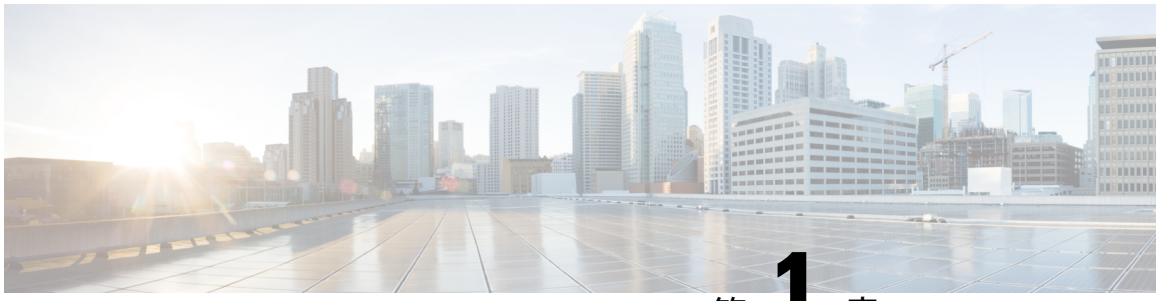
Cisco NCS 560 シリーズ ルータ（IOS XR リリース 6.6.x）システム モニタリング コンフィギュレーション ガイド

初版：2019 年 5 月 30 日

シスコシステムズ合同会社
〒107-6227 東京都港区赤坂9-7-1 ミッドタウン・タワー
http://www.cisco.com/jp
お問い合わせ先：シスコ コンタクトセンター
0120-092-255 （フリーコール、携帯・PHS含む）
電話受付時間：平日 10:00～12:00、13:00～17:00
http://www.cisco.com/jp/go/contactcenter/

目 次



第 1 章

システム ロギングの実装

このモジュールでは、ロギング サービスをルータに実装する必要があるタスクを説明します。

Cisco IOS XR ソフトウェアには基本ロギング サービスが用意されています。ロギング サービスでは、システム ロギング（syslog）メッセージモニタリングおよびトラブルシューティングのロギング情報を収集し、取得したロギング情報のタイプを選択できます。

システム ロギング実装の機能履歴

リリース	変更内容
リリース 6.1.2	プラットフォーム自動モニタリング（PAM）ツールは、すべての Cisco IOS XR 64 ビット プラットフォームに導入されました。

システム ロギングの実装

システム ロギング（Syslog）は、システム ログ メッセージの送信に使用される標準アプリケーションです。ログ メッセージは、デバイスの正常性を示すか、発生した問題を指摘します。重大度に応じて通知メッセージを簡素化する場合もあります。IOS XR ルータは、syslog メッセージを syslog プロセスに送信します。デフォルトでは、syslog メッセージはコンソール端末に送信されます。しかし、syslog メッセージは、ロギング バッファ、syslog サーバ、端末回線などのコンソール以外の宛先に送信することができます。

syslog メッセージ形式

デフォルトでは、Cisco IOS XR ソフトウェアの syslog プロセスで生成される syslog メッセージの一般形式は、次のようになります。

```
node-id : timestamp : process-name [pid] : % message category -group -severity -message
 -code : message-text
```

次の表は、Cisco IOS XR ソフトウェアでの syslog メッセージの一般形式について説明しています。

表 *1 : syslog* メッセージの形式

フィールド	説明
node-id	syslogメッセージの生成元となるノードです。
timestamp	month day HH:MM:SS 形式のタイム スタンプです。メッセージが生成された日時を示します。 （注） タイム スタンプ形式は、service timestamps コマンドを使用して変更できます。
process-name	syslog メッセージを生成したプロセスのプロセスです。
size	syslog メッセージを生成したプロセスのプロセス ID（pid）です。
[pid]	syslog メッセージに関連付けられているメッセージ カテゴリ、グループ名、重大度、メッセージ コードです。
message-text	syslog メッセージを説明する文字列です。

syslog メッセージの重大度

コンソール端末、syslog サーバ、および端末回線などのロギング先の場合、syslog メッセージの重大度を指定することによって、ロギング先に送信されるメッセージの数を制限できます。ただし、ロギングバッファ宛先では、指定された重大度に関係なく、すべての重大度のsyslogメッセージが送信されます。この場合、重大度レベルは、**show logging** コマンドの出力に表示される syslog メッセージを、指定された値以下で制限するだけです。次の表では、severity 引数に指定できる重大度キーワードおよび対応する UNIX syslog 定義を、最も重大度の高いレベルから低いレベルの順に一覧で示します。

表 *2 : syslog* メッセージの重大度

重大度のキーワード	レベル	説明
emergencies	0	システムが使用不可
alert	1	即時処理が必要
critical	2	クリティカルな状態
errors	3	エラー状態
warnings	4	警告状態

重大度のキーワード	レベル	説明
notifications	5	正常だが注意を要する状態
informational	6	情報メッセージだけ
debugging	7	デバッグ メッセージ

システム ロギングの設定に関する前提条件

Network Operating Center（NOC）でシステム メッセージのロギングを設定するには、次の前提条件が必要です。

- 適切なタスク ID を含むタスク グループに関連付けられているユーザ グループに属している必要があります。このコマンド リファレンスには、各コマンドに必要なタスク ID が含まれます。ユーザ グループの割り当てが原因でコマンドを使用できないと考えられる場合、AAA 管理者に連絡してください。
- syslog サーバ ホストを syslog メッセージの受信先に設定するには、syslog サーバに接続できる必要があります。

システム ログ機能の設定

必要に応じてシステム ロギングを設定するには、この項のタスクを実行します。

ロギング バッファへのロギングの設定

Syslogメッセージは、ロギングバッファと呼ばれる内部循環バッファを含む複数の宛先に送信できます。logging buffered コマンドを使用して、syslog メッセージをロギング バッファに送信できます。

設定例

次の例に、syslogメッセージをロギングバッファに送信するための設定を示します。ロギングバッファのサイズは3000000バイトに設定されています。ロギングバッファのサイズのデフォルト値は 2097152 バイトです。

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging buffered 3000000
RP/0/RP0/CPU0:Router(config)# commit
```

リモート サーバへのロギングの設定

Syslog メッセージは、ロギング バッファ、syslog サーバ、端末回線などのコンソール以外の宛先に送信することができます。logging コマンドを使用して syslog サーバの IP アドレスまたはホスト名を指定することにより、syslog メッセージを外部の syslog サーバに送信できます。また、logging facility コマンドを使用して、syslog メッセージが送信される syslog ファシリティを設定できます。

次の表に、syslog サーバに送信される syslog メッセージの管理に役立つ、Cisco IOS XR ソフトウェアでサポートされている機能を示します。

表 *3: Syslog* メッセージを管理するための機能

機能	説明
UNIX システム ログ機能	Facility は、ログ メッセージを送信したアプリケーションまたはプロセスを記述するために UNIX が使用する識別子です。logging facility コマンドを使用して、syslog メッセージが送信される syslog ファシリティを設定できます。
ホスト名プレフィックス ロギング	Cisco IOS XR ソフトウェアは、ホスト名のプレフィックス ロギングをサポートしています。イネーブルにすると、ホスト名プレフィックス ロギングでは、ルータから syslog サーバに送信される syslog メッセージにホスト名プレフィックスを追加します。ホスト名プレフィックスを使用して、さまざまなネットワーキング デバイスから特定の syslog サーバに送信されるメッセージを分類することができます。syslog サーバに送信される syslog メッセージにホスト名プレフィックスを追加するには、logging hostname コマンドを使用します。
syslog 送信元アドレス ロギング	デフォルトでは、syslog サーバに送信された syslog メッセージには、ルータから出るために使用するインターフェイスの IP アドレスが含まれています。syslog メッセージがどのインターフェイスを使用してルータを終了するかに関係なく、すべての syslog メッセージに同じ IP アドレスが含まれるように設定するには、logging source-interface コマンドを使用します。

設定例

次の例に、syslog メッセージを外部の syslog サーバに送信するための設定を示します。IP アドレス 10.3.32.154 は syslog サーバとして設定され、logging trap コマンドは重大度に基づいて syslog サーバに送信される syslog メッセージを制限するために使用されています。

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging 10.3.32.154
(Optional) RP/0/RP0/CPU0:Router(config)# logging 10.3.32.155 vrf vrfa
RP/0/RP0/CPU0:Router(config)# logging trap warnings
RP/0/RP0/CPU0:Router(config)# logging facility kern (optional)
RP/0/RP0/CPU0:Router(config)# logging hostnameprefix 123.12.35.7 (optional)
```

```
RP/0/RP0/CPU0:Router(config)# logging source-interface HundredGigE 0/0/1/0 (optional)
RP/0/RP0/CPU0:Router(config)# commit
```

関連項目

- ロギング バッファへのロギングの設定 （3 ページ）
- 端末回線へのロギングの設定 （5 ページ）

端末回線へのロギングの設定

デフォルトでは、syslogメッセージはコンソール端末に送信されます。しかし、syslogメッセージは、コンソール以外の端末回線に送信することもできます。logging monitor コマンドを使用して、syslog メッセージをロギング バッファに送信できます。

設定例

次の例に、syslog メッセージをコンソール以外の端末回線に送信するための設定を示します。この例では、重大度レベルが critical に設定されています。terminal monitor コマンドは、ターミナル セッション中に syslog メッセージを表示するように設定されています。デフォルトの重大度は debugging です。

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging monitor critical
RP/0/RP0/CPU0:Router(config)# commit
RP/0/RP0/CPU0:Router# terminal monitor
```

コンソール端末へのロギングの変更

デフォルトでは、syslog メッセージはコンソール端末に送信されます。コンソール端末への syslog メッセージのロギングを変更できます

設定例

次に、コンソール端末への syslog メッセージのロギングを変更する例を示します。

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging console alerts
RP/0/RP0/CPU0:Router(config)# commit
```

タイム スタンプ形式の変更

デフォルトでは、syslog メッセージのタイム スタンプが有効になっています。タイム スタンプは、month day HH:MM:SS の形式で生成され、メッセージが生成された日時を示します。

設定例

次の例では、syslogメッセージおよびデバッグ メッセージのタイムスタンプを変更する方法を示します。

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# service timestamps log datetime localtime msec or service
 timestamps log uptime
RP/0/RP0/CPU0:Router(config)# service timestamps debug datetime msec show-timezone or
```

```
service timestamps debug uptime
RP/0/RP0/CPU0:Router(config)# commit
```

重複 syslog メッセージの抑制

特に大規模ネットワークで、重複メッセージが作成されないようにすると、メッセージクラッターを減らし、ログの解釈作業を効率化できます。重複メッセージの抑制機能により、ロギング ヒストリと syslog ファイルの両方で、重複するイベント メッセージを大幅に削減できます。

設定例

次の例に、重複する syslog メッセージが連続してロギングされないようにする方法を示します。

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging suppress duplicates
RP/0/RP0/CPU0:Router(config)# commit
```

ローカル ストレージ デバイスへのシステム ロギング メッセージのアーカイブ

syslog メッセージは、ハード ディスクやフラッシュ ディスクなどのローカル ストレージ デバイスのアーカイブに保存することもできます。メッセージは重大度に基づいて保存できます。アーカイブのサイズ、メッセージが追加される頻度（日次または週次）、アーカイブに保存するメッセージの週合計などの属性を指定できます。logging archive コマンドを使用して、ロギング アーカイブを作成し、ロギング メッセージの収集および保存方法を指定できます。

この表では、ロギング アーカイブ サブモードでアーカイブ属性を指定するために使用されるコマンドを一覧で示します。

表 *4: syslog* アーカイブ属性を設定するために使用するコマンド

機能	説明
archive-length weeks	アーカイブでアーカイブ ログが保持される最長週数を指定します。保存期間がこの週数を超えるログは、自動的にアーカイブから削除されます。
archive-size size	ストレージ デバイス上にある syslog アーカイブの最大合計サイズを指定します。このサイズを超過すると、新しいログ用の領域を確保するため、アーカイブ内の最も古いファイルが削除されます。

機能	説明
device {disk0 | disk1 | harddisk}	syslog がアーカイブされるローカル ストレージ デバイスを指定します。デフォルトでは、ログは device/**var/log** ディレクトリに作成されます。デバイスが設定されていない場合は、他のすべてのロギング アーカイブ設定が拒否されます。フラッシュ ディスクよりもハードディスクの容量の方が大きいため、syslog はハードディスクにアーカイブすることを推奨します。
file-size size	アーカイブにある 1 つのログ ファイルの最大ファイル サイズ（メガバイト単位）を指定します。この制限サイズに達すると、自動的に新しいファイルが作成され、1 つずつ順に大きいシリアル番号が付与されます。
frequency {daily | weekly}	ログが収集される頻度を日次または週次で指定します。
severity severity	アーカイブするログ メッセージの最小重大度を指定します。設定されたこのレベル以上の syslog メッセージがすべてアーカイブされ、これらのレベルより小さいメッセージは除外されます。

設定例

次に、ローカル ストレージ デバイス上のアーカイブに syslog メッセージを保存する例を示します。

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging archive
RP/0/RP0/CPU0:Router(config-logging-arch)# device disk1
RP/0/RP0/CPU0:Router(config-logging-arch)# frequency weekly
RP/0/RP0/CPU0:Router(config-logging-arch)# severity warnings
RP/0/RP0/CPU0:Router(config-logging-arch)# archive-length 6
RP/0/RP0/CPU0:Router(config-logging-arch)# archive-size 50
RP/0/RP0/CPU0:Router(config-logging-arch)# file-size 10
RP/0/RP0/CPU0:Router(config)# commit
```

プラットフォーム自動モニタリング

プラットフォーム自動モニタリング（PAM）は、プロセス クラッシュ、メモリ リーク、CPU ホッグ、トレースバック、syslog、ディスク使用率などの問題をモニタするために Cisco IOS XR ソフトウェア イメージ に統合されたシステム モニタリング ツールです。PAM はすべての Cisco IOS XR 64 ビット プラットフォームでデフォルトで有効になっています。PAM ツールは、これらのシステムの問題を検出すると、問題のトラブルシューティングに必要なデータを

収集し、問題を示すSyslogメッセージを生成します。自動収集されたトラブルシューティング情報は、harddisk:/cisco_support/ または /misc/disk1/cisco_support/ ディレクトリに個別のファイルとして保存されます。

PAM イベント

PAM は、プロセスのクラッシュ、トレースバック、潜在的なメモリ リーク、CPU ホッグ、またはフル ファイル システムを検出すると、自動的にログを収集し、これらのログ（該当する場合はコア ファイルとともに）を *.tgz* ファイルとして harddisk:/cisco_support/ または /misc/disk1/cisco_support/ ディレクトリに保存します。また、PAM は重大度が warning の syslog メッセージを生成し、それぞれの問題について言及します。

.tgz ファイルの形式は *PAM-<platform>-<PAM event>-<node-name>-<PAM process>-<YYYYMMDD>-<checksum>.tgz* です。たとえば、*PAM--crash-xr_0_RP0_CPU0-ipv4_rib-2016Aug16-210405.tgz* は、PAM がプロセスのクラッシュを検出した場合に収集されるファイルです。

PAM は、コア ファイルがデフォルトのアーカイブ フォルダ（harddisk:/ または /misc/disk1/）に保存されていることを前提としているため、コアアーカイブの場所を変更したり（例外ファイルパスを設定することによって）、PAM がイベントを検出した後に生成されたコア ファイルを削除したりしないでください。従わない場合、PAM はプロセスのクラッシュを検出しません。また、一度報告した後は、PAM は同じノード内の同じプロセスにおける同じ問題を再度報告しません。

ログの収集中に使用されるコマンドのリストについては、PAM ツールによって収集されるファイル （12 ページ） を参照してください。

以下の項では、PAM の主なイベントについて説明します。

クラッシュのモニタリング

PAMは、すべてのノードのプロセスクラッシュをリアルタイムでモニタします。以下は、PAMがプロセスのクラッシュを検出したときに生成されるサンプル syslog です。

```
RP/0/RP0/CPU0:Aug 16 21:04:06.442 : logger[69324]: %OS-SYSLOG-4-LOG_WARNING : PAM detected
 crash for ipv4_rib on 0_RP0_CPU0.
All necessary files for debug have been collected and saved at
0/RP0/CPU0 : harddisk:/cisco_support/PAM--crash-xr_0_RP0_CPU0-ipv4_rib-2016Aug16-210405.tgz

Please copy tgz file out of the router and send to Cisco support. This tgz file will be
 removed after 14 days.)
```

トレースバックのモニタリング

PAM は、すべてのノードのトレースバックをリアルタイムでモニタします。以下は、PAM がトレースバックを検出したときに生成されるサンプル syslog です。

```
RP/0/RP0/CPU0:Aug 16 21:42:42.320 : logger[66139]: %OS-SYSLOG-4-LOG_WARNING : PAM detected
 traceback for ipv4_rib on 0_RP0_CPU0.
All necessary files for debug have been collected and saved at
0/RP0/CPU0 :
```

