変数の値を増加させるには、次の作業を実行します。このタスクでは変数の値が 20 に設定され、次に値が 12 だけ増加します。

手順の概要

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **action** *label* **set**
5. **action** *label* **increment** *variable-name long-integer*

手順の詳細

	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	**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 3	**event manager applet** *applet-name* 例： `Device(config)# event manager applet increment`	Embedded Event Manager（EEM）にアプレットを登録し、アプレット コンフィギュレーション モードを開始します。
ステップ 4	**action** *label* **set** 例： `Device(config-applet)#action 1.0 set varname 20`	イベントに対するアクションを設定します。 • この例では、変数の値が 20 に設定されます。
ステップ 5	**action** *label* **increment** *variable-name long-integer* 例： `Device(config-applet)#action 2.0 increment varname 12`	変数の値が指定された長整数だけ増加します。 • この例では、変数の値が 12 だけ増加します。

イベント SNMP オブジェクトの設定

SNMP オブジェクトのサンプリングによって実行される Embedded Event Manager（EEM）アプレットの簡易ネットワーク管理プロトコル（SNMP）オブジェクトイベントを登録するには、次の作業を実行します。

手順の概要

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **event snmp-object oid** *oid-value* **type** *value* **sync** {**yes** | **no**} **skip** {**yes** | **no**} **istable** {**yes** | **no**} [**default** *seconds*] [**maxrun** *maxruntime-number*]
5. **exit**

手順の詳細

	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	**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 3	**event manager applet** *applet-name* 例： `Device(config)# event manager applet manual-policy`	Embedded Event Manager にアプレットを登録し、アプレット コンフィギュレーション モードを開始します。

ステップ 4	**event snmp-object oid** *oid-value* **type** *value* **sync** {**yes** | **no**} **skip** {**yes** | **no**} **istable** {**yes** | **no**} [**default** *seconds*] [**maxrun** *maxruntime-number*]
	例：
	`Device(config-applet)# event snmp-object oid 1.9.9.9.9 type gauge sync yes`
	例：
	`action 1 syslog msg "oid = $_snmp_oid"`
	例：
	`action 2 syslog msg "request = $_snmp_request"`
	例：
	`action 3 syslog msg "request_type = $_snmp_request_type"`
		Embedded Event Manager（EEM）アプレット用の簡易ネットワーク管理プロトコル（SNMP）オブジェクトイベントを登録し、オブジェクトの SNMP GET および SET 要求を代行受信します。
		デフォルトでは、このコマンドは設定されていません。このコマンドが設定されると、デフォルトは構文オプションの説明と同一になります。
		• **oid** キーワードは、SNMP オブジェクト識別子（object ID）を指定します。
		• *oid-value* 引数は、SNMP ドット付き表記のデータ エレメントのオブジェクト ID 値です。OID は、関連する MIB（CISCO-EMBEDDED-EVENT-MGR-MIB）内にタイプとして定義され、各タイプはオブジェクト値を保持します。
		• **istable** キーワードは、OID が SNMP テーブルかどうかを指定します。
		• sync キーワードは、アプレットを同期モードで実行するよう指定します。アプレットからの戻りコードは、SNMP 要求に応答するかどうかを示します。コード 0 は「要求に応答しない」、コード 1 は「要求に応答する」を意味します。アプレットからの戻りコードが要求に応答すると、**action snmp-object-value** コマンドを使用して、オブジェクトのアプレットで値が指定されます。
		• **type** キーワードは、オブジェクトのタイプを指定します。
		• *value* 引数はオブジェクトの値です。
		• skip キーワードは、CLI コマンドの実行をスキップするかどうかを指定します

	コマンドまたはアクション	目的
		• **default** キーワードは、アプレットが通常処理する SET 要求または GET 要求の時間を指定します。**default** キーワードが指定されない場合は、デフォルトの時間が 30 秒に設定されます。 • *milliseconds* 引数は、SNMP オブジェクト イベントディテクタがポリシーの終了を待つ時間です。 • **maxrun** キーワードは、アプレットの最大ランタイムを指定します。**maxrun** キーワードを指定した場合、*maxruntime-number* 値を指定する必要があります。**maxrun** キーワードが指定されていない場合、デフォルトのアプレットランタイムは 20 秒です。 • *milliseconds* 引数は、ミリ秒単位のアプレットの最大ランタイムです。この引数が指定されない場合、デフォルトの 20 秒ランタイム制限が使用されます。
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。

AAA 認証の無効化

トリガーされたときに、EEM ポリシーが AAA 認証をバイパスするようにするには、次の作業を実行します。

手順の概要

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name* [**authorization bypass**] [**class** *class-options*] [**trap**]
4. **exit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**enable** 例：	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）。

	コマンドまたはアクション	目的
	`Device> enable`	
ステップ 2	**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 3	**event manager applet** *applet-name* [**authorization bypass**] [**class** *class-options*] [**trap**] 例： `Device(config)# event manager applet one class A authorization bypass`	Embedded Event Manager（EEM）にアプレットを登録し、アプレット コンフィギュレーション モードを開始します。
ステップ 4	**exit** 例： `Device(config-aaplet)# exit`	デバイス コンフィギュレーション アプレット モードを終了し、特権 EXEC モードに戻ります。

Embedded Event Manager アプレットの説明の設定

EEM アプレットについて記述するには、次の作業を実行します。アプレットの説明は、他のアプレット設定の前でも後でも、任意の順序で追加できます。すでに説明があるアプレットに新しい説明を設定すると、現在の説明が上書きされます。アプレットの説明はオプションです。

アプレットに新しい説明を設定するには、次の作業を実行します。

手順の概要

1. **enable**
2. **configure terminal**
3. **event manager applet** *applet-name*
4. **description** *line*
5. **event syslog pattern** *regular-expression*
6. **action** *label* **syslog msg** *msg-text*
7. **end**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**enable** 例：	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）。