```
harddisk:/cisco_support/PAM--traceback-xr_0_RP0_CPU0-ipv4_rib-2016Aug16-214242.tgz
Please copy tgz file out of the router and send to Cisco support. This tgz file will be
 removed after 14 days.)
```

メモリ使用率モニタリング

PAM は、すべてのノードのプロセス メモリ使用率をモニタします。PAM は、メモリ使用率の傾向をモニタし、収集されたデータに独自のアルゴリズムを適用することによって、潜在的なメモリ リークを検出します。デフォルトでは、すべてのノードで最大出力を 30 分間隔で定期的に収集します。

以下は、PAM が潜在的なメモリ リークを検出したときに生成されるサンプル syslog です。

```
RP/0/RP0/CPU0:Aug 17 05:13:32.684 : logger[67772]: %OS-SYSLOG-4-LOG_WARNING : PAM detected
 significant memory increase
(from 13.00MB at 2016/Aug/16/20:42:41 to 28.00MB at 2016/Aug/17/04:12:55) for
pam_memory_leaker on 0_RP0_CPU0.
All necessary files for debug have been collected and saved at
0/RP0/CPU0 :
harddisk:/cisco_support/PAM--memory_leak-xr_0_RP0_CPU0-pam_memory_leaker-2016Aug17-051332.tgz

(Please copy tgz file out of the router and send to Cisco support. This tgz file will
be removed after 14 days.)
```

CPU モニタリング

PAM は、すべてのノードの CPU 使用率を 30 分間隔で定期的にモニタします。PAM は、次のシナリオのいずれかで CPU ホッグを報告します。

- プロセスが常に多くの CPU を消費する場合（つまり、しきい値の 90% を超える場合）
- CPU 使用率の高い状態が 60 分よりも長く続く場合

以下は、PAM が CPU ホッグを検出したときに生成されるサンプル syslog です。

```
RP/0/RP0/CPU0:Aug 16 00:56:00.819 : logger[68245]: %OS-SYSLOG-4-LOG_WARNING : PAM detected
 CPU hog for cpu_hogger on 0_RP0_CPU0.
All necessary files for debug have been collected and saved at 0/RP0/CPU0 :
harddisk:/cisco_support/PAM--cpu_hog-xr_0_RP0_CPU0-cpu_hogger-2016Aug16-005600.tgz
(Please copy tgz file out of the router and send to Cisco support. This tgz file will
be removed after 14 days.)
RP/0/RP0/CPU0:Jun 21 15:33:54.517 : logger[69042]: %OS-SYSLOG-1-LOG_ALERT : PAM detected
 ifmgr is hogging CPU on 0_RP0_CPU0!
```

ファイル システム モニタリング

PAM は、すべてのノードのディスク使用率を 30 分間隔で定期的にモニタします。以下は、PAM がシステム ファイルがいっぱいであることを検出したときに生成されるサンプル syslog です。

```
RP/0/RP0/CPU0:Jun 20 13:59:04.986 : logger[66125]: %OS-SYSLOG-4-LOG_WARNING : PAM detected
 /misc/config is full on 0_1_CPU0
(please clean up to avoid any fault caused by this). All necessary files for debug have
```

```
 been collected and saved at
0/RP0/CPU0 : harddisk:/cisco_support/PAM--disk_usage-xr_0_1_CPU0-2016Jun20-135904.tgz
(Please copy tgz file out of the router and send to Cisco support. This tgz file will
be removed after 14 days.)
```

PAM の無効化と再有効化

PAM ツールは、monitor_cpu.pl、monitor_crash.pl、monitor_show_show_logging.pl の 3 つのモニタリング プロセスで構成されています。

PAM を無効化または再度有効化する前に、次のオプションを使用して、PAM がルータにインストールされているかどうかを確認します。

- Cisco IOS XR コマンドライン インターフェイスから：

```
Router# show processes pam_manager location all
Tue Jun 14 17:58:42.791 UTC
node:      node0_RP0_CPU0
                  Job Id: 317
                     PID: 14070
         Executable path:
/opt/cisco/XR/packages/iosxr-infra.rp-6.1.1.17I/bin/pam_manager
              Instance #: 1
              Version ID: 00.00.0000
                 Respawn: ON
           Respawn count: 4
            Last started: Mon Jun 13 23:08:43 2016
           Process state: Run
           Package state: Normal
                    core: MAINMEM
                Max. core: 0
                   Level: 999
               Placement: None
             startup_path:
/opt/cisco/XR/packages/iosxr-infra.rp-6.1.1.17I/startup/pam_manager.startup
                   Ready: 0.166s
        Process cpu time: 0.200 user, 0.310 kernel, 0.510 total
JID    TID  Stack  pri  state        NAME             rt_pri
317    14070    0K 20 Sleeping     pam_manager         0
317    14071    0K 20 Sleeping     lwm_debug_threa     0
317    14076    0K 20 Sleeping     pam_manager         0
317    14077    0K 20 Sleeping     lwm_service_thr     0
317    14078    0K 20 Sleeping     qsm_service_thr     0
317    14080    0K 20 Sleeping     pam_manager         0
```

- ルータ シェル プロンプトから：

```
Router# run ps auxw|egrep perl
Tue Jun 14 18:00:25.514 UTC
root     14324  0.0  0.2  84676 34556 ?  S Jun13   0:40 /usr/bin/perl
/pkg/opt/cisco/pam//monitor_cpu.pl
root     14414  0.0  0.1  65404 14620 ?  S Jun13   0:00 /usr/bin/perl
/pkg/opt/cisco/pam//monitor_crash.pl
```

PAM の無効化

PAM エージェントをシャットダウンするには、XR EXEC モードから次のコマンドを実行します。

```
For local RP:
Router# process shutdown pam_manager

For all RPs:
Router# process shutdown pam_manager location all
```

PAM の再有効化

pam_manager は必須プロセスではないため、手動で無効にした場合は自動的に再起動されません（システム リロードの場合を除く）。PAM エージェントを再起動するには、XR EXEC モードから次のコマンドを実行します。

```
For local RP:
Router# process start pam_manager

For all RPs:
Router# process start pam_manager location all
```

（注） すべてのロケーションで PAM を開始するには、**process start pam_manager** コマンドの **location all** オプションを使用して、すべてのノードで *pam_manager* プロセスを再起動する必要があります。

PAM でのデータ アーカイブ

任意の時点で、PAM は 200 MB を超えるハードディスクのスペースを占有しません。200 MB より多く必要な場合、PAM は古いファイルをアーカイブし、ログを自動的にローテーションします。

PAM は、CPU またはメモリ使用量（**top -b -n1** コマンドを使用）を定期的に 30 分間隔で収集します。ファイルは、<node name>.log というファイル名で harddisk:/cisco_support/ ディレクトリに保存されます（例：harddisk:/cisco_support/xr-0_RP0_CPU0.log）。ファイル サイズが 15MB の制限を超えると、ファイルは .tgz ファイルにアーカイブ（圧縮）され、最大 2 回ローテーションされます（つまり、.tgz ファイルは 2 つしか保持されません）。.tgz ファイルの最大ローテーション数は 3 です。また、ノードがリロードされると、古いファイル（ASCII データ）がアーカイブされ、ローテーションされます。たとえば、RP0 がリロードされると、xr-0_RP0_CPU0.log がアーカイブされます。

PAM によって生成されたコア ファイルは手動で削除しないでください。コア ファイルには、*<process name>_pid.by_user.<yyyymmdd>-<hhmmss>.<node>.<checksum>.core.gz* と名前が付けられます。

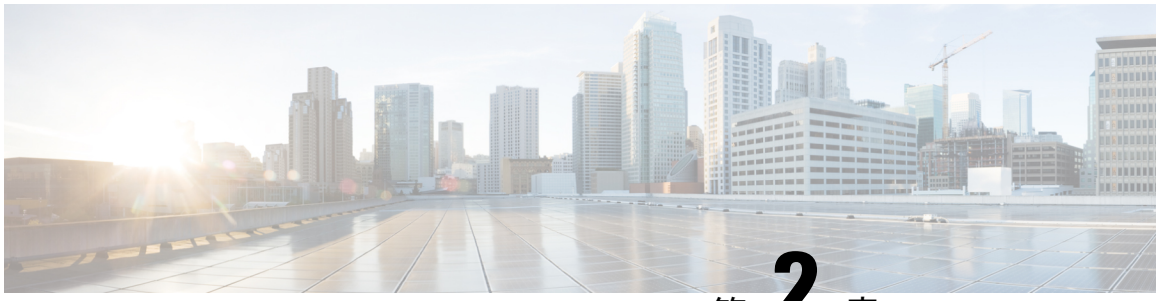
PAM ツールによって収集されるファイル

次の表は、さまざまな PAM イベントと、各イベントに対し PAM によって収集される各コマンドとファイルを示しています。

シスコ テクニカル サポートでサービス リクエスト（SR）を発行するときに、個々の *.tgz* ファイルを添付できます。

イベント名	PAM によって収集されるコマンドおよびファイル
プロセスのクラッシュ	• **show install active** • **show platform** • **show version** • コア（gz）ファイル • core.txt ファイル
プロセスのトレースバック	• **show dll** • **show install active** • **show logging** • **show platform** • **show version**
メモリ リーク	• **show install active** • **show platform** • **show version** • コア（gz）ファイル • 実行中のダンプコア • 連続メモリ使用量のスナップショット
ロギング イベントの表示	• **show install active** • **show logging** • **show platform** • **show version** • コア（gz）ファイル • core.txt ファイル

イベント名	PAM によって収集されるコマンドおよびファイル
CPU ホッグ	• **follow process** • **pstack** • **show dll** • **show install active** • **show platform** • **show version** • **top -H** • コア（gz）ファイル • CPU 使用率のスナップショット
ディスク使用量	• **show install active** • **show platform** • **show version** • コンソール ログ • コア（gz）ファイル • ディスク使用率のスナップショット



第 2 章

アラーム ログ相関の実装

このモジュールでは、アラーム ログ相関の設定に関する概念とタスクについて説明します。アラーム ログ相関は、さまざまなアプリケーションおよびシステム サーバで生成されたメッセージのグループ化機能とフィルタリング機能、およびルータ上のルート メッセージの分離機能を含めるように、システム ロギングを拡張します。

アラーム ログ相関の実装

アラーム ログ相関は、さまざまなアプリケーションおよびシステム サーバで生成されたメッセージのグループ化機能とフィルタリング機能、およびルータ上のルート メッセージの分離機能を含めるように、システム ロギングを拡張します。このモジュールでは、アラーム ログ相関の設定とアラーム ログのモニタリングに関連する概念とタスクについて説明します。

アラーム ログ相関の実装に関する前提条件

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

アラーム ログ相関の実装に関する情報

アラーム ロギングおよびデバッグ イベント管理システム

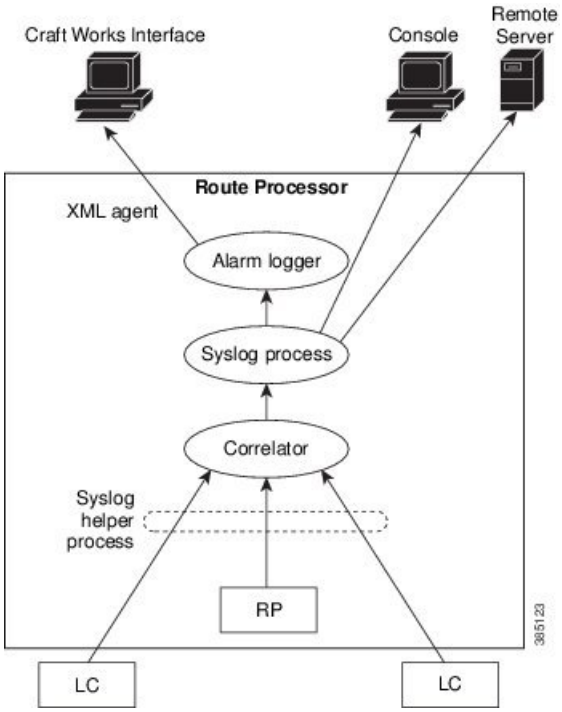
Cisco IOS XR ソフトウェアのアラーム ロギングおよびデバッグ イベント管理システム（ALDEMS）は、システム サーバおよびアプリケーションから転送されるアラーム メッセージをモニタリングし、格納するために使用されます。また、ALDEMS は、単一の根本原因のために転送されたアラーム メッセージ同士を相関します。

ALDEMS は、Cisco IOS XR ソフトウェアの基本的なロギング機能およびモニタリング機能を拡大して、数百のラインカードと数千のインターフェイスを備える可能性のある高度に分散化されたシステムに必要とされるレベルのアラームとイベント管理を実現します。

Cisco IOS XR ソフトウェアは、システムのノード全体にロギング アプリケーションを分散することによって、この必要なレベルのアラーム管理およびイベント管理を実現します。

図 1 : ALDEMS コンポーネント通信 （16 ページ）に、ALDEMS を構成するコンポーネント間の関係を示します。

図 *1 : ALDEMS* コンポーネント通信



コリレータ

コリレータは、ルータ上のノード全体に分散されたシステム ロギング（syslog）ヘルパープロセスからメッセージを受け取り、syslog メッセージを syslog プロセスに転送します。ロギング相関ルールが設定されている場合は、コリレータはルールで指定されているメッセージと一致するメッセージを検索して、そのメッセージをキャプチャします。コリレータは、一致を検出すると、ルールに指定されているタイムアウト間隔に対応するタイマーを開始させます。コリレータは、タイマーの期限が切れるまで、ルール内のメッセージとの一致を検索し続けます。根本原因メッセージを受信した場合は、相関が実行されます。受信しなかった場合は、キャプチャされたメッセージがすべてsyslogに転送されます。相関が実行された場合、相関メッセージはロギング相関バッファに保存されます。相関メッセージの各セットには、コリレータにより相関 ID がタグ付けされます。

システム ロギング プロセス

アラーム ロガーは、ルータに転送されるシステム ロギング メッセージの最終宛先です。アラーム ロガーには、ロギング イベント バッファ内のアラーム メッセージが保存されます。ロギング イベント バッファは循環バッファであるため、いっぱいになるとバッファ内の最も古いメッセージが上書きされます。

アラーム ロガー

アラーム ロガーは、ルータに転送されるシステム ロギング メッセージの最終宛先です。アラーム ロガーには、ロギング イベント バッファ内のアラーム メッセージが保存されます。ロギング イベント バッファは循環バッファであるため、いっぱいになるとバッファ内の最も古いメッセージが上書きされます。

（注） アラームは、ロギング イベント バッファ内で優先順位付けされます。アラーム レコードを上書きする必要がある場合は、ロギング イベント バッファは、最初に非バイステート アラーム、次に CLEAR ステートのバイステート アラーム、最後に SET ステートのバイステート アラームの順序でメッセージを上書きしていきます。

SET ステートのバイステート アラームにより発行されたメッセージでテーブルがいっぱいになると、（着信時刻ではなくメッセージのタイム スタンプ基準で）一番古いバイステート アラームがその他のメッセージよりも先に上書きされます。したがって、メモリ消費量が要件内に収まるように、ロギング イベント バッファおよびロギング相関バッファのバッファ サイズを調整する必要があります。

テーブルフル アラームは、ロギング イベント バッファが一巡するたびに生成されます。しきい値超過通知は、ロギング イベント バッファが容量のしきい値に到達するたびに生成されます。

ロギング イベント バッファに保存されたメッセージに対してクライアントからクエリーを実行して、特定の条件に一致するレコードを特定できます。アラーム ロギング メカニズムにより、各アラーム メッセージには連番で一意の ID が割り当てられます。

アラーム ログ相関の設定

必要に応じてアラーム ログ相関を設定するには、この項の設定タスクを実行します。

ロギング相関ルールの設定

ロギング相関を使用して、システム パフォーマンスに影響を及ぼすイベントの最上位ルート メッセージを分離できます。相関ルールが設定されている場合、セカンダリ（非根本原因）メッセージを生成する共通ルート イベントを分離して syslog に送信することで、セカンダリ メッセージを抑制できます。オペレータは、ロギング コリレータ バッファから相関メッセージをすべて取得して、発生した相関イベントを表示できます。相関ルールをルータ全体に適用した場合、メッセージのコンテキストまたはロケーション設定にかかわらず、設定されたルールの原因値に一致するメッセージだけで相関が発生します。相関ルールを特定のコンテキスト

またはロケーションのセットに適用した場合、ルールで設定されている原因値にするメッセージ、およびこれらのコンテキストまたはロケーションのいずれか1つに一致するメッセージだけで相関が発生します。

相関ルールが設定され適用された場合、コリレータにより、ルールの指定に従ってメッセージの一致が検索されます。タイムアウトは、一致が見つかったらメッセージ検索の時間間隔を指定するように設定できます。タイムアウトは、コリレータが相関ルールで指定されたアラームメッセージをキャプチャしたときに開始されます。

設定例

次に、ロギング相関ルールを設定して適用する例を示します。この例では、タイムアウトは60000 ミリ秒として設定されています。

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging correlator rule rule1 type stateful
RP/0/RP0/CPU0:Router(config-corr-rule-st)# timeout 60000
RP/0/RP0/CPU0:Router(config)# logging correlator apply-rule rule1
RP/0/RP0/CPU0:Router(config-corr-apply-rule)# all-of-router
or
RP/0/RP0/CPU0:Router(config-corr-apply-rule)# location 0/1/CPU0
or
RP/0/RP0/CPU0:Router(config-corr-apply-rule)# context HundredGigE_0_0_1_0
RP/0/RP0/CPU0:Router(config)# commit
```

ロギング相関ルール セットの設定

ロギング相関ルール セットを設定し、複数の相関ルールを組み込むことができます。

設定例

次に、複数の相関ルール用にロギング相関ルールセットを設定して適用する例を示します。ロギング相関ルール セットは、ルータ全体または特定のコンテキストまたは場所に適用できます。

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging correlator ruleset ruleset1
RP/0/RP0/CPU0:Router(config-corr-ruleset)# rulename stateful_rule1
RP/0/RP0/CPU0:Router(config-corr-ruleset)# rulename stateful_rule2
RP/0/RP0/CPU0:Router(config)# logging correlator apply ruleset ruleset1
RP/0/RP0/CPU0:Router(config-corr-apply-rule)# all-of-router
or
RP/0/RP0/CPU0:Router(config-corr-apply-rule)# location 0/2/CPU0
or
RP/0/RP0/CPU0:Router(config-corr-apply-rule)# context HundredGigE_0_0_1_0
RP/0/RP0/CPU0:Router(config)# commit
```

根本原因アラームと非根本原因アラームの相関

根本原因メッセージは、相関ルールに設定された最初のメッセージ（カテゴリ、グループ、およびコードの3つが設定されたもの）により定義されます。根本原因メッセージは、必ずsyslogプロセスに転送されます。根本原因を 1 つ以上の非根本原因アラームと相関させ、それらをルールの一部として設定することができます。

設定例

次の例では、根本原因を1つ以上の非根本原因アラームと相関させ、それらをルールに設定する方法を示します。

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging correlator rule rule_stateful type stateful
RP/0/RP0/CPU0:Router(config-corr-rule-st)# rootcause CAT_BI_1 GROUP_BI_1 CODE_BI_1
RP/0/RP0/CPU0:Router(config-corr-rule-st)# nonrootcause
RP/0/RP0/CPU0:Router(config-corr-rule-st-nonrc)# alarm CAT_BI_2 GROUP_BI_2 CODE_BI_2
RP/0/RP0/CPU0:Router(config)# commit
```

階層的な相関ルール フラグの設定

階層的な相関は、1つのアラームがあるルールの根本原因であり、かつ別のルールの非根本原因でもある場合、およびアラームが生成され、結果として両方のルールに関連する正常な相関となった場合に発生します。非根本原因アラームに起こったことが、相関根本原因アラームの動作を決定します。これらの階層に関連するステートフル動作を制御する必要があるケース、および非バイステートアラームの再配置および再発行などのフラグを実装する必要があるケースがあります。階層的な相関および相関フラグの詳細については、を参照してください。階層的な相関 （24 ページ）

設定例

次に、階層的な相関ルールのフラグを設定する例を示します。

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging correlator rule rule_nonstateful type nonstateful
RP/0/RP0/CPU0:Router(config-corr-rule-st)# reissue-nonbistate
RP/0/RP0/CPU0:Router(config-corr-rule-st)# reparent
RP/0/RP0/CPU0:Router(config)# commit
RP/0/RP0/CPU0:Router# show logging correlator rule all (optional)
```

ロギング抑制ルールの設定

アラームロギング抑止機能を使用すると、抑止するアラームのタイプを指定するロギング抑止ルールを定義することで、アラームのロギングを抑制できます。ロギング抑制ルールでは、すべてのタイプのアラーム、または特定のメッセージカテゴリ、グループ名、およびメッセージコードを持つアラームを指定できます。ロギング抑制ルールは、ルータ上のすべてのロケーションから発生するアラームに対して適用するか、または特定のノードから発生するアラームに対して適用できます。

設定例

次の例に、ロギング抑制ルールの設定方法を示します。

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging suppress rule infobistate
RP/0/RP0/CPU0:Router(config-suppr-rule)# alarm MBGL COMMIT SUCCEEDED
RP/0/RP0/CPU0:Router(config)# logging suppress apply rule infobistate
RP/0/RP0/CPU0:Router(config-suppr-apply-rule)# all-of-router
RP/0/RP0/CPU0:Router(config)# commit
```

ロギング イベント バッファ設定の変更

アラーム ロガーには、ロギング イベント バッファ内のアラーム メッセージが保存されます。ロギング イベント バッファは、バッファがいっぱいになったときに最も古いメッセージを上書きします。ロギング イベント バッファの設定は、ネットワークのパフォーマンスに影響するユーザアクティビティ、ネットワーク イベント、システム設定イベントの変更、またはネットワーク モニタリング要件の変更に対応して調整できます。適切な設定は、システムの設定および要件に応じて異なります。しきい値超過通知は、ロギング イベント バッファが容量のしきい値に到達するたびに生成されます。

設定例

次の例は、ロギング イベント バッファのサイズ、しきい値、およびアラーム フィルタの設定を示しています。

```
RP/0/RP0/CPU0:Router# configure terminal
RP/0/RP0/CPU0:Router(config)# logging events buffer-size 50000
RP/0/RP0/CPU0:Router(config)# logging events threshold 85
RP/0/RP0/CPU0:Router(config)# logging events level warnings
RP/0/RP0/CPU0:Router(config)# commit
```

ロギング相関バッファ設定の変更

相関が実行された場合、相関メッセージはロギング相関バッファに保存されます。ロギング相関バッファのサイズは、予想される着信相関メッセージを収容できるように調整できます。レコードを指定してバッファからレコードを削除したり、バッファにあるレコードをすべてクリアしたりできます。

設定例

次の例では、相関バッファのサイズを設定し、バッファからレコードを削除します。

```
RP/0/RP0/CPU0:Router# configure terminal
RP/0/RP0/CPU0:Router(config)# logging correlator buffer-size 100000
RP/0/RP0/CPU0:Router(config)# exit
RP/0/RP0/CPU0:Router# clear logging correlator delete 48 49 50 (optional)
RP/0/RP0/CPU0:Router# clear logging correlator delete all-in-buffer (optional)
```

バイステート アラームのアラーム ソース ロケーション表示フィールドのイネーブル化

バイステート アラームは、システムのハードウェアに関連付けられている状態の変化によって生成されます。バイステート アラーム メッセージの形式は、syslog メッセージに似ています。オプションで、出力に実際のアラーム ソースのロケーションが含まれるように設定できます。このアラーム ソースは、アラームをロギングしたプロセスとは異なる場合があります。バイステート アラームの詳細については、を参照してください。バイステート アラーム （23 ページ）

設定例

次の例に、バイステート アラームのアラーム ソース ロケーション表示フィールドをイネーブルにする方法を示します。

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# logging events display-location
RP/0/RP0/CPU0:Router(config)# commit
```

SNMP 相関ルールの設定

大規模システムでは、定期的な間隔で出力される多数の SNMP トラップに遭遇する状況になる可能性があります。これらのトラップは、Cisco IOS XR によるトラップの処理時間を延長させます。また、追加のトラップはトラブルシューティングを遅くし、モニタリング システムおよびオペレータの作業負荷を増大します。SNMP アラーム相関は、既存の syslog コリレータから相関機能の一般的な部分を抽出するのに役立ちます。相関ルールを設定して、SNMP トラップの相関ルールを定義し、特定のトラップ宛先に適用することができます。

設定例

次に、SNMP トラップの相関ルールを設定および適用する例を示します。SNMP コリレータのバッファ サイズも 600 バイトに設定されています。バッファ サイズのデフォルト値は 64KB です。

```
RP/0/RP0/CPU0:Router# configure terminal
RP/0/RP0/CPU0:Router(config)# snmp-server correlator buffer-size 600 (optional)
RP/0/RP0/CPU0:Router(config)# snmp-server correlator rule test rootcause A varbind A1
value regex RA1  nonrootcause  trap B varbind B1 index regex RB1
RP/0/RP0/CPU0:Router(config)# snmp-server correlator apply rule test host ipv4 address
1.2.3.4
RP/0/RP0/CPU0:Router(config)# commit
```

SNMP 相関ルールセットの設定

SNMP 相関ルール セットを設定し、複数の SNMP 相関ルールを組み込むことができます。

設定例

次に、複数のルールを 1 つのグループにグループ化できるルールセットを設定する例を示します。指定したグループをホストのセットまたはすべてのホストに適用できます。

```
RP/0/RP0/CPU0:Router# configure terminal
RP/0/RP0/CPU0:Router(config)# snmp-server correlator ruleset rule1 rulename rule2
RP/0/RP0/CPU0:Router(config)# snmp-server correlator apply ruleset rule1 host ipv4 address
 1.2.3.4
RP/0/RP0/CPU0:Router(config)# commit
```

アラーム ロギング相関の詳細

アラーム ロギング相関を使用して、システム パフォーマンスに影響を及ぼすイベントの最上位ルート メッセージを分離できます。たとえば、ライン カードの活性挿抜（OIR）を示す元のメッセージが分離され、根本原因メッセージのみ表示され、同じイベントに関連するすべての後続メッセージが相関になる場合があります。相関ルールが設定されている場合、セカンダリ（非根本原因）メッセージを生成する共通ルート イベントを分離して syslog に送信することで、セカンダリ メッセージを抑制できます。オペレータは、ロギング コリレータ バッファから相関メッセージをすべて取得して、発生した相関イベントを表示できます。

相関ルール

相関ルールを設定して、システム アラームを生成する可能性のあるルート メッセージを分離できます。相関ルールは、アラームロギングおよびデバッグイベント管理システム（ALDEMS）に、不要なメッセージの蓄積に起因する不要なストレスを与えないようにします。各相関ルールはメッセージIDに依存し、メッセージカテゴリ、メッセージグループ名、メッセージコードで構成されます。コリレータプロセスは、メッセージをスキャンしてメッセージの発生を検出します。コリレータがルートメッセージを受信すると、コリレータはそのメッセージをロギング コリレータ バッファに保存し、さらにRPのsyslogプロセスに転送します。その後、syslogプロセスにより、そのルートメッセージはロギング イベント バッファ内のアラーム ロガーに転送され、保存されます。また、ネットワーク デバイスの構成に応じて、ルート メッセージが syslog プロセスからコンソール、リモート ターミナル、リモート サーバ、障害管理システム、および簡易ネットワーク管理プロトコル（SNMP）エージェントなどの宛先に転送される場合もあります。同一の条件に一致する後続のメッセージ（別に発生したルートメッセージを含む）は、ロギング相関バッファに保存され、ルータの syslog プロセスに転送されます。

メッセージが複数の相関ルールに一致する場合、一致したルールすべてが適用され、そのメッセージはロギング コリレータ バッファ内の一致する相関キューすべての一部になります。次のメッセージ フィールドを使用して、ロギング相関ルールのメッセージが定義されます。

- メッセージ カテゴリ
- メッセージ グループ
- メッセージ コード

いずれのメッセージ フィールドでも、ワイルドカードを使用してより幅広いメッセージ セットをカバーできます。

根本原因メッセージ、ステートフル相関および非ステートフル相関を分離するために、ルールには 2 つのタイプの相関が設定されています。非ステートフル相関は、発生後に固定されます。抑制された非根本原因アラームがsyslogプロセスに転送されることはありません。非根本原因アラームはすべて相関バッファにバッファされた状態で残ります。ステートフル相関は、バイステート根本原因アラームがクリアされると、相関発生後に変更される場合があります。アラームがクリアされると、すべての相関された非根本原因アラームはsyslogに送信され、相関バッファからは削除されます。ステートフル相関は、疑われる根本原因がすでに存在しないにもかかわらず、存在し続けている非根本原因状態を検出する場合に役立ちます。

アラーム重大度とフィルタリング

フィルタ設定を使用して、重大度に基づいて情報を表示できます。アラーム フィルタ表示は、アラーム、レコード数、現在のログ サイズ、最大ログ サイズのレポートに使用される重大度の設定を示します。

アラームは、次の表に示されている重大度に応じてフィルタリングできます。

表 *5:* イベント ロギングのアラーム重大度

重大度	システムの状態
0	緊急
1	アラート
2	クリティカル
3	エラー
4	警告
5	通知
6	情報

バイステート アラーム

バイステート アラームは、アクティブから非アクティブへのインターフェイス ステートの変化、ライン カードの活性挿抜（OIR）、またはコンポーネントの温度の変化など、システム ハードウェアに関連するステート変更によって生成されます。デフォルトでは、バイステート アラーム イベントはロギング イベント バッファにレポートされます。情報メッセージおよびデバッグ メッセージはレポートされません。

Cisco IOS XR ソフトウェアには、アラームをリセットおよびクリアする機能があります。システムのアラームをモニタリングする必要のあるクライアントは、モニタリング対象のアラームの状態が変化したときに非同期通知を受信するためのアラーム ロギング メカニズムを登録できます。

バイステート アラーム通知により、次のことがわかります。

- 発信元 ID。発生またはクリアされるアラームの発信元であるリソースを一意に特定します。このリソースは、インターフェイス、ラインカード、または特定用途向け集積回路（ASIC）などです。発信元 ID は、ロケーション、ジョブ ID、メッセージ グループ、メッセージ コンテキストの一意の組み合わせです。

デフォルトでは、バイステート アラーム メッセージの一般形式はすべての syslog メッセージで同一です。

node-id:*timestamp* : *process-name* [*pid*] : %*category-group-severity-code* : *message-text*

次に、バイステート アラーム メッセージの例を示します。

```
LC/0/0/CPU0:Jan 15 21:39:11.325 2016:ifmgr[163]: %PKT_INFRA-LINEPRO
TO-5-UPDOWN : Line protocol on Interface HundredGigE 0/0/1/0, changed state to Down
```

メッセージのテキストには、アラームをロギングしたプロセスのロケーションが含まれます。この例では、アラームは HundredGigE インターフェイス 0/0/1/0 のライン プロトコルによってロギングされました。オプションで、出力に実際のアラーム ソースが含まれるように設定でき

ます。このアラーム ソースは、アラームをロギングしたプロセスとは異なる場合があります。これは、メッセージ テキストの前の追加表示フィールドに表示されます。

アラーム ソースのロケーションを表示した場合、一般形式は次のようになります。

node-id:*timestamp* : *process-name* [*pid*] : %*category-group-severity-code* : *source-location message-text*

次に、アラーム ソースのロケーションが表示されている場合の例を示します。