	コマンドまたはアクション	目的
	`Device> enable`	
ステップ 2	**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 3	**event manager applet** *applet-name* 例： `Device(config)# event manager applet increment`	EEM にアプレットを登録し、アプレット コンフィギュレーション モードを開始します。
ステップ 4	**description** *line* 例： `Device(config-applet)# description "This applet looks for the word count in syslog messages"`	簡易ネットワーク管理プロトコル（SNMP）のサンプリングによって実行される EEM アプレットの説明を追加または変更します。
ステップ 5	**event syslog pattern** *regular-expression* 例： `Device(config-applet)# event syslog pattern "count"`	syslog メッセージの一致によって実行される Embedded Event Manager（EEM）アプレットのイベント基準を指定します。
ステップ 6	**action** *label* **syslog msg** *msg-text* 例： `Device(config-applet)# action 1 syslog msg hi`	EEM アプレットがトリガーされたときに実行されるアクションを指定します。 • この例では、実行されるアクションは syslog にメッセージを書き込むことです。 • *msg-text* 引数は、文字テキスト、環境変数、またはその両方の組み合わせが可能です。
ステップ 7	**end** 例： `Device(config-applet)# end`	アプレット コンフィギュレーション モードを終了し、特権 EXEC モードに戻ります。

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

Embedded Event Manager アプレットの設定例

次に、一部の EEM イベント ディテクタの EEM アプレット作成例を示します。次の例は、Embedded Event Manager アプレットの登録と定義 （15 ページ）で説明した手順に従っています。

Application-Specific イベント ディテクタ

次に、EventPublish_A という名前のポリシーが、20 秒ごとに実行され、番号が 1 のイベント タイプを、番号 798 のサブシステムにパブリッシュする例を示します。サブシステムの値、798 は、イベントのパブリッシュが EEM ポリシーから発生することを指定します。EventPublish_B という名前の別のポリシーは、EEM イベント タイプ 1 が発生したときに実行されるように、subsystem 798 に登録されます。EventPublish_B ポリシーは実行されるときに、EventPublish_A から引数として渡されたデータを含むメッセージを syslog に送信します。

```
event manager applet EventPublish_A
 event timer watchdog time 20.0
 action 1.0 syslog msg "Applet EventPublish_A"
 action 2.0 publish-event sub-system 798 type 1 arg1 twenty
 exit
event manager applet EventPublish_B
 event application sub-system 798 type 1
 action 1.0 syslog msg "Applet EventPublish_B arg1 $_application_data1"
```

CLI イベント ディテクタ

次に、Cisco IOS **write memory** CLI コマンドが実行されたときに実行する EEM アプレットを指定する例を示します。アプレットは、このイベントが syslog メッセージによって生成した通知を提供します。この例では、**sync** キーワードが yes 引数とともに設定されています。これは、このポリシーの実行が完了したときに、イベントディテクタに通知されることを意味します。ポリシーの終了状態が、CLI コマンドが実行されるかどうかを決定します。この例では、ポリシーの終了状態は 1 に設定され、CLI コマンドは実行されます。

```
event manager applet cli-match
 event cli pattern "write mem.*" sync yes
 action 1.0 syslog msg "$_cli_msg Command Executed"
 set 2.0 _exit_status 1
```

次に、**cli pattern** と test 引数を照合するアプレットの例を示します。**show access-list test** が入力されると、CLI イベントディテクタは、test 引数を照合し、アプレットがトリガーされます。**debug event manager detector cli** 出力が追加され、num_matches が 1 に設定されていることが示されます。

```
!
```

```
event manager applet EEM-PIPE-TEST
 event cli pattern "test" sync yes
 action 1.0 syslog msg "Pattern matched!"
!
*Aug 23 23:19:59.827: check_eem_cli_policy_handler: command_string=show access-lists
test
*Aug 23 23:19:59.827: check_eem_cli_policy_handler: num_matches = 1, response_code = 4
*Aug 23 23:19:59.843: %HA_EM-6-LOG: EEM-PIPE-TEST: Pattern matched!
```

（注） CLI イベント ディテクタによる機能は、有効な IOS CLI コマンドでの正規表現パターン比較機能だけです。これには、リダイレクションが使用される場合のパイプ記号（|）以降のテキストは含まれません。

次に、**show version | include test** が入力された場合にアプレットがトリガーされなかった例を示します。CLI イベントディテクタでパイプ（|）文字の後ろに入力された文字との一致がなく、**debug event manager detector cli** 出力で num_matches がゼロと表示されいるためにトリガーされませんでした。

```
*Aug 23 23:20:16.827: check_eem_cli_policy_handler: command_string=show version
*Aug 23 23:20:16.827: check_eem_cli_policy_handler: num_matches = 0, response_code = 1
```

Counter イベント ディテクタおよび Timer イベント ディテクタ

次に、EventCounter_A ポリシーが 1 分に 1 回実行されるように設定され、既知のカウンタ critical_errors を増加させる例を示します。2 番目のポリシー、EventCounter_B は、critical_errors がという既知のカウンタがしきい値 3 を超えたときにトリガーされるように登録されます。EventCounter_B ポリシーが実行されたとき、カウンタは 0 にリセットされます。

```
event manager applet EventCounter_A
 event timer watchdog time 60.0
 action 1.0 syslog msg “EventCounter_A”
 action 2.0 counter name critical_errors op inc value 1
 exit
event manager applet EventCounter_B
 event counter name critical_errors entry-op gt entry-val 3 exit-op lt exit-val 3
 action 1.0 syslog msg “EventCounter_B”
 action 2.0 counter name critical_errors op set value 0
```