```
LC/0/0/CPU0:Jan 15 21:39:11.325 2016:ifmgr[163]: %PKT_INFRA-LINEPRO
TO-5-UPDOWN : interface HundredGigE 0/0/1/0: Line protocol on Interface HundredGigE
0/0/1/0, changed state to Down
```

階層的な相関

階層的な相関は、次の条件が満たされた場合に有効になります。

- 1 つのアラームが、あるルールの根本原因であり、かつ別のルールの非根本原因でもある場合。
- アラームが生成され、結果として両方のルールに関連する正常な相関となった場合。

次に、2 つの階層的な相関ルールの例を示します。

ルール 1	カテゴリ	グループ	コード
Root Cause 1	Cat 1	Group 1	Code 1
Non-root Cause 2	Cat 2	Group 2	Code 2
ルール 2			
Root Cause 2	Cat 2	Group 2	Code 2
Non-root Cause 3	Cat 3	Group 3	Code 3

Cause 1、2、3 に対して 3 つのアラームが生成され、すべてのアラームがそれぞれの相関タイムアウト期間内に着信した場合、階層的な相関は次のように出現します。

Cause 1 -> Cause 2 -> Cause 3

相関バッファには、2 つ（Cause 1 と Cause 2 に対して 1 個、Cause 2 と Cause 3 に対して 1 個）の異なる相関が示されます。ただし、階層的な関係は暗黙的に定義されます。

（注） アラームの再配置および再発行などのステートフル動作は、ステートフルとして定義されているルールの場合（つまり、相関が変化する可能性がある場合）にサポートされます。

コンテキスト相関フラグ

コンテキスト相関フラグを使用すると、「コンテキストごと」に相関が行われるかどうかを設定できます。

このフラグを使用すると、ルールが1つ以上のコンテキストに適用される場合のみ、動作が変化するようになります。このフラグは、ルータ全体またはロケーション ノード全体に対して適用されている場合は、有効になりません。

次に、コンテキスト相関動作のシナリオを示します。

- Rule 1 には、根本原因 A が含まれ、非根本原因が関連付けられている。
- Rule 1 にはコンテキスト相関フラグは設定されていない。
- Rule 1 はコンテキスト 1 および 2 に適用されている。

Rule 1 にコンテキスト相関フラグが設定されていない場合、コンテキスト 1 からアラーム A が生成され、コンテキスト 2 からアラーム B が生成されるシナリオでは、コンテキストのタイプにかかわらず、ルールが両方のコンテキストに適用されます。

Rule 1 にコンテキスト相関フラグが設定され、同じアラームが生成された場合、これらのアラームは、異なるコンテキストからのものであるとして、相関されません。

フラグが設定されていると、アラームが同じコンテキストから送信された場合に限り、コリレータはアラームをルールに照らして分析します。つまり、アラーム A がコンテキスト 1 から生成され、アラーム B がコンテキスト 2 から生成された場合、相関は行われません。

時間タイムアウト フラグ

根本原因タイムアウト（指定されている場合）は、特定のルールの根本原因アラームが着信する前に、非根本原因アラームが着信した状況の場合に使用する代替ルール タイムアウトです。通常、このタイムアウトは、根本原因アラームが着信する可能性が低く、そのため非根本原因アラームの保持がすぐに解除されることを想定して、より短いタイムアウトを設定する状況で使用されます。

再配置フラグ

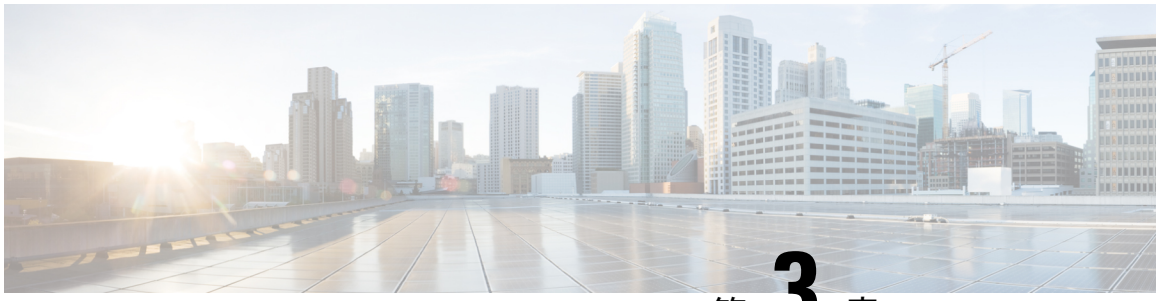
再配置フラグは、非根本原因アラームの直接の根本原因がクリアされた場合に、階層的な相関においてその非根本原因アラームがどのように処理されるかを指定します。

次に、コンテキスト相関動作の例を示します。

- Rule 1 には、根本原因 A が含まれ、非根本原因が関連付けられている。
- Rule 1 にはコンテキスト相関フラグは設定されていない。
- Rule 1 はコンテキスト 1 および 2 に適用されている。

このシナリオでは、コンテキスト 1 から生成されたアラーム A とコンテキスト 2 から生成されたアラーム B が送信された場合、コンテキストにかかわらず相関が行われます。

Rule 1 にコンテキスト相関フラグが設定され、同じアラームが生成された場合、これらのアラームは、異なるコンテキストからのものであるため、相関されません。



第 3 章

パフォーマンス管理の実装

Cisco IOS XR ソフトウェアのパフォーマンス管理（PM）では、次のタスクを実行するためのフレームワークが提供されます。

- データを保管および取得するために PM 統計情報を収集して TFTP サーバにエクスポートする
- 拡張マークアップ言語（XML）のクエリを使用してシステムをモニタする
- しきい値条件が一致するときにシステム ロギング メッセージを生成するしきい値条件を設定する

PM システムでは、システム リソースの使用率をグラフ化して、容量を計画したり、トラフィック エンジニアリングに使用したり、傾向を分析したりするために役立つデータを収集します。

パフォーマンス管理を実装する前提条件

ネットワーク オペレーション センター（NOC）にパフォーマンス管理を導入する前に、次の前提条件を満たしていることを確認します。

- 管理ソフトウェアのパッケージインストールエンベロープ（PIE）インストールしてアクティブにする必要があります。
- 適切なタスク ID を含むタスク グループに関連付けられているユーザ グループに属している必要があります。このコマンド リファレンスには、各コマンドに必要なタスク ID が含まれます。ユーザ グループの割り当てが原因でコマンドを使用できないと考えられる場合、AAA 管理者に連絡してください。
- TFTP サーバへの接続が必要です。

パフォーマンス管理の実装に関する情報

PM 機能の概要

パフォーマンス管理（PM）フレームワークは次の 2 つの主なコンポーネントで構成されています。

- PM 統計情報サーバ
- PM 統計情報収集機能

PM 統計情報サーバ

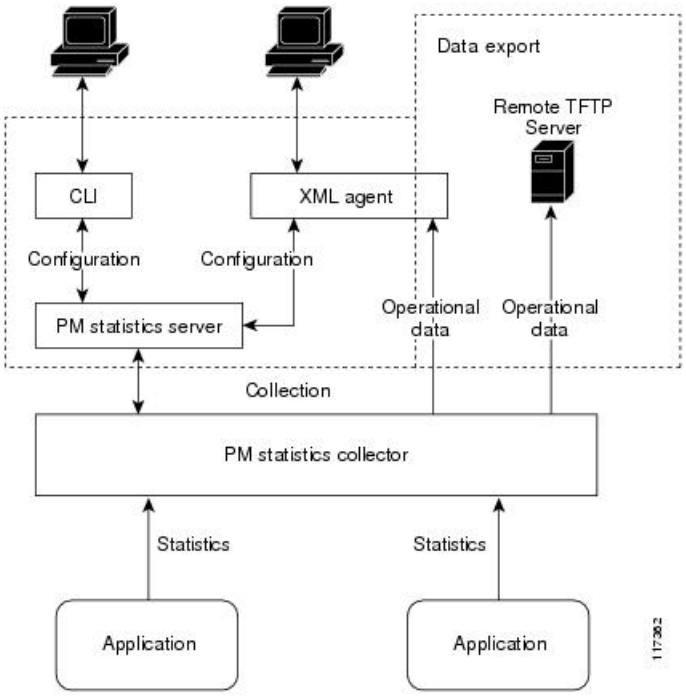
PM 統計情報サーバは統計情報収集、エンティティインスタンスモニタリング収集、しきい値モニタリングのフロントエンドです。コマンドライン インターフェイス（CLI）または XML スキームから設定されたすべての PM 統計情報収集およびしきい値条件は、PM 統計情報サーバによって処理され、PM 統計情報機能に分散されます。

PM 統計情報収集機能

PM 統計情報収集機能ではエンティティインスタンスから統計情報を収集し、そのデータをメモリに格納します。プロセスの再起動時に情報を利用できるように、メモリの内容のチェックポイントが行われます。さらに、PM 統計情報の収集機能は、XML エージェントおよび TFTP サーバへの動作データのエクスポートを担当します。

図 2 : PM コンポーネントの通信 （29 ページ） PM システムを構成するコンポーネントの関係を表しています。

図 *2 : PM* コンポーネントの通信



PM の利点

PM システムには次の利点があります。

- データ収集ポリシーを設定可能
- TFTP を経由してバイナリ形式で統計データを効率的に転送
- エンティティ インスタンス モニタリングをサポート
- しきい値モニタリングをサポート
- プロセスの再起動時およびプロセッサのフェールオーバー時にデータの一貫性を確保

PM 統計情報収集の概要

PM 統計情報収集では、はじめに PM システム内にあるエンティティのすべてのインスタンスに関連付けられているすべての属性から統計情報を収集します。次に、統計データをバイナリファイル形式で TFTP サーバにエクスポートします。たとえば、マルチプロトコル ラベル スイッチング（MPLS）ラベル配布プロトコル（LDP）統計情報収集では、ルータ上のすべての MPLS LDP セッションに関連付けられているすべての属性から統計データを収集します。

この表では、PM システムのエンティティおよび関連インスタンスを一覧で示します。

表 *6:* エンティティ クラスおよび関連付けられているインスタンス

エンティティ クラス	インスタンス
BGP	ネイバーまたはピア
インターフェイス基本カウンタ	インターフェイス
インターフェイスデータ レート	インターフェイス
インターフェイス汎用カウンタ	インターフェイス
MPLS LDP	LDP セッション
ノード CPU	ノード
ノード メモリ	ノード
ノード プロセス	プロセス
OSPFv2	プロセス
OSPFv3	プロセス

（注） PM システムを構成するエンティティに関連付けられているすべての属性のリストについては、表 9 : 属性と値 （38 ページ）を参照してください。

（注） インターフェイス タイプに応じて、インターフェイスはインターフェイスの汎用カウンタまたは基本カウンタのいずれかをサポートします。インターフェイスの基本カウンタをサポートするインターフェイスは、インターフェイスのデータ レートをサポートしません。

PM 統計情報をエクスポートするためのバイナリ ファイル形式

次のサンプルでは、バイナリ ファイル形式を説明します。

```
Version : 4 Bytes
NoOf Entities : 1 Byte  (e.g. . 4 )
Entity Identifier         : 1 Byte  (e.g NODE=1,Interface=2,BGP=3)
Options                    :2 Bytes
NoOf SubEntities             :1 Byte  (2)
SubEntity Identifier     :1 Byte  (e.g BGP-PEERS )
Time Stamp 4 Bytes (Reference Time : Start Ref Time)
No Of Instances    :2 Byte  (e.g 100)
                  Key Instance   :Variable
```

```
                        NoOfSamples: 1 Byte (e.g 10 Samples)
                           SampleNo : 1 Byte (e.g Sample No 1)
Time Stamp 4 Bytes (Sample Time)
                            StatCounterName :1 Byte (PeerSessionsEst=1)
                            StatCounterValue :8 Bytes ( for all counters)
                            Repeat for Each StatCounterName
                         Repeat for Each Sample No(Time Interval)
                 Repeat for All Instances
         Repeat for All SubTypes
Repeat for All Entities
```

エンティティのバイナリ ファイル ID 割り当て、サブエンティティ、統計情報カウンタ名

この表では、バイナリ ファイルでのさまざまな値の割り当ておよびキーを説明します。

表 7: バイナリ形式の値とキー

エンティティ	サブエンティティ	キー	統計情報カウンタ
ノード（1）	CPU（1）	CPU キー <Node ID>	表 8 : エンティティとサブエンティティでサポートされる統計情報カウンタ （32 ページ）を参照してください
	メモリ（2）	メモリ キー <Node ID>	
	プロセス（3）	ノード プロセス キー <NodeProcessID>	
インターフェイス（2）	汎用カウンタ（1）	汎用カウンタ キー <ifName>	
	データ レート カウンタ（2）	データ レート カウンタ キー <ifName>	
	基本カウンタ（3）	基本カウンタ キー <ifName>	
BGP（3）	ピア（1）	ピア キー <IpAddress>	
MPLS（4）	予約済み（1）	—	
	予約済み（2）	—	
	LDP（4）	LDP セッション キー <IpAddress>	

エンティティ	サブエンティティ	キー	統計情報カウンタ
OSPF（5）	v2protocol（1）	インスタンス <process_instance>	
	v3protocol（2）	インスタンス <process_instance>	

（注） <ifName>：長さの値は変数です。最初の2バイトにはインスタンスIDのサイズが含まれます。その次にインスタンス ID 文字列（インターフェイス名）が続きます。

<IpAddress>：IP アドレスが含まれる 4 バイトです。

<NodeProcessID>：64 ビットのインスタンス ID です。最初の 32 ビットにはノード ID が含まれ、次の 32 ビットにはプロセス ID が含まれます。

<NodeID>：ノード ID が含まれる 32 ビット インスタンスです。

<process_instance>：長さの値は変数です。最初の 2 バイトにはインスタンス ID のサイズが含まれ、その次にインスタンス ID 文字列（プロセス名）が続きます。

（注） 括弧の中の数字（表 7 : バイナリ形式の値とキー （31 ページ）の各エンティティとサブエンティティに関連付けられている数字）は、TFTP ファイルに表示されるエンティティ ID とサブエンティティ ID を表します。

この表では、エンティティとサブエンティティのバイナリ ファイルに収集される、サポート対象の統計情報カウンタを説明します。

表 *8*: エンティティとサブエンティティでサポートされる統計情報カウンタ

エンティティ	サブエンティティ	統計情報カウンタ
ノード（1）	CPU（1）	AverageCPUUsed、NoProcesses
	メモリ （2）	CurrMemory、PeakMemory
	プロセス（3）	PeakMemory、AverageCPUUsed、NoThreads
インターフェイス（2）	汎用カウンタ（1）	InPackets、InOctets、OutPackets、OutOctets、InUcastPkts、InMulticastPkts、InBroadcastPkts、OutUcastPkts、OutMulticastPkts、OutBroadcastPkts、OutputTotalDrops、InputTotalDrops、InputQueueDrops、InputUnknownProto、OutputTotalErrors、OutputUnderrun、InputTotalErrors、InputCRC、InputOverrun、InputFrame

エンティティ	サブエンティティ	統計情報カウンタ
	データ レート カウンタ（2）	InputDataRate、InputPacketRate、OutputDataRate、OutputPacketRate、InputPeakRate、InputPeakPkts、OutputPeakRate、OutputPeakPkts、Bandwidth
	基本カウンタ	InPackets、InOctets、OutPackets、OutOctets、InputTotalDrops、InputQueueDrops、InputTotalErrors、OutputTotalErrors、OutputQueueDrops、OutputTotalErrors
BGP（3）	ピア（1）	InputMessages、OutputMessages、InputUpdateMessages、OutputUpdateMessages、ConnEstablished、ConnDropped、ErrorsReceived、ErrorsSent
MPLS（4）	LDP（4）	TotalMsgsSent、TotalMsgsRcvd、InitMsgsSent、InitMsgsRcvd、AddressMsgsSent、AddressMsgsRcvd、AddressWithdrawMsgsSent、AddressWithdrawMsgsRcvd、LabelMappingMsgsSent、LabelMappingMsgsRcvd、LabelWithdrawMsgsSent、LabelWithdrawMsgsRcvd、LabelReleaseMsgsSent、LabelReleaseMsgsRcvd、NotificationMsgsSent、NotificationMsgsRcvd KeepAliveMsgsSent、KeepAliveMsgsRcvd
OSPF（5）	v2protocol（1）	InputPackets、OutputPackets、InputHelloPackets、OutputHelloPackets、InputDBDs、InputDBDsLSA、OutputDBDs、OutputDBDsLSA、InputLSRequests、InputLSRequestsLSA、OutputLSRequests、OutputLSRequestsLSA、InputLSAUpdates、InputLSAUpdatesLSA、OutputLSAUpdates、OutputLSAUpdatesLSA、InputLSAAcks、InputLSAAcksLSA、OutputLSAAcks、OutputLSAAcksLSA、ChecksumErrors
	v3protocol（2）	InputPackets、OutputPackets、InputHelloPackets、OutputHelloPackets、InputDBDs、InputDBDsLSA、OutputDBDs、OutputDBDsLSA、InputLSRequests、InputLSRequestsLSA、OutputLSRequests、OutputLSRequestsLSA、InputLSAUpdates、InputLSAUpdatesLSA、OutputLSAUpdates、OutputLSAUpdatesLSA、InputLSAAcks、InputLSAAcksLSA、OutputLSAAcks、OutputLSAAcksLSA

バイナリ ファイルに適用されるファイルの命名規則

次のファイルの命名規則は、TFTP サーバに設定されているディレクトリの場所に送信される PM 統計情報収集に適用されます。

<LR_NAME>_<EntityName>_<SubentityName>_<TimeStamp>

パフォーマンス管理の実装方法

PM 統計情報収集用の外部 TFTP サーバまたはローカル ディスクの設定

PM 統計データを外部の TFTP サーバにエクスポートしたり、ローカル ファイル システムにダンプしたりすることができます。ローカル宛先と TFTP 宛先はともに相互に排他的で、一度に設定できるのはどちらか一方です。

設定例

次の例では、PM 統計情報収集用に外部の TFTP サーバを設定します。

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# performance-mgmt resources tftp-server 10.3.40.161 directory
 mypmdata/datafiles
RP/0/RP0/CPU0:Router(config)# commit
```

次の例では、PM 統計情報収集用にローカル ディスクを設定します。

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# performance-mgmt resources dump local
RP/0/RP0/CPU0:Router(config)# commit
```

PM 統計情報収集テンプレートの設定

PM 統計情報収集は、PM 統計情報収集テンプレートから設定されます。PM 統計情報収集テンプレートには、エンティティ、サンプル間隔、TFTP サーバにデータをエクスポートするまでに実行されるサンプリング動作の回数が含まれます。PM 統計情報収集テンプレートがイネーブルの場合、PM 統計情報収集は、テンプレートに設定されているエンティティに関連付けられているすべてのインスタンスからの属性の統計情報を収集します。特定のエンティティに複数のテンプレートを定義できます。ただし、特定のエンティティに一度にイネーブルにできる PM 統計情報テンプレートは 1 つだけです。

PM 統計情報収集テンプレートを設定するガイドライン

PM 統計情報テンプレートの作成時は、次のガイドラインに従ってください。

- リモート TFTP サーバやローカル ディスクに統計情報データをエクスポートする場合、TFTP サーバ リソースやローカル ダンプ リソースを設定する必要があります。
- 特定のエンティティに対して複数のテンプレートを定義できますが、特定のエンティティに対して同時に有効にできる PM 統計情報収集テンプレートは 1 つだけです。
- テンプレートを設定するときは、デフォルトのキーワードを使用してエンティティのテンプレートをデフォルトのテンプレートとして指定するか、またはテンプレートに名前を付けることができます。デフォルト テンプレートには、次のデフォルト値が含まれています。

- 10 分のサンプル間隔。
- 5 つのサンプリング動作のサンプル サイズ。

- サンプル間隔は、サンプリング サイクル中に実行されるサンプリング動作の頻度を設定します。sample-interval コマンドを使用して、サンプルの間隔を設定できます。範囲は 1 ～ 60 分です。
- sample size では、データを TFTP サーバにエクスポートする前に実行されるサンプリング動作の数を設定します。 **sample-size** コマンドを使用して、サンプルのサイズを設定できます。範囲は 1 ～ 60 サンプルです。

（注） 小さいサンプル間隔を指定するとCPU使用率が増加し、大きいサンプル サイズを指定するとメモリ使用率が増加します。そのため、システムのオーバーロードを防ぐために、サンプル サイズとサンプル間隔の調整が必要になる場合があります。

- エクスポート サイクルでは、PM 統計情報収集データが TFTP サーバにエクスポートされる頻度を決定します。エクスポート サイクルは、サンプル間隔にサンプル サイズを掛け合わせて計算します（サンプル間隔 x サンプル サイズ = エクスポート サイクル）。
- テンプレートをイネーブルにすると、performance-mgmt apply statistics コマンドの no 形式でテンプレートをディセーブルにするまで、サンプリングとエクスポートのサイクルは継続されます。

- 次のエンティティの PM 統計情報収集を有効または無効にするときは、**location** コマンドを使用してノードを指定するか、または **location all** コマンドを使用してすべてのノードの PM 統計情報収集を有効にする必要があります。
  - ノード CPU
  - ノード メモリ
  - ノード プロセス

設定例

次の例では、PM 統計情報収集テンプレートの作成方法およびイネーブル方法を示します。

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# performance-mgmt statistics interface generic-counters
template template 1
RP/0/RP0/CPU0:Router(config)# performance-mgmt statistics interface generic-counters
template 1 sample-size 10
RP/0/RP0/CPU0:Router(config)# performance-mgmt statistics interface generic-counters
template 1 sample-interval 5
RP/0/RP0/CPU0:Router(config)# performance-mgmt apply statistics interface generic-counters
 1
RP/0/RP0/CPU0:Router# commit
```

PM エンティティ インスタンス モニタリングのイネーブル化

エンティティ インスタンス モニタリングでは、特定のエンティティ インスタンスに関連付けられている属性から統計情報を収集します。エンティティ インスタンスのモニタリングがイネーブルな場合、PM システムは指定したエンティティ インスタンスに関連する属性の統計情報だけを収集します。PM システムでは、モニタリング対象のエンティティの PM 統計情報収集テンプレートで設定されているサンプリング サイクルを使用します。ただし、エンティティ インスタンス モニタリングは、PM 統計情報収集のプロセスとは別のプロセスです。そのため、PM統計情報収集とは連携しません。さらに、エンティティインスタンスモニタリング収集からのデータは PM 統計情報収集から独立しています。PM 統計情報収集とは異なり、エンティティインスタンスモニタリングからのデータはTFTPサーバにエクスポートされません。各エンティティ インスタンスに関連付けられているすべての属性とコマンドの詳細については、パフォーマンス管理：詳細 （37 ページ） を参照してください。

設定例

次の例に、ノードCPUエンティティ インスタンスのエンティティ インスタンス モニタリングを有効にする方法を示します。

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor node cpu location 0/RP0/CPU0
 default
RP/0/RP0/CPU0:Router(config)# commit
```

PM しきい値モニタリング テンプレートの設定

PM システムでは、しきい値条件を設定して、しきい値違反の属性をモニタできます。しきい値条件は PM しきい値モニタリング テンプレートから設定されます。PM しきい値テンプレートがイネーブルの場合、PM システムはテンプレートに設定されているしきい値条件の属性のインスタンスをすべてモニタします。サンプル間隔の最後にしきい値条件が一致すると、PM システムではしきい値条件に一致したそれぞれのインスタンスにシステム ロギング メッセージを生成します。すべてのエンティティの属性および各属性に関連付けられている値の範囲のリストについては、を参照してください。 パフォーマンス管理：詳細 （37 ページ）

PM しきい値モニタリング テンプレートを設定するガイドライン

PM しきい値モニタリングテンプレートを設定するときには、次のガイドラインに従います。

- テンプレートが有効になると、**performance-mgmt apply thresholds** コマンドの **no** 形式でテンプレートが無効になるまで、しきい値モニタリングが継続されます。
- 1 つのエンティティで一度にイネーブルにできる PM しきい値テンプレートは 1 つだけです。
- 次のエンティティに対しPMしきい値モニタリングテンプレートを有効または無効にするときは、**location** コマンドを使用してノードを指定するか、または **location all** コマンドを使用してすべてのノードの PM 統計情報収集を有効にする必要があります。
  - ノード CPU

- ノード メモリ
- ノード プロセス

設定例

この例では、PM しきい値モニタリング テンプレートの作成方法およびイネーブル方法を示します。この例では、ノード CPU エンティティの AverageCpuUsed 属性の PM しきい値テンプレートが作成されます。この PM しきい値条件のしきい値条件では、AverageCpuUsed 属性をモニタして、CPU 平均使用率が 25 % より大きいかどうかを決定します。

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# performance-mgmt thresholds node cpu template template20
RP/0/RP0/CPU0:Router(config-threshold-cpu)# AverageCpuUsed gt 25 percent
RP/0/RP0/CPU0:Router(config-threshold-cpu)# exit
RP/0/RP0/CPU0:Router(config)# performance-mgmt apply thresholds node cpu location all
template20
RP/0/RP0/CPU0:Router# commit
```

正規表現によるインスタンス フィルタリングの設定

このタスクでは、1 つ以上の統計テンプレートまたはしきい値テンプレートに適用できる正規表現グループの定義について説明します。複数の正規表現インデックスを含めることもできます。正規表現グループを使用したインスタンス フィルタリングの利点は次のとおりです。

- 複数のテンプレートに適用できる同じ正規表現グループを使用できる。
- 同じ索引値を割り当てると、柔軟性を向上できる。
- 正規表現の OR 条件を適用すると、パフォーマンスを向上できる。

（注） 正規表現によるインスタンスのフィルタリングは、現在インターフェイス エンティティでのみサポートされています（Interface basic-counters、generic-counters、data-rates）。

設定例

次に、正規表現グループを定義する例を示します。

```
RP/0/RP0/CPU0:Router# configure
RP/0/RP0/CPU0:Router(config)# performance-mgmt regular-expression regexp
RP/0/RP0/CPU0:Router(config-perfmgmt-regex)# index 10 match
RP/0/RP0/CPU0:Router(config)# commit
```

パフォーマンス管理：詳細

この項には、パフォーマンス管理を構成する際に役立つ追加情報が含まれています。

この表では、PM システムを構成するすべてのエンティティの各属性に関連付けられている属性と値の範囲を説明します。

表 *9:* 属性と値

エンティティ	属性	説明	値
bgp	ConnDropped	接続がドロップされた回数。	範囲は 0 ～ 4294967295 です。
	ConnEstablished	接続が確立された回数。	範囲は 0 ～ 4294967295 です。
	ErrorsReceived	接続で受信されたエラー通知の数。	範囲は 0 ～ 4294967295 です。
	ErrorsSent	接続で送信されたエラー通知の数。	範囲は 0 ～ 4294967295 です。
	InputMessages	受信されたメッセージの数。	範囲は 0 ～ 4294967295 です。
	InputUpdateMessages	受信されたアップデートメッセージの数。	範囲は 0 ～ 4294967295 です。
	OutputMessages	送信されたメッセージの数。	範囲は 0 ～ 4294967295 です。
	OutputUpdateMessages	送信されたアップデートメッセージの数。	範囲は 0 ～ 4294967295 です。

エンティティ	属性	説明	値
interface data-rates	Bandwidth	帯域幅（kbps 単位）。	範囲は 0 ～ 4294967295 です。
	InputDataRate	入力データ レート（kbps 単位）。	範囲は 0 ～ 4294967295 です。
	InputPacketRate	入力パケット/秒。	範囲は 0 ～ 4294967295 です。
	InputPeakRate	ピーク入力データ レート。	範囲は 0 ～ 4294967295 です。
	InputPeakPkts	ピーク入力パケット レート。	範囲は 0 ～ 4294967295 です。
	OutputDataRate	出力データ レート（kbps 単位）。	範囲は 0 ～ 4294967295 です。
	OutputPacketRate	出力パケット/秒。	範囲は 0 ～ 4294967295 です。
	OutputPeakPkts	ピーク出力パケット レート。	範囲は 0 ～ 4294967295 です。
	OutputPeakRate	ピーク出力データ レート。	範囲は 0 ～ 4294967295 です。

エンティティ	属性	説明	値
interface basic-counters	InPackets	受信されたパケット数。	範囲は 0 ～ 4294967295 です。
	InOctets	受信されたバイト数。	範囲は 0 ～ 4294967295 です。
	OutPackets	送信されたパケット数。	範囲は 0 ～ 4294967295 です。
	OutOctets	送信されたバイト数。	範囲は 0 ～ 4294967295 です。
	InputTotalDrops	インバウンドの廃棄された適正なパケット。	範囲は 0 ～ 4294967295 です。
	InputQueueDrops	入力キューのドロップ。	範囲は 0 ～ 4294967295 です。
	InputTotalErrors	インバウンドの廃棄された不正なパケット。	範囲は 0 ～ 4294967295 です。
	OutputTotalDrops	アウトバウンドの廃棄された適正なパケット。	範囲は 0 ～ 4294967295 です。
	OutputQueueDrops	出力キューのドロップ。	範囲は 0 ～ 4294967295 です。
	OutputTotalErrors	アウトバウンドの廃棄された不正なパケット。	範囲は 0 ～ 4294967295 です。

エンティティ	属性	説明	値
interface generic-counters	InBroadcastPkts	受信されたブロードキャストパケット。	範囲は 0 ～ 4294967295 です。
	InMulticastPkts	受信されたマルチキャストパケット。	範囲は 0 ～ 4294967295 です。
	InOctets	受信されたバイト数。	範囲は 0 ～ 4294967295 です。
	InPackets	受信されたパケット数。	範囲は 0 ～ 4294967295 です。
	InputCRC	不正な CRC で廃棄されたインバウンドパケット。	範囲は 0 ～ 4294967295 です。
	InputFrame	インバウンドフレーム エラー。	範囲は 0 ～ 4294967295 です。
	InputOverrun	入力オーバーラン。	範囲は 0 ～ 4294967295 です。
	InputQueueDrops	入力キューのドロップ。	範囲は 0 ～ 4294967295 です。
	InputTotalDrops	インバウンドの廃棄された適正なパケット。	範囲は 0 ～ 4294967295 です。
	InputTotalErrors	インバウンドの廃棄された不正なパケット。	範囲は 0 ～ 4294967295 です。
	InUcastPkts	受信されたユニキャスト パケット。	範囲は 0 ～ 4294967295 です。
	InputUnknownProto	不明なプロトコルで廃棄されたインバウンド パケット。	範囲は 0 ～ 4294967295 です。
	OutBroadcastPkts	送信されたブロードキャストパケット。	範囲は 0 ～ 4294967295 です。
	OutMulticastPkts	送信されたマルチキャストパケット。	範囲は 0 ～ 4294967295 です。
	OutOctets		

エンティティ	属性	説明	値
		送信されたバイト数。	範囲は 0 ～ 4294967295 です。
	OutPackets	送信されたパケット数。	範囲は 0 ～ 4294967295 です。
	OutputTotalDrops	アウトバウンドの廃棄された適正なパケット。	範囲は 0 ～ 4294967295 です。
	OutputTotalErrors	アウトバウンドの廃棄された不正なパケット。	範囲は 0 ～ 4294967295 です。
	OutUcastPkts	送信されたユニキャスト パケット。	範囲は 0 ～ 4294967295 です。
	OutputUnderrun	出力アンダーラン。	範囲は 0 ～ 4294967295 です。

エンティティ	属性	説明	値
mpls ldp	AddressMsgsRcvd	受信されたアドレスメッセージ。	範囲は 0 ～ 4294967295 です。
	AddressMsgsSent	送信されたアドレスメッセージ。	範囲は 0 ～ 4294967295 です。
	AddressWithdrawMsgsRcvd	受信されたアドレスウィズドロー メッセージ。	範囲は 0 ～ 4294967295 です。
	AddressWithdrawMsgsSent	送信されたアドレスウィズドロー メッセージ。	範囲は 0 ～ 4294967295 です。
	InitMsgsSent	送信された初期メッセージ。	範囲は 0 ～ 4294967295 です。
	InitMsgsRcvd	受信された初期メッセージ。	範囲は 0 ～ 4294967295 です。
	KeepaliveMsgsRcvd	受信されたキープアライブ メッセージ。	範囲は 0 ～ 4294967295 です。
	KeepaliveMsgsSent	送信されたキープアライブ メッセージ。	範囲は 0 ～ 4294967295 です。
	LabelMappingMsgsRcvd	受信されたラベルマッピング メッセージ。	範囲は 0 ～ 4294967295 です。
	LabelMappingMsgsSent	送信されたラベルマッピング メッセージ。	範囲は 0 ～ 4294967295 です。
	LabelReleaseMsgsRcvd	受信されたラベル リリース メッセージ。	範囲は 0 ～ 4294967295 です。
	LabelReleaseMsgsSent	送信されたラベル リリース メッセージ。	範囲は 0 ～ 4294967295 です。
	LabelWithdrawMsgsRcvd	受信されたラベルウィズドロー メッセージ。	範囲は 0 ～ 4294967295 です。
	LabelWithdrawMsgsSent	送信されたラベルウィズドロー メッセージ。	範囲は 0 ～ 4294967295 です。

エンティティ	属性	説明	値
	NotificationMsgsRcvd	受信された通知メッセージ。	範囲は 0 ～ 4294967295 です。
	NotificationMsgsSent	送信された通知メッセージ。	範囲は 0 ～ 4294967295 です。
	TotalMsgsRcvd	受信されたメッセージの合計数。	範囲は 0 ～ 4294967295 です。
	TotalMsgsSent	送信されたメッセージの合計数。	範囲は 0 ～ 4294967295 です。
node cpu	AverageCPUUsed	平均 CPU 利用率。	範囲は 0 ～ 100 のパーセンテージです。
	NoProcesses	プロセス数。	範囲は 0 ～ 4294967295 です。
node memory	CurrMemory	現在使用中のアプリケーション メモリ（バイト単位）。	範囲は 0 ～ 4294967295 です。
	PeakMemory	ブートアップ後に使用された最大システム メモリ（MB 単位）。	範囲は 0 ～ 4194304 です。
node process	AverageCPUUsed	平均 CPU 利用率。	範囲は 0 ～ 100 のパーセンテージです。
	NoThreads	スレッド数。	範囲は 0 ～ 4294967295 です。
	PeakMemory	起動時以降に使用された最大ダイナミック メモリ（KB 単位）。	範囲は 0 ～ 4194304 です。

エンティティ	属性	説明	値
ospf v2protocol	InputPackets	受信されたパケットの合計数。	範囲は 0 ～ 4294967295 です。
	OutputPackets	送信されたパケットの合計数。	範囲は 0 ～ 4294967295 です。
	InputHelloPackets	受信された hello パケットの合計数。	範囲は 0 ～ 4294967295 です。
	OutputHelloPackets	送信された hello パケットの合計数。	範囲は 0 ～ 4294967295 です。
	InputDBDs	受信された DBD パケットの合計数。	範囲は 0 ～ 4294967295 です。
	InputDBDsLSA	DBDパケットで受信された LSA の数。	範囲は 0 ～ 4294967295 です。
	OutputDBDs	送信された DBD パケットの合計数。	範囲は 0 ～ 4294967295 です。
	OutputDBDsLSA	DBDパケットで送信された LSA の数。	範囲は 0 ～ 4294967295 です。
	InputLSRequests	受信された LS 要求の数。	範囲は 0 ～ 4294967295 です。
	InputLSRequestsLSA	LS要求で受信された LSA の数。	範囲は 0 ～ 4294967295 です。
	OutputLSRequests	送信された LS 要求の数。	範囲は 0 ～ 4294967295 です。
	OutputLSRequestsLSA	LS要求で送信された LSA の数。	範囲は 0 ～ 4294967295 です。
	InputLSAUpdates	受信されたLSAアップデートの数。	範囲は 0 ～ 4294967295 です。
	InputLSAUpdatesLSA	LSAアップデートで受信された LSA の数。	範囲は 0 ～ 4294967295 です。
	OutputLSAUpdates	送信されたLSAアップデートの数。	範囲は 0 ～ 4294967295 です。
	OutputLSAUpdatesLSA		範囲は 0 ～ 4294967295 です。

エンティティ	属性	説明	値
		LSAアップデートで送信されたLSAの数。	
	InputLSAAcks	受信されたLSAアクノレッジメントの数。	範囲は0～4294967295です。
	InputLSAAcksLSA	LSAアクノレッジメントで受信されたLSAの数。	範囲は0～4294967295です。
	OutputLSAAcks	送信されたLSAアクノレッジメントの数。	範囲は0～4294967295です。
	OutputLSAAcksLSA	LSAアクノレッジメントで送信されたLSAの数。	範囲は0～4294967295です。
	ChecksumErrors	チェックサムエラーで受信されたパケット数。	範囲は0～4294967295です。