Interface Counter イベント ディテクタ

次に、EventInterface という名前のポリシーが、ファスト イーサネット インターフェイス 0/0 の receive_throttle カウンタが 5 ずつ増加するたびに、トリガーされる例を示します。カウンタをチェックするポーリング間隔は、90 秒ごとに 1 回実行するように指定されます。

```
event manager applet EventInterface
 event interface name FastEthernet0/0 parameter receive_throttle entry-op ge entry-val
5
 entry-val-is-increment true poll-interval 90
 action 1.0 syslog msg "Applet EventInterface"
```

Resource イベント ディテクタ

次に、CPU 使用率が高い場合に報告するように定義された、ポリシーの ERM イベント レポートに基づいてイベント基準を指定する例を示します。

```
event manager applet policy-one
 event resource policy cpu-high
 action 1.0 syslog msg “CPU high at $_resource_current_value percent”
```

RF イベント ディテクタ

RF イベント ディテクタは、デュアル ルート プロセッサ（RP）を備えたネットワーキング デバイスでだけ利用できます。次に、RF 状態変化通知に基づいてイベント基準を指定する例を示します。

```
event manager applet start-rf
 event rf event rf_prog_initialization
 action 1.0 syslog msg "rf state rf_prog_initialization reached"
```

Remote Procedure Call（RPC）イベント ディテクタ

RPC イベント ディテクタによって、外部エンティティがデバイスに対して Simple Object Access Protocol（SOAP）要求を作成でき、定義された EEM ポリシーまたはスクリプトを実行できます。次に、Event_RPC という名前の EEM アプレットが EEM スクリプトを実行するように登録されている例を示します。

```
event manager applet Event_RPC
 event rpc
 action print puts "hello there"
```

次に、SOAP 要求と返信メッセージの形式の例を示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://www.cisco.com/eem.xsd">
  <SOAP:Body>
    <run_eemscript>
      <script_name>Event_RPC</script_name>
    </run_eemscript>
  </SOAP:Body>
</SOAP:Envelope>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?><SOAP:Envelope
xmlns:SOAP="http://www.cisco.com/eem.xsd"><SOAP:Body>
<run_eemscript_response><return_code>0</return_code><output></output></run_eemscript_response></SOAP:Body></SOAP:Envelope>]]>]]>
```

SNMP イベント ディテクタ

次に、CPU 使用率が 75% を上回ったときに実行する EEM アプレットを指定する例を示します。EEM アプレットを実行すると、CLI コマンドの **enable** と **show cpu processes** が実行され、**show cpu processes** コマンドの結果が含まれている電子メールがエンジニアに送信されます。

```
event manager applet snmpcpuge75
 event snmp oid 1.3.6.1.4.1.9.9.109.1.1.1.1.3.1 get-type exact entry-op ge entry-val 75
```

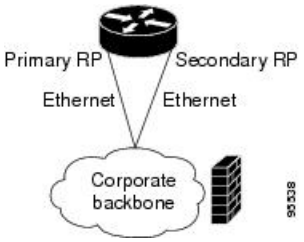
```
poll-interval 10
 action 1.0 cli command "enable"
 action 2.0 cli command "show process cpu"
 action 3.0 mail server "192.168.1.146" to "engineer@cisco.com" from "devtest@cisco.com"
subject "B25 PBX Alert" body "$_cli_result"
```

次の例はより複雑で、プライマリ ルート プロセッサ（RP）がメモリ不足で実行されているときに、セカンダリ（冗長）RP に切り替えるように EEM アプレットを設定する例を示します。

次に、メモリ リークの原因となるソフトウェア障害に対する予防措置を実施する例を示します。ここで実行されるアクションは、メモリ リークの可能性が検出されたときに、冗長 RP へ切り替えることによってダウンタイムを削減することを意図しています。

次の図は、EEM イメージを実行しているデュアル RP デバイスを示しています。EEM アプレットは、**event manager applet** コマンドを使用して CLI によって登録されています。プライマリ RP の使用可能なメモリが、指定されたしきい値 5,120,000 バイトを下回ったときに、アプレットは実行されます。アプレットのアクションは、利用可能なメモリのバイト数を示すメッセージを syslog に書き込み、セカンダリ RP へスイッチします。

図 *1:* デュアル *RP* トポロジ



ポリシーの登録に使用されるコマンドは、次のとおりです。

```
event manager applet memory-demo
 event snmp oid 1.3.6.1.4.1.9.9.48.1.1.1.6.1 get-type exact entry-op lt entry-val 5120000
poll-interval 90
 action 1.0 syslog priority critical msg "Memory exhausted; current available memory is
$_snmp_oid_val bytes"
 action 2.0 force-switchover
```

登録済みのアプレットは、**show event manager policy registered** コマンドを使用して表示できます。

```
Device# show event manager policy registered
No.  Type    Event Type          Time Registered           Name
1    applet  snmp                Thu Jan30  05:57:16 2003  memory-demo
 oid {1.3.6.1.4.1.9.9.48.1.1.1.6.1} get-type exact entry-op lt entry-val {5120000}
poll-interval 90
 action 1.0 syslog priority critical msg "Memory exhausted; current available memory is
$_snmp_oid_val bytes"
 action 2.0 force-switchover
```

この例を示すため、デバイスでメモリを強制的に枯渇させ、一連の **show memory** コマンドを実行させてメモリの枯渇を監視します。

```
Device# show memory
                Head    Total(b)     Used(b)     Free(b)   Lowest(b)  Largest(b)
Processor   53585260   212348444   119523060    92825384    92825384    92365916
```

```
Fast        53565260      131080       70360       60720       60720       60668
Device# show memory
                Head    Total(b)     Used(b)     Free(b)   Lowest(b)  Largest(b)
Processor   53585260   212364664   164509492    47855172    47855172    47169340
Fast        53565260      131080       70360       60720       60720       60668
Device# show memory
                Head    Total(b)     Used(b)     Free(b)   Lowest(b)  Largest(b)
Processor   53585260   212369492   179488300    32881192    32881192    32127556
Fast        53565260      131080       70360       60720       60720       60668
```

しきい値に達したときに、EEM イベントがトリガーされます。memory-demo という名前のアプレットが実行され、これによって、syslog メッセージがコンソールに出力され、セカンダリ RP へのスイッチが発生します。次のメッセージが記録されます。

```
00:08:31: %HA_EM-2-LOG: memory-demo: Memory exhausted; current available memory is
4484196 bytes
00:08:31: %HA_EM-6-FMS_SWITCH_HARDWARE: fh_io_msg: Policy has requested a hardware
switchover
```

次に、プライマリ RP とセカンダリ（冗長）RP の両方での **show running-config** コマンドの出力の一部を示します。

```
redundancy
 mode sso
.
.
!
event manager applet memory-demo
 event snmp oid 1.3.6.1.4.1.9.9.48.1.1.1.6.1 get-type exact entry-op lt entry-val
5120000 poll-interval 90
 action 1.0 syslog priority critical msg "Memory exhausted; current available memory
is $_snmp_oid_val bytes"
 action 2.0 force-switchover
```

SNMP 通知イベント ディテクタ

次に、**event snmp-notification** を設定する前に、**snmp-server community** パブリック RW コマンドと **snmp-server manager** コマンドを設定する例を示します。

```
snmp-server community public RW
 snmp-server manager
```

次に、値が 10 であるオブジェクト ID 1 の宛先 IP アドレス 192.168.1.1 でデバイスが SNMP 通知を受け取ったときに、EEM スクリプトを実行するように SNMP_Notification という名前の EEM アプレットを登録する例を示します。

```
event manager applet SNMP_Notification
 event snmp-notification dest_ip_address 192.168.1.1 oid 1 op eq oid-value 10
 action 1 policy eem_script
```

syslog イベント ディテクタ

次に、syslog がイーサネット インターフェイス 1/0 のダウンを認識したときに実行する EEM アプレットを指定する例を示します。アプレットはインターフェイスに関するメッセージを syslog に送信します。

```
event manager applet interface-down
 event syslog pattern "˜.*UPDOWN.*Ethernet1/0.*" occurs 4
 action 1.0 syslog msg "Ethernet interface 1/0 changed state 4 times"
```

Embedded Event Manager アプレットの設定例

ID イベント ディテクタの例

次に、「EventIdentity」というポリシーが、ファストイーサネット インターフェイス 0 で認証が成功するたびにトリガーされる例を示します。

```
event manager applet EventIdentity
 event identity interface FastEthernet0 authc success
action 1.0 syslog msg "Applet EventIdentity"
```

MAT イベント ディテクタの例

次に、「EventMat」というポリシーが、mac-address-table で MAC アドレスが学習されるたびにトリガーされる例を示します。

```
event manager applet EventMat
 event mat interface FastEthernet0
action 1.0 syslog msg "Applet EventMat"
```

ネイバー検出イベント ディテクタの例

次に、「EventNeighbor」というポリシーが、Cisco Discovery Protocol（CDP）キャッシュ エントリが変化するときにトリガーされる例を示します。

```
event manager applet EventNeighbor
 event neighbor-discovery interface FastEthernet0 cdp all
action 1.0 syslog msg "Applet EventNeighbor"
```

Embedded Event Manager の手動によるポリシー実行の例

次に、手動で実行する EEM ポリシー（アプレットまたはスクリプト）の設定に None イベント ディテクタを使用する例を示します。

イベント マネージャ run コマンドの使用

次に、**event manager run** コマンドを使用して、手動でポリシーを実行する例を示します。ポリシーはアプレット コンフィギュレーション モードで **event none** コマンドを使用して登録されてから、グローバル コンフィギュレーション モードで **event manager run** コマンドを使用して実行されます。

```
event manager applet manual-policy
 event none
 action 1.0 syslog msg "Manual-policy triggered"
 end
```

```
!
event manager run manual-policy
```

action policy コマンドの使用

次に、**action policy** コマンドを使用して、手動でポリシーを実行する例を示します。ポリシーはアプレット コンフィギュレーション モードで **event none** コマンドを使用して登録されてから、アプレット コンフィギュレーション モードで **action policy** コマンドを使用して実行されます。

```
event manager applet manual-policy
 event none
 action 1.0 syslog msg "Manual-policy triggered"
 exit
!
event manager applet manual-policy-two
 event none
 action 1.0 policy manual-policy
 end
!
event manager run manual-policy-two
```

Embedded Event Manager Watchdog System Monitor（Cisco IOS）イベント ディテクタの設定例

次に、Cisco IOS watchdog system monitor (IOSWDSysMon) イベント ディテクタの動作を具体的に表示する 3 個の EEM アプレットの設定例を示します。

Watchdog System Monitor サンプル 1 ポリシー

第 1 のポリシーは、IP Input という名前のプロセスの平均 CPU 使用率が 10 秒間 1% 以上になったときにアプレットをトリガーします。

```
event manager applet IOSWD_Sample1
 event ioswdsysmon sub1 cpu-proc taskname "IP Input" op ge val 1 period 10
 action 1.0 syslog msg "IOSWD_Sample1 Policy Triggered"
```

Watchdog System Monitor サンプル 2 ポリシー

第 2 のポリシーは、Net Input という名前のプロセスによる合計メモリ使用量が 100 kb を超えたときアプレットをトリガーします。

```
event manager applet IOSWD_Sample2
 event ioswdsysmon sub1 mem-proc taskname "Net Input" op gt val 100 is-percent false
 action 1.0 syslog msg "IOSWD_Sample2 Policy Triggered"
```

Watchdog System Monitor サンプル 3 ポリシー

第 3 のポリシーは、IP RIB Update という名前のプロセスによる合計メモリ使用量が、60 秒のサンプリング時間全体で、50% を超えて増加したときにアプレットをトリガーします。