エンティティ	属性	説明	値
ospf v3protocol	InputPackets	受信されたパケットの合計数。	範囲は 0 ～ 4294967295 です。
	OutputPackets	送信されたパケットの合計数。	範囲は 0 ～ 4294967295 です。
	InputHelloPackets	受信された hello パケットの合計数。	範囲は 0 ～ 4294967295 です。
	OutputHelloPackets	送信された hello パケットの合計数。	範囲は 0 ～ 4294967295 です。
	InputDBDs	受信された DBD パケットの合計数。	範囲は 0 ～ 4294967295 です。
	InputDBDsLSA	DBDパケットで受信された LSA の数。	範囲は 0 ～ 4294967295 です。
	OutputDBDs	送信された DBD パケットの合計数。	範囲は 0 ～ 4294967295 です。
	OutputDBDsLSA	DBDパケットで送信された LSA の数。	範囲は 0 ～ 4294967295 です。
	InputLSRequests	受信された LS 要求の数。	範囲は 0 ～ 4294967295 です。
	InputLSRequestsLSA	LS要求で受信された LSA の数。	範囲は 0 ～ 4294967295 です。
	OutputLSRequests	送信された LS 要求の数。	範囲は 0 ～ 4294967295 です。
	OutputLSRequestsLSA	LS要求で送信された LSA の数。	範囲は 0 ～ 4294967295 です。
	InputLSAUpdates	受信されたLSAアップデートの数。	範囲は 0 ～ 4294967295 です。
	InputLSRequestsLSA	LS要求で受信された LSA の数。	範囲は 0 ～ 4294967295 です。
	OutputLSAUpdates	送信されたLSAアップデートの数。	範囲は 0 ～ 4294967295 です。
	OutputLSAUpdatesLSA	LSAアップデートで送信された LSA の数。	範囲は 0 ～ 4294967295 です。

エンティティ	属性	説明	値
	InputLSAAcks	受信されたLSAアクノレッジメントの数。	範囲は 0 ～ 4294967295 です。
	InputLSAAcksLSA	LSAアクノレッジメントで受信されたLSA の数。	範囲は 0 ～ 4294967295 です。
	OutputLSAAcks	送信されたLSAアクノレッジメントの数。	範囲は 0 ～ 4294967295 です。
	OutputLSAAcksLSA	LSAアクノレッジメントで送信されたLSA の数。	範囲は 0 ～ 4294967295 です。

この表では、さまざまなエンティティ インスタンスに対しエンティティ インスタンス モニタリングをイネーブルにするために使用されるコマンドを説明します。

表 *10*：エンティティ インスタンスとモニタリング コマンド

エンティティ	コマンドの説明
BGP	BGP エンティティ インスタンスのエンティティ インスタンス モニタリングを有効にするには、 **performance-mgmt apply monitor bgp** コマンドを使用します。 **構文**： **performance-mgmt** **apply monitor** **bgp** *ip-address* *template-name* | **default**} `RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor bgp 10.12.0.4 default`

エンティティ	コマンドの説明

インターフェイス データ レート	インターフェイス データ レートのエンティティ インスタンスのエンティティ インスタンス モニタリングを有効にするには、 **performance-mgmt apply monitor data-rates** コマンドを使用します。
	構文：
	performance-mgmt
	apply
	monitor
	interface
	data-rates
	type
	interface-path-id {*template-name* |
	default}
	`RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor interface data-rates`
	` HundredGigE 0/0/1/0 default`

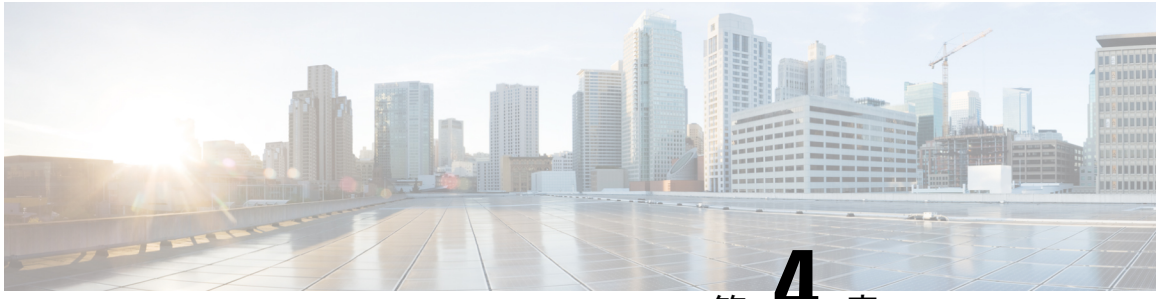
インターフェイス基本カウンタ	インターフェイス基本カウンタのエンティティ インスタンスのエンティティ インスタンス モニタリングを有効にするには、 **performance-mgmt apply monitor interface basic-counters** コマンドを使用します。
	構文：
	performance-mgmt
	apply
	monitor
	interface
	basic-counters
	type
	interface-path-id {*template-name* |
	default}
	`RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor interface basic-counters`
	`HundredGigE 0/0/1/0 default`

エンティティ	コマンドの説明

インターフェイス汎用カウンタ	インターフェイス汎用カウンタのエンティティ インスタンスのエンティティ インスタンス モニタリングを有効にするには、 **performance-mgmt apply monitor interface generic-counters** コマンドを使用します。
	構文：
	performance-mgmt
	apply
	monitor
	interface
	generic-counters
	type
	interface-path-id {*template-name* |
	default}
	`RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor interface generic-counters`
	`HundredGigE 0/0/1/0 default`

MPLS LDP	MPLS LDP エンティティ インスタンスのエンティティ インスタンス モニタリングを有効にするには、 **performance-mgmt apply monitor mpls ldp** コマンドを使用します。 **構文**： **performance-mgmt** **apply monitor** **mpls** **ldp** *ip-address* {*template-name* | **default**} `RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor mpls ldp 10.34.64.154 default`
ノード CPU	ノード CPU エンティティ インスタンスのエンティティ インスタンス モニタリングを有効にするには、 **performance-mgmt apply monitor node cpu** コマンドを使用します。 **構文**： **performance-mgmt** **apply** **monitor** **node** **cpu** **location** *node-id* {*template-name* | **default**} `RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor node cpu location 0/RP0/CPU0 default`

エンティティ	コマンドの説明
ノード メモリ	ノード メモリ エンティティ インスタンスのエンティティ インスタンス モニタリングを有効にするには、 **performance-mgmt apply monitor node memory** コマンドを使用します。 **構文**： **performance-mgmt** **apply** **monitor** **node** **memory** **location** *node-id* {*template-name* | **default**} `RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor node memory location 0/RP0/CPU0 default`
ノード プロセス	ノード プロセス エンティティ インスタンスのエンティティ インスタンス モニタリング コレクションを有効にするには、 **performance-mgmt apply monitor node process** コマンドを使用します。 **構文**： **performance-mgmt** **apply monitor node** **process** **location** *node-id* *pid* {*template-name* | **default**} `RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor node process location p 0/RP0/CPU0 275 default`



第 4 章

Embedded Event Manager ポリシーの設定および管理

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

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

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

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

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

- Embedded Event Manager ポリシーの設定および管理の前提条件 （54 ページ）
- Embedded Event Manager ポリシーの設定および管理について （54 ページ）
- Embedded Event Manager ポリシーの設定および管理方法 （65 ページ）

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

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

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

Event Management

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

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

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

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

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

システム イベント処理

イベント通知を受信すると、EEM は次のアクションを実行します。

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

Embedded Event Manager スクリプト

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

さらに、IOS XR オペレーティング システムに付属のオンボード Tcl スクリプトを使用して、ユーザは独自の TCL ベースのポリシーを記述できます。シスコでは、EEM ポリシーの記述に活用できる Tcl コマンド拡張の形式で、Tcl 言語の機能を拡張しています。EEM Tcl コマンド拡張の詳細については、を参照してください。 Embedded Event Manager ポリシー Tcl コマンド拡張カテゴリ （55 ページ）

EEM スクリプトの作成には、次の手順が含まれます。

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

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

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

表 *11 : Embedded Event Manager Tcl* コマンド拡張カテゴリ

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

Embedded Event Manager 用のシスコ ファイル命名規則

すべての EEM ポリシー名、ポリシー サポート ファイル（たとえば、電子メール テンプレート ファイル）、およびライブラリ ファイル名は、シスコのファイル命名規則に従う必要があります。これに関連し、EEM ポリシーのファイル名は次の仕様に従います。

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

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

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

表 *12 : 2* 文字の省略形の指定

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

Embedded Event Manager の組み込みアクション

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

次の表に、各 EEM ハンドラ要求またはアクションを示します。

表 *13 : Embedded Event Manager* の組み込みアクション

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

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

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

アプリケーション固有の組み込みイベント管理

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

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

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

イベント検出とリカバリ

EEM は、イベント ディテクタと呼ばれるソフトウェア エージェントを使用してシステム内の異なるコンポーネントのモニタリングをサポートする、柔軟でポリシードリブンのフレームワークです。イベント ディテクタは、他の Cisco IOS XR ソフトウェア コンポーネントと EEM の間のインターフェイスを提供する個別のプログラムです。イベント ディテクタ（イベント パブリッシャ）はイベントを選別し、イベント サブスクライバ（ポリシー）によって提供されたイベント仕様と一致するときに、それらをパブリッシュします。イベント ディテクタは、注目するイベントが発生したときに EEM サーバに通知します。

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

- システム EEM イベント
- アプリケーション定義イベント

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

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

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

次のイベント ディテクタがサポートされています。

System Manager イベント ディテクタ

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

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

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

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

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

表 *14 : System Manager* イベント ディテクタ イベント モニタリング要求

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

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

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

タイマー サービス イベント ディテクタ

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

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

syslog イベント ディテクタ

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

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

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

表 *15 : syslog* イベント ディテクタイベント モニタリング要求

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

None イベント ディテクタ

None イベント ディテクタは、Cisco IOS XR ソフトウェア **event manager run** CLI コマンドが EEM ポリシーを実行したときにイベントをパブリッシュします。EEM は、ポリシーそのものに含まれるイベント仕様に基づいてポリシーをスケジューリングし、実行します。EEM ポリシーは識別される必要があり、手動での実行が許可されるように、**event manager run** コマンドが実行される前に登録される必要があります。

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

Watchdog System Monitor イベント ディテクタ

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

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

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

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

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

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

表 *16 : Watchdog System Monitor* イベント ディテクタ要求

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

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

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

分散イベント ディテクタ

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

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

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

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

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

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

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

信頼性統計情報

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

ハードウェア カードの信頼性メトリック データ

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

ハードウェア カードによって保持されるデータは、次のとおりです。

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

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

プロセス信頼性メトリック データ

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

プロセスの終了には次の場合が含まれます。

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

プロセスが保持するデータは次のとおりです。

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

Embedded Event Manager ポリシーの設定および管理方法

環境変数の設定

EEM 環境変数は、ポリシーの実行前にポリシーに対して外部定義された Tcl グローバル変数です。EEM ポリシー エンジンは、障害およびその他のイベントが発生したときに通知を受け取ります。EEM ポリシーは、システムの現在の状態に基づいて回復を実行し、該当するイベントのポリシーに指定されたアクションを実行します。回復アクションはポリシーが実行されたときにトリガーされます。

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

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

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

設定例

次の例では、一連の EEM 環境変数を定義する方法を示します。

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

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

Embedded Event Manager ポリシーの登録

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

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

EEM ポリシーの登録時に以下を指定する必要があります。

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

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

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

設定例

次に、ユーザ定義の EEM ポリシーを登録する例を示します。

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

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

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

この項では、次の作業について説明します。

EEM Tcl スクリプトの登録と定義

環境変数を設定し、EEMポリシーを登録するには、この作業を実行します。EEMは、ポリシーそのものに含まれるイベント仕様に基づいてポリシーをスケジューリングし、実行します。EEM ポリシーが登録されると、ソフトウェアによって、ポリシーが調べられ、指定されたイベントの発生時に実行されるよう、登録されます。

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

設定例

次に、EEM ポリシーを登録して定義する例を示します。

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

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

EEM ポリシー実行の一時停止

一時的なパフォーマンスまたはセキュリティ面での理由から、ポリシーの登録解除ではなく一時停止が必要なことがあります。必要に応じて、**event manager scheduler suspend** コマンドを使用してすべての EEM ポリシーの実行をただちに一時停止することができます。

設定例

次に、すべての EEM ポリシーの実行を一時停止する例を示します。

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

EEM ポリシーを格納するディレクトリの指定

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

設定例

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

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

Tcl を使用した EEM ポリシーのプログラミング

Tcl コマンド拡張を使用してポリシーをプログラムするには、この作業を実行します。既存のポリシーをコピーし、変更することを推奨します。EEM Tcl ポリシーには、event_register Tcl コマンド拡張と本体の 2 つの必須部分が存在する必要があります。Tcl ポリシー構造と要件の詳細については、を参照してください。 TCL を使用した EEM ポリシー：詳細 （79 ページ）

手順

	コマンドまたはアクション	目的
ステップ 1	**show event manager policy available** [**system** | **user**] **例**： `RP/0/RP0/CPU0:Router# show event manager policy available`	登録可能な EEM ポリシーを表示します。
ステップ 2	画面に表示されたサンプルポリシーの内容を、テキストエディタにカットアンドペーストします。	—
ステップ 3	必要な event_register Tcl コマンド拡張を定義します。	検出するイベントについて、適切な event_register Tcl コマンド拡張を選択し、ポリシーに追加します。有効なイベント登録 Tcl コマンド拡張を次に示します。 • event_register_appl • event_register_counter • event_register_stat • event_register_wdsysmon • event_register_oir • event_register_process • event_register_syslog • event_register_timer • event_register_timer_subscriber • event_register_hardware • event_register_none
ステップ 4	適切な名前空間を、::cisco 階層構造に追加します。	ポリシーの開発者は、Cisco IOS XR EEMによって使用されるすべての拡張をグループ化するため、Tclポリシーで新しい名前空間 ::cisco を使用できま

	コマンドまたはアクション	目的
		す。::cisco 階層の下には、2 つの名前空間があります。各名前空間に属する名前空間と EEM Tcl コマンド拡張カテゴリは次のとおりです。 • ::cisco::eem • EEM イベント登録 • EEM イベント情報 • EEM イベント パブリッシュ • EEM アクション • EEM ユーティリティ • EEM コンテキスト ライブラリ • EEM システム情報 • CLI ライブラリ • ::cisco::lib • SMTP ライブラリ （注） 適切な名前空間がインポートされていることを確認するか、前述のコマンドを使用する場合は修飾されたコマンド名を使用します。
ステップ 5	Must Define セクションをプログラムし、このポリシーで使用される各環境変数をチェックします。	この手順は任意です。Must Define は、ポリシーによって必要とされるすべての EEM 環境変数が、回復アクションの実行前に定義されているかどうかをテストする、ポリシーのセクションです。ポリシーによって EEM 環境変数が使用されない場合、Must Define セクションは不要です。EEM スクリプトの EEM 環境変数は、ポリシーの実行前にポリシーに対して外部定義された Tcl グローバル変数です。EEM 環境変数を定義するには、EEM コンフィギュレーション コマンド **event manager environment** を使用します。規則とし

	コマンドまたはアクション	目的
		て、すべてのシスコ EEM 環境変数の先頭は、「_」（アンダースコア）になっています。将来的な競合を避けるため、「_」で始まる新しい変数を定義しないことを推奨します。 （注） **show event manager environment** コマンドを使用して、システムの Embedded Event Manager 環境変数セットを表示できます。 たとえば、サンプルポリシーで定義されている EEM 環境変数には、電子メール変数が含まれます。適切に動作させるためには、電子メールを送信するサンプルポリシーに、次に示す変数が設定されている必要があります。EEM サンプルポリシーで使用される電子メール特有の環境変数について説明します。 • **_email_server**：電子メール送信に使用されるシンプル メール転送プロトコル（SMTP）メール サーバ（たとえば mailserver.example.com） • **_email_to**：電子メールの送信先アドレス（たとえば engineering@example.com） • **_email_from**：電子メールの送信元アドレス（たとえば devtest@example.com） • **_email_cc**：電子メールのコピーの送信先アドレス（たとえば manager@example.com）
ステップ 6	スクリプトの本体をプログラムします。	スクリプトのこのセクションでは、次のいずれかを定義できます。 • 検出されたイベントに関する情報の EEM への問い合わせに使用される **event_reqinfo** イベント情報の Tcl コマンド拡張。

	コマンドまたはアクション	目的
		• EEM 特有のアクションの指定に使用される、**action_syslog** などのアクション Tcl コマンド拡張。 • 一般的なシステム情報の取得に使用される、**sys_reqinfo_routername** などのシステム情報の Tcl コマンド拡張。 • 他のポリシーによって使用される Tcl 変数の保存に使用される **context_save** および **context_retrieve** の Tcl コマンド拡張。 • ポリシーからの、SMTP ライブラリ（電子メール通知を送信）または CLI ライブラリ（CLI コマンドを実行）の使用。
ステップ 7	開始ステータスをチェックし、ポリシーがこのイベントに対して前に実行されたかどうかを判断します。	前のポリシーが正常終了した場合、現在のポリシーは、実行が必要な場合と、実行が不要な場合があります。開始ステータス指定には、0（前のポリシーが正常終了した）、Not=0（前のポリシーに障害が発生した）、および Undefined（実行された前のポリシーがない）の、3 つの値のうちいずれか 1 つを使用できます。
ステップ 8	終了ステータスをチェックし、デフォルトアクションが存在する場合に、このイベントのデフォルトアクションが適用されたかどうかを判断します。	値 0 は、デフォルトのアクションを実行しないことを意味します。0 以外の値は、デフォルトのアクションを実行する必要があることを意味します。終了ステータスは、同じイベントで実行される後続ポリシーに渡されます。
ステップ 9	Cisco エラー番号（_cerrno）の Tcl グローバル変数を設定します。	一部の EEM Tcl コマンド拡張によって、Cisco エラー番号の Tcl グローバル変数の _cerrno が設定されます。_cerrno が設定されるたびに、他の 4 つの Tcl グローバル変数が _cerrno から分岐し、それとともに設定されます（_cerr_sub_num、_cerr_sub_err、および _cerr_str）。