```
event manager applet IOSWD_Sample3
 event ioswdsysmon sub1 mem-proc taskname "IP RIB Update" op gt val 50 is-percent true
period 60
 action 1.0 syslog msg "IOSWD_Sample3 Policy Triggered"
```

3 個のポリシーが設定され、複数のワークステーションからネットワーキング デバイスに対して繰り返し大量の ping が実行されます。そのためネットワーキング デバイスは一定の利用量を記録します。これにより、ポリシー 1 およびポリシー 2 がトリガーされ、コンソールに次のメッセージが表示されます。

```
00:42:23: %HA_EM-6-LOG: IOSWD_Sample1: IOSWD_Sample1 Policy Triggered
00:42:47: %HA_EM-6-LOG: IOSWD_Sample2: IOSWD_Sample2 Policy Triggered
```

登録したポリシーを表示するには、**show event manager policy registered** コマンドを使用します。

```
Device# show event manager policy registered
No.  Class   Type    Event Type          Trap  Time Registered           Name
1    applet  system  ioswdsysmon         Off   Fri Jul 23 02:27:28 2004  IOSWD_Sample1
 sub1 cpu_util {taskname {IP Input} op ge val 1 period 10.000 }
 action 1.0 syslog msg "IOSWD_Sample1 Policy Triggered"
2    applet  system  ioswdsysmon         Off   Fri Jul 23 02:23:52 2004  IOSWD_Sample2
 sub1 mem_used {taskname {Net Input} op gt val 100 is_percent FALSE}
 action 1.0 syslog msg "IOSWD_Sample2 Policy Triggered"
3    applet  system  ioswdsysmon         Off   Fri Jul 23 03:07:38 2004  IOSWD_Sample3
 sub1 mem_used {taskname {IP RIB Update} op gt val 50 is_percent TRUE period 60.000 }
 action 1.0 syslog msg "IOSWD_Sample3 Policy Triggered"
```

SNMP ライブラリ拡張の設定例

SNMP get オペレーションの例

次に、get 要求をローカル ホストに送信する例を示します。

```
Device(config)# event manager applet snmp
Device(config-applet)# event snmp oid
 1.3.6.1.2.1.1.1.0 get-type exact entry-op
 lt entry-val
 5120000 poll-interval
90
Device(config-applet)# action 1.3 info type snmp oid
 1.3.6.1.2.1.1.1.0 get-type exact
community
 public
Device(config-applet)# action 1.3 info type snmp oid
 1.3.6.1.2.1.1.4.0 get-type next community
public
```

次のログ メッセージが SNMP イベント マネージャ ログに書き込まれます。

```
1d03h:%HA_EM-6-LOG: lg: 1.3.6.1.2.1.1.1.0
1d04h:%HA_EM-6-LOG: lgn: 1.3.6.1.2.1.1.5.0
```

次に、get 要求をリモート ホストに送信する例を示します。

```
Device(config)# event manager applet snmp
Device(config-applet)# event snmp oid
 1.3.6.1.2.1.1.1.0 get-type exact entry-op
 lt entry-val
 5120000 poll-interval
 90
Device(config-applet)# action 1.3 info type snmp oid
1.3.6.1.2.1.1.4.0 get-type next community
 public ipaddr
 172.17.16.69
Device(config-applet)# action 1.3 info type snmp getid
 1.3.6.1.2.1.1.1.0 community
 public ipaddr
 172.17.16.69
```

次のログ メッセージが SNMP イベント マネージャ ログに書き込まれます。

```
1d03h:%HA_EM-6-LOG: lg: 1.3.6.1.2.1.1.1.0
1d04h:%HA_EM-6-LOG: lgn: 1.3.6.1.2.1.1.5.0
```

SNMP GetID オペレーションの例

次に、getid 要求をローカル ホストに送信する例を示します。

```
Device(config)# event manager applet snmp
Device(config-applet)# event snmp oid
1.3.6.1.2.1.1.1.0 get-type exact entry-op
 lt entry-val
5120000 poll-interval
 90
Device(config-applet)# action 1.3 info type snmp getid
 community
 public
```

次のログ メッセージが SNMP イベント マネージャ ログに書き込まれます。

```
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysname_oid=1.3.6.1.2.1.1.5.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysname_value=jubjub.cisco.com
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_syslocation_oid=1.3.6.1.2.1.1.6.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_syslocation_value=
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysdescr_oid=1.3.6.1.2.1.1.1.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysobjectid_oid=1.3.6.1.2.1.1.2.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysobjectid_value=products.222
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysuptime_oid=1.3.6.1.2.1.1.3.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysuptime_oid=10131676
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_syscontact_oid=1.3.6.1.2.1.1.4.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_syscontact_value=YYY
```

次に、getid 要求をリモート ホストに送信する例を示します。

```
Device(config)# event manager applet snmp
Device(config-applet)# event snmp oid
 1.3.6.1.2.1.1.1.0 get-type exact entry-op
 lt entry-val
 5120000 poll-interval
 90
Device(config-applet)# action 1.3 info type snmp getid
 1.3.6.1.2.1.1.1.0 community
```

```
 public ipaddr
 172.17.16.69
```

次のログ メッセージが SNMP イベント マネージャ ログに書き込まれます。

```
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysname_oid=1.3.6.1.2.1.1.5.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysname_value=jubjub.cisco.com
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_syslocation_oid=1.3.6.1.2.1.1.6.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_syslocation_value=
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysdescr_oid=1.3.6.1.2.1.1.1.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysobjectid_oid=1.3.6.1.2.1.1.2.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysobjectid_value=products.222
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysuptime_oid=1.3.6.1.2.1.1.3.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_sysuptime_oid=10131676
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_syscontact_oid=1.3.6.1.2.1.1.4.0
1d04h:%HA_EM-6-LOG: lgid: _info_snmp_syscontact_value=YYY
```

set オペレーションの例

次に、set オペレーションをローカル ホストで実行する例を示します。