	コマンドまたはアクション	目的
ステップ 10	新しいファイル名で Tcl スクリプトを保存し、Tcl スクリプトをルータにコピーします。	Embedded Event Manager ポリシー ファイル名は、次の仕様に従っています。 • オプションのプレフィックス Mandatory. がある場合、これは、システム ポリシーがまだ登録されていない場合に、自動的に登録される必要があるシステム ポリシーであることを示します。たとえば、Mandatory.sl_text.tcl などです。 • 指定された 1 つめのイベントの 2 文字の省略形が含まれるファイル名の本体部（表 12 : 2 文字の省略形の指定 （57 ページ） を参照）、下線文字部、および、ポリシーをさらに示す説明フィールド部。 • ファイル名拡張子部は .tcl と定義されます。 詳細については、Embedded Event Manager 用のシスコ ファイル命名規則（56 ページ）を参照してください。 ルータ上のフラッシュ ファイル システム（通常は disk0:）に、ファイルをコピーします。
ステップ 11	**configure**	グローバル コンフィギュレーション モードを開始します。
ステップ 12	**event manager directory user** { **library** *path* | **policy** *path*} 例： `RP/0/RP0/CPU0:Router(config)# event manager directory user library disk0:/user_library`	ユーザ ライブラリ ファイルまたはユーザ定義 EEM ポリシーの保存に使用するディレクトリを指定します。
ステップ 13	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]] 例： `RP/0/RP0/CPU0:Router(config)# event`	ポリシー内で定義された指定イベントが発生した場合に、EEM ポリシーを実行するよう、定義します。

	コマンドまたはアクション	目的
	`manager policy test.tcl username user_a type user`	
ステップ 14	**commit**	
ステップ 15	ポリシーを実行し、ポリシーを観察します。	—
ステップ 16	ポリシーが正しく実行されていない場合、デバッグのテクニックを使用します。	—

EEM ユーザ Tcl ライブラリ索引の作成

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

手順

	コマンドまたはアクション	目的

ステップ 1	ワークステーション（UNIX、Linux、PC、または Mac）で、ライブラリ ディレクトリを作成し、Tcl ライブラリ ファイルをディレクトリにコピーします。	次の例ファイルを使用すると、Tcl シェルが実行されているワークステーション上で、tclIndex を作成できます。
		lib1.tcl
		`proc test1 {} {`
		`  puts "In procedure test1"`
		`}`
		`proc test2 {} {`
		`  puts "In procedure test2"`
		`}`
		lib2.tcl
		`proc test3 {} {`
		`  puts "In procedure test3"`
		`}`

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

	コマンドまたはアクション	目的

ステップ 3	**auto_mkindex** *directory_name* ***.tcl**
	例：
	`workstation% auto_mkindex eem_library *.tcl`
		auto_mkindex コマンドを使用して、tclIndex ファイルを作成します。すべての手順のディレクトリが含まれる tclIndex ファイルは、Tcl ライブラリ ファイルに含まれていました。どのディレクトリにも 1 つの tclIndex ファイルのみを存在させることができ、他の Tcl ファイルはグループ化しておくことが可能であるため、ディレクトリ内で **auto_mkindex** を実行することを推奨します。ディレクトリ内で **auto_mkindex** を実行すると、特定の tclIndex を使用してどの Tcl ソース ファイルを索引化できるかが判断されます。
		lib1.tcl ファイルと lib2.tcl ファイルがライブラリ ファイル ディレクトリにあり、**auto_mkindex** コマンドが実行されたときに、次の例に示す tclIndex が作成されます。
		tclIndex
		`# Tcl autoload index file, version 2.0`
		`# This file is generated by the "auto_mkindex" command`
		`# and sourced to set up indexing information for one or`
		`# more commands. Typically each line is a command that`
		`# sets an element in the auto_index array, where the`
		`# element name is the name of a command and the value is`
		`# a script that loads the command.`
		`set auto_index(test1) [list source [file join $dir lib1.tcl]]`
		`set auto_index(test2) [list source [file join $dir lib1.tcl]]`
		`set auto_index(test3) [list source [file join $dir lib2.tcl]]`

ステップ 4	ターゲット ルータ上のユーザ ライブラリ ファイルの保存に使用されるディレクトリに、ステップ 1 から Tcl ライブラリ ファイルをコピーし、ステップ 3 から tclIndex ファイルをコピーします。	—

	コマンドまたはアクション	目的

ステップ 5	Tcl で記述されたユーザ定義 EEM ポリシーファイルを、ターゲットルータ上でユーザ定義 EEM ポリシーの保存に使用されるディレクトリにコピーします。	ディレクトリは、ステップ 4 で使用されるディレクトリと同じディレクトリを使用できます。
		次に、EEM でサポートされる Tcl ライブラリのテストに、ユーザ定義 EEM ポリシーを使用できる例を示します。
		libtest.tcl
		`::cisco::eem::event_register_none`
		`namespace import ::cisco::eem::*`
		`namespace import ::cisco::lib::*`
		`global auto_index auto_path`
		`puts [array names auto_index]`
		`if { [catch {test1} result]} {`
		`  puts "calling test1 failed result = $result $auto_path"`
		`}`
		`if { [catch {test2} result]} {`
		`  puts "calling test2 failed result = $result $auto_path"`
		`}`
		`if { [catch {test3} result]} {`
		`  puts "calling test3 failed result = $result $auto_path"`
		`}`

ステップ 6	**configure**	
ステップ 7	**event manager directory user library** *path* 例： `RP/0/RP0/CPU0:Router(config)# event manager directory user library disk0:/eem_library`	EEM ユーザ ライブラリのディレクトリを指定します。これは、ステップ 4 のファイルがコピーされたディレクトリです。
ステップ 8	**event manager directory user policy** *path* 例： `RP/0/RP0/CPU0:Router(config)# event manager directory user policy disk0:/eem_policies`	EEM ユーザ ポリシーのディレクトリを指定します。これは、ステップ 5 のファイルがコピーされたディレクトリです。
ステップ 9	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]] 例： `RP/0/RP0/CPU0:Router(config)# event manager policy libtest.tcl username user_a`	ユーザ定義の EEM ポリシーを登録します。

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

EEM ユーザ Tcl パッケージ索引の作成

すべての Tcl パッケージのディレクトリと、Tcl パッケージ ファイルのライブラリに含まれるバージョン情報が含まれる、Tcl パッケージの索引ファイルを作成するには、この作業を実行します。Tcl パッケージは、Tcl **package** キーワードを使用してサポートされます。

Tcl パッケージは、EEM システム ライブラリ ディレクトリまたは EEM ユーザ ライブラリ ディレクトリのいずれかにあります。**package require** Tcl コマンドが実行されると、ユーザ ライブラリ ディレクトリで、まず、pkgIndex.tcl ファイルが検索されます。pkgIndex.tcl ファイルがユーザディレクトリで見つからない場合、システム ライブラリ ディレクトリが検索されます。

この作業では、**pkg_mkIndex** コマンドを使用して、適切なライブラリ ディレクトリに Tcl パッケージ ディレクトリ pkgIndex.tcl ファイルが作成され、バージョン情報とともに、ディレクトリに含まれるすべての Tcl パッケージについての情報が含められます。索引が作成されない場合、**package require** Tcl コマンドが含まれる EEM ポリシーが実行されたときに、Tcl パッケージは見つかりません。

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

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

手順

	コマンドまたはアクション	目的
ステップ 1	ワークステーション（UNIX、Linux、PC、または Mac）で、ライブラリ ディレクトリを作成し、Tcl パッケージ ファイルをディレクトリにコピーします。	–
ステップ 2	**tclsh** 例： `workstation% tclsh`	Tcl シェルを開始します。

	コマンドまたはアクション	目的

ステップ 3	**pkg_mkindex** *directory_name* ***.tcl**
	例：
	`workstation% pkg_mkindex eem_library *.tcl`
		pkg_mkindex コマンドを使用して、pkgIndex ファイルを作成します。すべてのパッケージのディレクトリが含まれる pkgIndex ファイルは、Tcl ライブラリ ファイルに含まれていました。どのディレクトリにも 1 つの pkgIndex ファイルのみを存在させることができ、他の Tcl ファイルはグループ化しておくことが可能であるため、ディレクトリ内で**pkg_mkindex** コマンドを実行することを推奨します。ディレクトリ内で**pkg_mkindex** コマンドを実行すると、特定の pkgIndex を使用してどの Tcl パッケージ ファイルを索引化できるかが判断されます。
		次に、いくつかの Tcl パッケージがライブラリ ファイル ディレクトリにあり、pkg_mkindex コマンドが実行されたときに、pkgIndex が作成される例を示します。
		pkgIndex
		`# Tcl package index file, version 1.1`
		`# This file is generated by the "pkg_mkIndex" command`
		`# and sourced either when an application starts up or`
		`# by a "package unknown" script. It invokes the`
		`# "package ifneeded" command to set up package-related`
		`# information so that packages will be loaded automatically`
		`# in response to "package require" commands. When this`
		`# script is sourced, the variable $dir must contain the`
		`# full path name of this file's directory.`
		`package ifneeded xmlrpc 0.3 [list source [file join $dir xmlrpc.tcl]]`

ステップ 4	ターゲット ルータ上のユーザ ライブラリ ファイルの保存に使用されるディレクトリに、ステップ 1 から Tcl パッケージ ファイルをコピーし、ステップ 3 から pkgIndex ファイルをコピーします。	—

	コマンドまたはアクション	目的

ステップ 5	Tcl で記述されたユーザ定義 EEM ポリシーファイルを、ターゲットルータ上でユーザ定義 EEM ポリシーの保存に使用されるディレクトリにコピーします。	ディレクトリは、ステップ 4 で使用されるディレクトリと同じディレクトリを使用できます。
		次に、EEM でサポートされる Tcl ライブラリのテストに、ユーザ定義 EEM ポリシーを使用できる例を示します。
		packagetest.tcl
		`::cisco::eem::event_register_none maxrun 1000000.000`
		`#`
		`# test if xmlrpc available`
		`#`
		`#`
		`# Namespace imports`
		`#`
		`namespace import ::cisco::eem::*`
		`namespace import ::cisco::lib::*`
		`#`
		`package require xmlrpc`
		`puts "Did you get an error?"`

ステップ 6	**configure**	
ステップ 7	**event manager directory user library** *path* 例： `RP/0/RP0/CPU0:Router(config)# event manager directory user library disk0:/eem_library`	EEM ユーザ ライブラリのディレクトリを指定します。これは、ステップ 4 のファイルがコピーされたディレクトリです。
ステップ 8	**event manager directory user policy** *path* 例： `RP/0/RP0/CPU0:Router(config)# event manager directory user policy disk0:/eem_policies`	EEM ユーザ ポリシーのディレクトリを指定します。これは、ステップ 5 のファイルがコピーされたディレクトリです。
ステップ 9	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]] 例： `RP/0/RP0/CPU0:Router(config)# event manager policy packagetest.tcl username user_a`	ユーザ定義の EEM ポリシーを登録します。
ステップ 10	**event manager run** *policy* [*argument*] 例：	手動で EEM ポリシーを実行します。

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

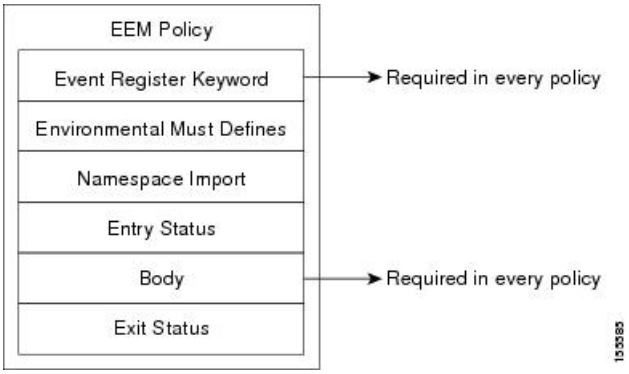
TCL を使用した EEM ポリシー：詳細

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

Tcl ポリシーの構造と要件

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

図 *3 : Tcl* ポリシーの構造と要件



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

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

次に、一部の環境変数をチェックし、定義する Tcl コードの例を示します。

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

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

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

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

ポリシーの本体は必須の構造で、次のものを含めることができます。

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

EEM 開始ステータス

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

開始ステータスの指定には、次の 3 種類の値のいずれかを使用できます。

- 0（前のポリシーが正常終了した）
- Not=0（前のポリシーに障害が発生した）
- Undefined（実行された前のポリシーがない）

EEM 終了ステータス

ポリシーでそのコードの実行を終了すると、終了値が設定されます。終了値は、EEM によって使用され、このイベントのデフォルトアクションがある場合に、それが適用されたかどうかが判断されます。値 0 は、デフォルトのアクションを実行しないことを意味します。0 以外の値は、デフォルトのアクションを実行する必要があることを意味します。終了ステータスは、同じイベントで実行される後続ポリシーに渡されます。

EEM ポリシーと Cisco エラー番号

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

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

```
XYSSSSSSSSSSSSSEEEEEEEEPPPPPPPPP
```

次の表に示されているように、この 32 ビットの整数は変数に分けられます。

表 *17*：*_cerrno*：*32* ビット エラー戻り値の変数

変数	説明
XY	エラー クラス（エラーの重大度を示します）。この変数は、32 ビットのエラー戻り値の最初の 2 ビットに対応しています。前述のケースの 10 は、CERR_CLASS_WARNING を示します。 この変数特有の 4 つのエラー クラス エンコーディングについては、表 18：エラー クラス エンコーディング （81 ページ） を参照してください。
SSSSSSSSSSSSS	最新のエラーが生成されたサブシステム番号（13 ビット＝値 8192）。これは、32 ビット シーケンスの次の 13 ビットで、その整数値は $_cerr_sub_num に含まれています。
EEEEEEEE	サブシステム固有のエラー番号（8 ビット＝値 256）。このセグメントは、32 ビット シーケンスの次の 8 ビットで、このエラー番号に対応する文字列は、$_cerr_sub_err に含まれています。

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

```
862439AE
```

この数字は、次の 32 ビット値として解釈されます。

```
10000110001001000011100110101110
```

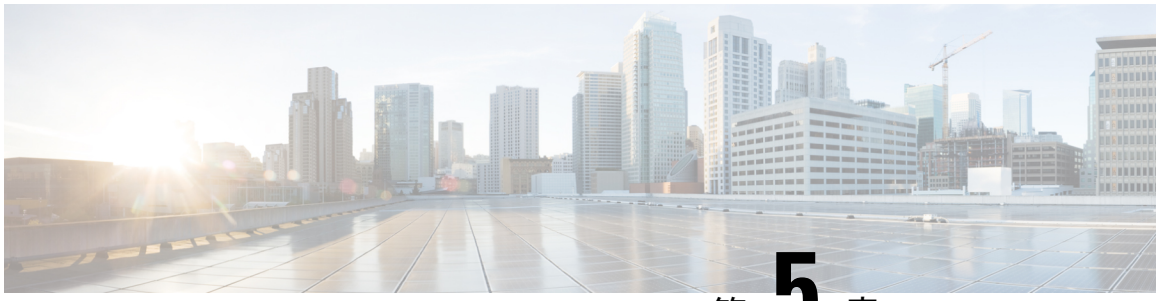
変数 XY は、次の表に示されているように、エラー クラス エンコーディングを参照しています。

表 *18*：エラー クラス エンコーディング

エラー戻り値	エラー クラス
00	CERR_CLASS_SUCCESS

エラー戻り値	エラー クラス
01	CERR_CLASS_INFO
10	CERR_CLASS_WARNING
11	CERR_CLASS_FATAL

ゼロのエラー戻り値は、SUCCESS を示します。



第 5 章

IP サービス レベル契約の実装

IP サービス レベル契約（IP SLA）は、Cisco IOS XR ソフトウェアを実行するほとんどのデバイスに組み込まれているテクノロジーのポートフォリオであり、ユーザはネットワーク アセスメントの実行、Quality of Service（QoS）の検証、新しいサービスの容易な展開、ネットワークのトラブルシューティングに関する管理者の支援を行うことができます。

この章では、この機能の詳細と、この機能の設定に必要なさまざま手順について説明します。

この章では、次のトピックについて取り上げます。

IP サービス レベル契約テクノロジーの概要

IP SLA は、連続的で、信頼でき、予測可能な方法でトラフィックを生成する、アクティブトラフィック モニタリングを使用してネットワークのパフォーマンスを測定します。IP SLA はネットワークにデータを送信し、複数のネットワーク間あるいは複数のネットワーク パス内のパフォーマンスを測定します。ネットワーク データおよびIPサービスをシミュレーションし、ネットワーク パフォーマンス情報をリアル タイムで収集します。次の情報が収集されます。