```
Device(config)# event manager applet snmp
Device(config-applet)# event snmp oid
 1.3.6.1.2.1.1.1.0 get-type exact entry-op
 lt entry-val
 5120000 poll-interval
 90
Device(config-applet)# action 1.3 info type snmp oid
1.3.6.1.2.1.1.4.0 set-type
 integer
 5 sysName.0 community
 public
```

次のログ メッセージが SNMP イベント マネージャ ログに書き込まれます。

```
1d04h:%HA_EM-6-LOG: lset: 1.3.6.1.2.1.1.4.0
1d04h:%HA_EM-6-LOG: lset: XXX
```

次に、set オペレーションをリモート ホストで実行する例を示します。

```
Device(config)# event manager applet snmp
Device(config-applet)# event snmp oid
1.3.6.1.2.1.1.1.0 get-type exact entry-op
 lt entry-val
 5120000 poll-interval
90
Device(config-applet)# action 1.3 info type snmp oid
1.3.6.1.2.1.1.4.0 set-type integer
 5 sysName.0 community
 public ipaddr
172.17.16.69
```

次のログ メッセージが SNMP イベント マネージャ ログに書き込まれます。

```
1d04h:%HA_EM-6-LOG: lset: 1.3.6.1.2.1.1.4.0
1d04h:%HA_EM-6-LOG: lset: XXX
```

SNMP 通知の生成の例

次に、sysUpTime.0 変数の SNMP トラップを設定する例を示します。

```
Device(config)# event manager applet snmp
Device(config-applet)# event snmp oid
 1.3.6.1.4.1.9.9.48.1.1.1.6.1 get-type exact entry-op
lt entry-val
 5120000 poll-interval
90
Device(config-applet)# action 1.3 info type snmp var
 sysUpTime.0 oid
1.3.6.1.4.1.9.9.43.1.1.6.1.3.41 integer
 2
Device(config-applet)# action 1.4 info type snmp trap
 enterprise-oid
 ciscoSyslogMIB.2 generic-trapnum
6 specific-trapnum
1 trap-oid
 1.3.6.1.4.1.9.9.41.2.0.1 trap-var
sysUpTime.0
```

debug snmp packets コマンドがイネーブルにされている場合、次の出力が生成されます。

```
Device# debug snmp packets
1d04h: SNMP: Queuing packet to 172.69.16.2
1d04h: SNMP: V1 Trap, ent ciscoSyslogMIB.2, addr 172.19.rap 1
clogHistoryEntry.3 = 4
clogHistoryEntry.6 = 9999
1d04h: SNMP: Queuing packet to 172.19.208.130
1d04h: SNMP: V1 Trap, ent ciscoSyslogMIB.2, addr 172.19.rap 1
clogHistoryEntry.3 = 4
clogHistoryEntry.6 = 9999
1d04h: SNMP: Packet sent via UDP to 172.69.16.2
1d04h: SNMP: Packet sent via UDP to 172.69.16.2
infra-view10:
Packet Dump:
30 53 02 01 00 04 04 63 6f 6d 6d a4 48 06 09 2b
06 01 04 01 09 09 29 02 40 04 ac 13 d1 17 02 01
06 02 01 01 43 04 00 9b 82 5d 30 29 30 12 06 0d
2b 06 01 04 01 09 09 29 01 02 03 01 03 02 01 04
30 13 06 0d 2b 06 01 04 01 09 09 29 01 02 03 01
06 02 02 27 0f
Received SNMPv1 Trap:
Community: comm
Enterprise: ciscoSyslogMIBNotificationPrefix
Agent-addr: 172.19.209.23
Enterprise Specific trap.
Enterprise Specific trap: 1
Time Ticks: 10191453
clogHistSeverity = error(4)
clogHistTimestamp = 9999
```

次に、sysUpTime.0 変数の SNMP インフォーム要求を設定する例を示します。

```
Device(config)# event manager applet snmp
Device(config-applet)# event snmp oid
1.3.6.1.4.1.9.9.48.1.1.1.6.1 get-type exact entry-op
 lt entry-val
 5120000 poll-interval
 90
```

```
Device(config-applet)# action 1.3 info type snmp var
 sysUpTime.0 oid
1.3.6.1.4.1.9.9.43.1.1.6.1.3.41 integer
 2
Device(config-applet)# action 1.4 info type snmp inform
trap-oid
 1.3.6.1.4.1.9.9.43.2.0.1 trap-var
 sysUpTime.0 community
 public ipaddr
 172.19.209.24
```

debug snmp packets コマンドがイネーブルにされている場合、次の出力が生成されます。

```
Device# debug snmp packets
1d04h: SNMP: Inform request, reqid 24, errstat 0, erridx 0
sysUpTime.0 = 10244391
snmpTrapOID.0 = ciscoConfigManMIB.2.0.1
ccmHistoryEventEntry.3.40 = 1
1d04h: SNMP: Packet sent via UDP to 172.19.209.24.162
1d04h: SNMP: Packet received via UDP from 172.19.209.24 on FastEthernet0/0
1d04h: SNMP: Response, reqid 24, errstat 0, erridx 0
1d04h: SNMP: Response, reqid 24, errstat 0, erridx 0
1d04h: SNMP: Inform request, reqid 25, errstat 0, erridx 0
sysUpTime.0 = 10244396
snmpTrapOID.0 = ciscoConfigManMIB.2.0.1
ccmHistoryEventEntry.3.41 = 2
1d04h: SNMP: Packet sent via UDP to 172.19.209.24.162
1d04h: SNMP: Packet received via UDP from 172.19.209.24 on FastEthernet0/0
1d04h: SNMP: Response, reqid 25, errstat 0, erridx 0
1d04h: SNMP: Response, reqid 25, errstat 0, erridx 0
Device# debug snmp packets
5d04h: SNMP: Packet received via UDP from 172.19.209.23 on FastEthernet0/0
5d04h: SNMP: Inform request, reqid 24, errstat 0, erridx 0
sysUpTime.0 = 10244391
snmpTrapOID.0 = ciscoConfigManMIB.2.0.1
ccmHistoryEventEntry.3.40 = 1
5d04h: dest if_index = 1
5d04h: dest ip addr= 172.19.209.24
5d04h: SNMP: Response, reqid 24, errstat 0, erridx 0
5d04h: SNMP: Packet sent via UDP to 172.19.209.23.57748
5d04h: SNMP: Packet received via UDP from 172.19.209.23 on FastEthernet0/0
5d04h: SNMP: Inform request, reqid 25, errstat 0, erridx 0
```

EEM アプレットの可変ロジックの設定例

このセクションでは、一部の選択された action コマンドの例を示します。アプレット内の可変ロジックをサポートするすべての action コマンドについては、次の表を参照してください。

この例では、条件付きのループである **while**、**if** および **foreach** を使用してデータを出力します。**action divide**、**action increment** および **action puts** のようなその他のアクションコマンドは、条件が満たされている場合に実行されるアクションを定義するために使用します。

```
event manager applet printdata
event none
action 100 set colors "red green blue"
action 101 set shapes "square triangle rectange"
action 102 set i "1"
action 103 while $i lt 6
action 104  divide $i 2
```

```
action 105  if $_remainder eq 1
action 106   foreach _iterator "$colors"
action 107    puts nonewline "$_iterator "
action 108   end
action 109   puts ""
action 110  else
action 111   foreach _iterator "$shapes"
action 112    puts nonewline "$_iterator "
action 113   end
action 114   puts ""
action 115  end
action 116  increment i
action 117 end
```

イベント マネージャ アプレット ex が実行されると、次の出力が得られます。

```
event manager run printdata
red green blue
square triangle rectange
red green blue
square triangle rectange
red green blue
```

次の例では、2 個の環境変数、poll_interface と max_rx_rate が、それぞれ、F0/0 と 3 に設定されます。30 秒ごとに、インターフェイスで rx 比率の調査が行われます。rx 比率がしきい値を上回った場合は、syslog メッセージが表示されます。

このアプレットは、インターフェイスの調査に foreach 条件付き文を使用します。また、RXPS に属する値を EEM 環境変数に設定された max_rx_rate と比較するために、if 条件付きブロックを使用します。

```
event manager environment poll_interfaces F0/0
event manager environment max_rx_rate 3
ev man app check_rx_rate
ev timer watchdog name rx_timer time 30
action 100 foreach int $poll_interfaces
action 101  cli command "en"
action 102  cli command "show int $int summ | beg ------"
action 103  foreach line $_cli_result "\n"
action 105  regexp ".*[0-9]+\s+[0-9]+\s+[0-9]+\s+[0-9]+\s+[0-9]+\s+([0-9]+)\s+.*" $line
 junk rxps
action 106   if $_regexp_result eq 1
action 107    if $rxps gt $max_rx_rate
action 108     syslog msg "Warning rx rate for $int is > than threshold. Current value
is $rxps
 (threshold is $max_rx_rate)"
action 109    end
action 110   end
action 111  end
action 112 end
```

syslog メッセージ例：

```
Oct 16 09:29:26.153: %HA_EM-6-LOG: c: Warning rx rate for F0/0 is > than threshold.
 Current value is 4 (threshold is 3)
The output of show int F0/0 summ is of the format:

#show int f0/0 summ

 *: interface is up
 IHQ: pkts in input hold queue     IQD: pkts dropped from input queue
```

```
 OHQ: pkts in output hold queue    OQD: pkts dropped from output queue
 RXBS: rx rate (bits/sec)          RXPS: rx rate (pkts/sec)
 TXBS: tx rate (bits/sec)          TXPS: tx rate (pkts/sec)
 TRTL: throttle count

  Interface               IHQ   IQD  OHQ   OQD  RXBS RXPS  TXBS TXPS TRTL
------------------------------------------------------------------------
* FastEthernet0/0          0 87283    0     0     0    0     0    0    0
```

（注） アプレット内の可変ロジックをサポートするその他の action コマンドを使用するには、次の表にあるコマンドを使用してください。

表 *6 :* 使用できる *action* コマンド

Action コマンド	**目的**
action add	EEM アプレットがトリガーされたときに 2 つの変数の値を足します。
action append	EEM アプレットがトリガーされたときに、与えられた値に変数の現在の値をアペンドします。
action break	EEM アプレットがトリガーされたときに、すぐにアクションのループを終了します。
action comment	EEM アプレットがトリガーされたときにアプレットにコメントを追加します。
action context retrieve	EEM アプレットがトリガーされたときに、与えられたコンテキスト名キーのセットで特定される変数を取得します。
action context save	EEM アプレットがトリガーされたときに、複数のポリシー トリガー全体の情報を保存します。
action continue	EEM アプレットがトリガーされたときに、アクションのループを継続します。
action decrement	EEM アプレットがトリガーされたときに、変数の値をデクリメントします。
action divide	EEM アプレットがトリガーされたときに、与えられた序数の値で非除数を割ります。
action else	EEM アプレットがトリガーされたときに、if / else 条件付きアクション ブロックの else 条件付きアクションブロックの開始を指定します。

Action コマンド	目的
action elseif	EEM アプレットがトリガーされたときに、else / if 条件付きアクション ブロックの else 条件付きアクション ブロックの開始を特定します。
action end	EEM アプレットがトリガーされたときに、if / else および while 条件付きアクション ブロックの条件付きアクション ブロック終了の ID を指定します。
action exit	EEM アプレットがトリガーされたときに、すぐに実行中のアプレット コンフィギュレーションを終了することを指定します。
action foreach	EEM アプレットがトリガーされたときに、デリミタをトークン化されたパターンとして使用した入力文字列の反復を指定します。
action gets	EEM アプレットがトリガーされたときに、同期アプレットのローカル TTY から入力を取得し、与えられた変数に値を格納します。
action if	EEM アプレットがトリガーされたときに、if 条件付きブロック開始の ID を指定します。
action if goto	EEM アプレットがトリガーされたときに、指定された条件が True であればアプレットが与えられたラベルにジャンプすることを指示します。
action increment	EEM アプレットがトリガーされたときに、変数の値を増加させます。
action info type interface-names	EEM アプレットがトリガーされたときに、インターフェイス名を取得するアクションを指定します。
action info type snmp getid	SNMP get オペレーション中に簡易ネットワーク管理プロトコル（SNMP）エンティティから各変数を取得します。
action info type snmp inform	EEM アプレットがトリガーされたときに、SNMP インフォーム要求を送信します。

Action コマンド	目的
action info type snmp oid	EEM アプレットがトリガーされたときに、SNMP get オペレーションのタイプ、および、SNMP set オペレーション中に取得するオブジェクトを指定します。
action info type snmp trap	EEM アプレットがトリガーされたときに、SNMP trap 要求を送信します。
action info type snmp var	SNMP オブジェクト ID（OID）の変数、およびその値を EEM アプレットから作成します。
action multiply	EEM アプレットがトリガーされたときに、変数値に指定された整数値を掛けるアクションを指定します。
action puts	EEM アプレットがトリガーされたときにデータを直接ローカル TTY に出力するアクションを有効にします。
action regexp	EEM アプレットがトリガーされたときに入力文字列の正規表現パターンと比較するアクションを指定します。
action set（EEM）	EEM アプレットがトリガーされたときに変数の値を設定するアクションを指定します。
action string compare	EEM アプレットがトリガーされたときに 2 個の等しくない文字列を比較するアクションを指定します。
action string equal	EEM アプレットがトリガーされたときに 2 個の文字列が等しいかどうかを検証するアクションを指定します。
action string first	EEM アプレットがトリガーされたときに string2 内に最初に string1 が見つかったインデックスを返すアクションを指定します。
action string index	EEM アプレットがトリガーされたときに与えられたインデックス値で指定される文字を返すアクションを指定します。
action string last	EEM アプレットがトリガーされたときに string 2 内に最後に string1 が見つかったインデックスを返すアクションを指定します。

Action コマンド	目的
action string length	EEM アプレットがトリガーされたときに文字列の文字数を返すアクションを指定します。
action string match	EEM アプレットがトリガーされたときに文字列がパターンに一致すれば、$_string_result に 1 を返すアクションを指定します。
action string range	EEM アプレットがトリガーされたときに文字列の文字の範囲を格納するアクションを指定します。
action string replace	EEM アプレットがトリガーされたときに指定された文字列の文字の範囲を置き換えることで新しい文字列を格納するアクションを指定します。
action string tolower	EEM アプレットがトリガーされたときに文字列の特定の範囲の文字を小文字で格納するアクションを指定します。
action string toupper	EEM アプレットがトリガーされたときに文字列の特定の範囲の文字を大文字で格納するアクションを指定します。
action string trim	EEM アプレットがトリガーされたときに文字列をトリムするアクションを指定します。
action string trimleft	EEM アプレットがトリガーされたときに、ある文字列の文字を別の文字列の左端からトリムするアクションを指定します。
action string trimright	EEM アプレットがトリガーされたときに、ある文字列の文字を別の文字列の右端からトリムするアクションを指定します。
action subtract	EEM アプレットがトリガーされたときに、別の値から変数の値を引きます。
action while	EEM アプレットがトリガーされたときに条件付きブロックのループの開始を特定するアクションを指定します。

イベント SNMP オブジェクトの設定例

次の例は、SET オペレーション、および、設定される値が $_snmp_value にありスクリプトで管理されることを示します。次の例は、oid とその値を、後で取得されるコンテキストとして格納します。

```
event manager applet snmp-object1
 description "APPLET SNMP-OBJ-1"
 event snmp-object oid 1.3.6.1.2.1.31.1.1.1.18 type string sync no skip no istable yes
default 0
 action 1 syslog msg "SNMP-OBJ1:TRIGGERED" facility "SNMP_OBJ"
 action 2 context save key myoid variable "_snmp_oid"
 action 3 context save key myvalue variable "_snmp_value"
```

EEM アプレットの説明の設定例

次に、簡易ネットワーク管理プロトコル（SNMP）のサンプリングによって実行される Embedded Event Manager（EEM）アプレットの説明を追加または変更する例を示します。

```
event manager applet test
 description "This applet looks for the word count in syslog messages"
 event syslog pattern "count"
 action 1 syslog msg hi
```

その他の参考資料

ここでは、Cisco IOS CLI を使用した EEM ポリシーの記述に関する関連資料について説明します。

関連資料

関連項目	マニュアル タイトル
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
Tcl を使用して Embedded Event Manager ポリシーを記述する	「Tcl を使用した Embedded Event Manager ポリシーの記述」の章
拡張オブジェクト トラッキングの設定	「Configuring Enhanced Object Tracking」の章

標準

標準	タイトル
新しい規格または変更された規格はサポートされていません。また、既存の規格に対するサポートに変更はありません。	--

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

Cisco IOS CLI を使用した EEM 4.0 ポリシーの記述の機能情報

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

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

表 *7 : Cisco IOS CLI* を使用した *EEM 4.0* ポリシーの記述の機能情報

機能名	リリース	機能情報
Embedded Event Manager 4.0	15.2(5)E1	この機能は、c2960cx プラットフォームにのみ導入され、サポートされています。

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