- 応答時間
- 単方向遅延、ジッター（パケット間の遅延のばらつき）
- パケット損失
- ネットワーク リソースのアベイラビリティ

IP SLA は、ルータ間またはルータとネットワーク アプリケーション サーバなどのリモート IP デバイスの間で、トラフィックを生成および分析してパフォーマンスを測定することにより、アクティブ モニタリングを行います。さまざまな IP SLA 動作によって得られる測定統計情報は、トラブルシューティング、問題分析、ネットワーク トポロジの設計に使用します。

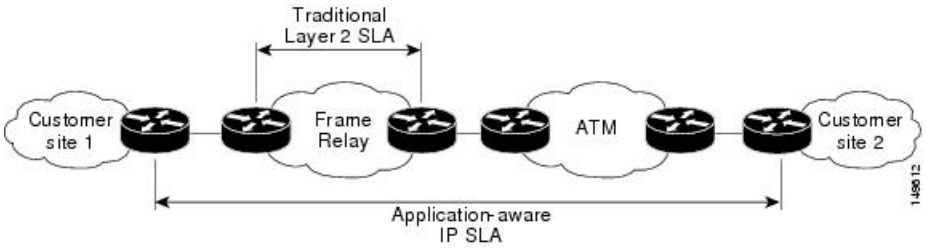
この項では、次のトピックについて取り上げます。

サービス レベル契約

インターネット ショッピングはこの数年で急激に成長し、テクノロジーの進化により高速で信頼性の高いインターネット アクセスが提供されるようになりました。多くの機能がオンライン アクセスを必要とし、その業務のほとんどをオンラインで行っており、サービスが失われると企業の収益に影響を与えることがあります。インターネット サービス プロバイダー（ISP）だけでなく、社内の IT 部門も、定義されたサービス レベル（サービス レベル契約）を提供し、顧客にある程度の予測性を提供するようになっています。

ネットワーク管理者は、アプリケーション ソリューションをサポートするサービス レベル契約をサポートする必要があります。図 4 : 従来のサービス レベル契約と IP SLA の範囲 （84 ページ）に、アプリケーションのサポートも含め、エンドツーエンドのパフォーマンス測定をサポートするために、IP SLA がどのように従来のレイヤ 2 サービス レベル契約の概念を取り込み、より広い範囲に適用しているかを示します。

図 *4 :* 従来のサービス レベル契約と *IP SLA* の範囲



次の表に、IP SLA の従来のサービス レベル契約に対する改良点の一覧を示します。

表 *19 :* 従来のサービス レベル契約に対する *IP SLA* の改良点

改良の種類	説明
エンドツーエンド測定	ネットワークの端からもう一方の端までパフォーマンスを測定できることにより、エンドユーザによるネットワーク利用状況をより広い到達範囲でより正確に表現できます。
詳細化	遅延、ジッター、パケット シーケンス、レイヤ 3 接続、パスとダウンロード時間などの双方向のラウンドトリップの数値に詳細化される統計情報により、レイヤ2リンクの帯域幅だけよりも詳細なデータが得られます。
精度	ネットワーク パフォーマンスのわずかな変化にも影響を受けやすいアプリケーションは、ミリ秒以下の単位での IP SLA の測定精度を必要とします。
展開の簡易化	IP SLA は、大きいネットワーク内で既存のシスコ デバイスを活用することにより、従来のサービス レベル契約で必要になることが多い物理的な動作よりも、簡単に実装されます。

改良の種類	説明
アプリケーション認識型の監視	IP SLA は、レイヤ 3 からレイヤ 7 の上で動作するアプリケーションによって生成されたパフォーマンス統計情報をシミュレートおよび測定できます。従来のサービス レベル契約では、レイヤ 2 パフォーマンスしか測定できません。
普及	IP SLA は、ローエンドからハイエンドまでのルータとスイッチに及ぶ、シスコ ネットワーキング デバイスでサポートされています。この幅広い展開により、IP SLA は、従来のサービス レベル契約よりも高い柔軟性を備えています。

IP サービス レベル契約の利点

次の表に、IP SLA を実装することの利点の一覧を示します。

表 *20 : IP SLA* の利点の一覧

利点	説明
IP SLA のモニタリング	サービス レベル契約モニタリング、評価、および検証の提供
ネットワーク パフォーマンス モニタリング。	ネットワーク内のジッター、遅延、パケット損失が測定できる。また、IP SLA は、予防的な通知に加えて、連続的で、信頼性が高く、予測可能な測定を提供します。
IP サービス ネットワーク稼働状態評価	既存の QoS が新しい IP サービスに対して十分であることの検証
ネットワーク動作のトラブルシューティング	問題をただちに特定し、トラブルシューティング時間を節約する、一貫し、信頼性が高い測定を提供します。

IP サービス レベル契約を実装する前提条件

一般的なネットワーキング プロトコルおよび特定のネットワーク設計の知識が必要です。ネットワーク管理アプリケーションについての知識が役立ちます。すべての動作を同時にスケジューリングすると、パフォーマンスに悪影響を及ぼすおそれがあるため、お勧めしません。

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

IP サービス レベル契約の実装の制限

- Cisco IOS XR ソフトウェアでサポートされている IP SLA 動作の最大数は 2048 です。
- Cisco IOS XR ソフトウェアでサポートされている IP SLA 設定可能操作の最大数は 2000 です。
- 同じ開始時刻にすべての操作をスケジューリングすることは、パフォーマンスに影響する可能性があるため、推奨しません。開始時間が同じ場合は、1秒あたりの操作数を10以下にスケジューリングする必要があります。`start after` コンフィギュレーションを使用することをお勧めします。

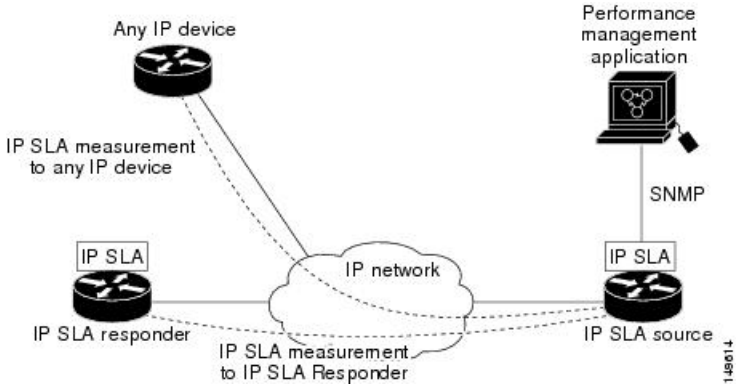
（注） 頻度を 60 秒未満に設定すると、送信されるパケット数が増加します。しかし、スケジュールされた動作の開始時刻が同じ場合、IP SLA 動作のパフォーマンスに悪影響を与える可能性があります。

- IP SLA は HA に対応していません。
- frequency、timeout、および threshold コマンドを設定する前に、次のガイドラインを検討してください。

IPサービス レベル契約によるネットワーク パフォーマンスの測定

IP SLA は、ルータなどの 2 台のネットワーキング デバイス間のネットワーク パフォーマンスを測定するために、生成されたトラフィックを使用します。図 5 : IP SLA の動作 （87 ページ）に、IP SLA デバイスが宛先デバイスに生成パケットを送信するときに IP SLA が開始される手順を示します。宛先デバイスがパケットを受信した後、動作が IP SLA コンポーネントを受信側（たとえば IP SLA レスポンダ）で使用している場合、応答パケットにはターゲット デバイスでの遅延に関する情報が含まれています。送信元デバイスはこの情報を使用して測定の精度を向上させます。IP SLA 動作は、動作にユーザ データグラム プロトコル（UDP）などの特定のプロトコルを使用した、送信元デバイスからネットワークの宛先へのネットワーク測定です。

図 *5 : IP SLA* の動作



IP SLA ネットワーク パフォーマンス測定を実現するためには、次のタスクを実行します。

1. 必要に応じて IP SLA レスポンダをイネーブルにします。
2. 必要な IP SLA 動作タイプを設定します。
3. 指定された IP SLA 動作タイプのオプションを設定します。
4. 必要であれば、応答条件を設定します。
5. 動作の実行をスケジュールします。その後、一定の時間動作を実行して統計情報を収集します。
6. Cisco IOS-XR ソフトウェアの CLI または XML を使用して、または NMS システムで SNMP を使用して、動作の結果を表示および解釈します。

ここでは、次の内容について説明します。

IP SLA レスポンダおよび IP SLA 制御プロトコル

IP SLA レスポンダは宛先シスコ ルーティング デバイスに組み込まれたコンポーネントで、システムが IP SLA 要求パケットを予想して応答します。IP SLA レスポンダは、高い測定精度を提供します。特許取得済みの IP SLA 制御プロトコルは、IP SLA レスポンダによって使用され、応答側がどのポートで待ち受けと応答を行うか応答側に通知するメカニズムを提供します。Cisco IOS XR ソフトウェア デバイスまたはその他のシスコ プラットフォームのみが宛先 IP SLA レスポンダの送信元になることができます。

図 5 : IP SLA の動作 （87 ページ）に、IP SLA レスポンダが IP ネットワークのどこに適しているかを示します。IP SLA レスポンダは、IP SLA 動作から送信されたコントロール プロトコル メッセージを指定されたポートで受信します。レスポンダは、制御メッセージを受信すると、制御メッセージで指定された UDP ポートを指定された期間イネーブルにします。この間に、レスポンダは要求を受け付け、応答します。応答側は、IP SLA パケットに応答した後、あるいは指定された期間が経過すると、ポートをディセーブルにします。セキュリティを強化するために、コントロール メッセージの MD5 認証も使用できます。

（注） IP SLA レスポンダは、ソケットを開いてローカル パケット転送サービス（LPTS）をプログラムするために少なくとも 1 秒必要です。したがって、IP SLA タイムアウトを少なくとも 2000 ミリ秒に設定します。

UDP ジッター操作では IP SLA レスポンダを使用する必要があります。ターゲット ルータですでに提供されているサービスが選択された場合、IP SLA レスポンダをイネーブルにする必要はありません。シスコ以外のデバイスの場合は、IP SLA レスポンダを設定できず、IP SLA は、それらのデバイスにネイティブなサービスのみに動作パケットを送信できます。

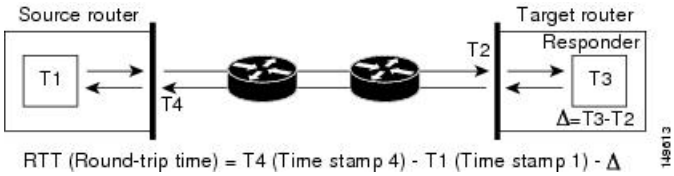
IP SLA の応答時間の計算

他の優先順位が高いプロセスにより、ルータは着信パケットを処理するために数十ミリ秒を要します。テストパケットへの応答が処理を待つ間キューに格納される可能性があるため、この遅延は応答時間に影響を与えます。この場合、応答時間は正しいネットワーク遅延を反映しません。IP SLA は、送信元ルータとターゲット ルータ上でこれらの処理遅延を最小化し、真のラウンドトリップ遅延を判定します（IP SLA レスポンダが使用されている場合）。一部の IP SLA プローブ パケットには、測定をより正確にするために、最終的な計算で使用される遅延情報が含まれています。

IP SLA レスポンダをイネーブルにすると、ターゲット デバイスは、パケットがインターフェイスに到着した時点と出て行く時点の両方でタイムスタンプを取得し、統計情報を計算するときにそれを考慮します。このタイムスタンプ処理は、ミリ秒以下の単位で行われます。

図 6 : IP SLA レスポンダのタイムスタンプ処理 （88 ページ） に、レスポンダの動作を示します。T3 は、応答パケットが IP SLA レスポンダ ノードで送信された時刻であり、T1 は送信元ノードで要求が送信された時刻です。RTT を算出するためのタイム スタンプが 4 つ付けられます。ターゲット ルータでレスポンダ機能がイネーブルの場合、タイム スタンプ 3（TS3）からタイム スタンプ 2（TS2）を引いてテスト パケットの処理にかかった時間を求め、デルタ（Δ）で表します。次に全体の RTT からこのデルタの値を引きます。精度を高めるため、優先度が高いパスで着信タイムスタンプ 4（TS4）が取得される送信元ルータ上で、同じ原理が IP SLA によって適用されることに注意してください。

図 *6 : IP SLA* レスポンダのタイムスタンプ処理



IP SLA 動作のスケジューリング

IP SLA 動作の設定が完了したら、その動作をスケジューリングして、統計情報の取得とエラー情報の収集を開始する必要があります。動作をスケジューリングすると、動作はただちに開始されるか、特定の月の特定の日に開始されます。また、動作を保留状態にすることができま

す。これは、その動作が、トリガーされるのを待つ反応（しきい値）動作である場合に使用されます。IP SLA 動作の通常のスケジューリングでは、一度に 1 つの動作をスケジューリングできます。

Two-Way Active Measurement Protocol（TWAMP）

Two-Way Active Measurement Protocol（TWAMP）は、任意の 2 つのデバイス間のラウンドトリップ IP パフォーマンスを測定し、それにより IP SLA コンプライアンスをチェックする柔軟な方法を定義します。

TWAMP の利点

- TWAMP によって、完全な IP パフォーマンス測定が可能になります。
- TWAMPはネットワークに展開されたすべてのデバイスをサポートしているため、ソリューションを柔軟に選択できます。

ここでは、次の内容について説明します。

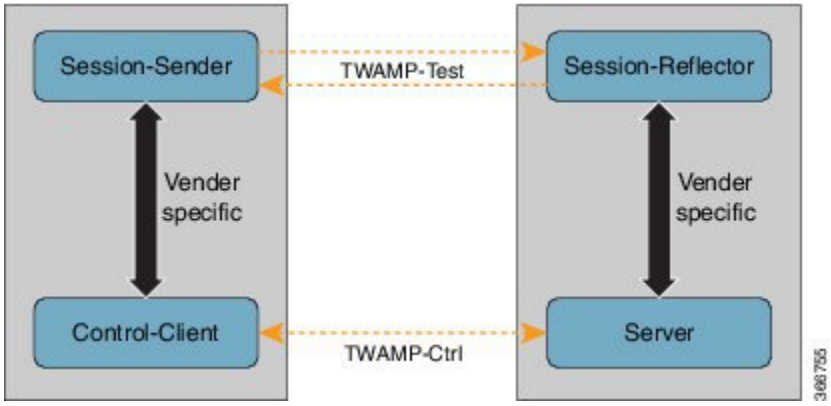
TWAMP エンティティ

TWAMP システムは、4 つの論理エンティティで構成されています。

- サーバ：1 つ以上の TWAMP セッションを管理し、エンドポイント内のセッションごとのポートも構成します。
- セッション リフレクタ：TWAMP テスト パケットを受信するとすぐに測定パケットを反映します。
- 制御クライアント：TWAMP テスト セッションの開始と停止を開始します。
- セッション送信元：セッション リフレクタに送信された TWAMP テスト パケットをインスタンス化します。

次の図は、TWAMP が 2 つの個別のホスト上で動作する TWAMP の実装を示しています。一方は制御クライアントとセッション送信元の役割を果たし、もう一方はサーバとセッション リフレクタの役割を果たします。ルータでは、セッション サーバおよびセッション リフレクタの機能のみサポートされています。TWAMP を使用すると、基礎となるトランスポートの IP パフォーマンスを、TWAMP サポートを含むネットワーク要素間の協力によって測定できます。

図 *7: TWAMP* エンティティ



TWAMP プロトコル

TWAMP プロトコルには、次の 3 つの異なるメッセージ交換カテゴリが含まれています。

- **接続セットアップ交換**：メッセージは、制御クライアントとサーバ間にセッション接続を確立します。最初に、通信ピアの ID が、チャレンジ応答メカニズムを介して確立されます。サーバがランダムに生成されたチャレンジを送信し、次に制御クライアントが共有秘密から派生したキーを使用してチャレンジを暗号化して応答を送信します。ID が確立された後、次のステップでは、後続の TWAMP 制御コマンドおよび TWAMP テスト ストリーム パケットをバインドするセキュリティ モードがネゴシエートされます。

  （注） サーバは、複数の制御クライアントからの接続要求を受け入れることができます。

- **TWAMP 制御交換**：TWAMP 制御プロトコルは TCP を介して実行され、測定セッションのインスタンス化および制御に使用されます。接続セットアップ交換とは異なり、TWAMP 制御コマンドは複数回送信できます。ただし、session-start コマンドの前に複数の request-session コマンドを送信することはできますが、メッセージの順序が乱れることはありません。コマンドのシーケンスは、次のとおりです。
  - request-session
  - start-session
  - stop-session

- **TWAMP テスト ストリーム交換**：TWAMP テストは UDP を介して実行され、セッション送信者とセッション リフレクタの間で TWAMP テスト パケットを交換します。これらのパケットには、パケットの出力と入力の瞬間を含むタイムスタンプ フィールドが含まれています。さらに、各パケットには、送信者（セッション送信元またはセッション リフレクタ）と外部の時刻源（GPS や NTP など）との同期の誤差を示すエラー推定値が含まれます。パケットにはシーケンス番号も含まれます。

TWAMP 制御および TWAMP テスト ストリームには、未認証、認証済み、および暗号化済みの 3 つのセキュリティ モードがあります。

ルータでの TWAMP の制限事項

- このルータでは、セッション サーバおよびセッション リフレクタの機能のみサポートされています。
- より精度の高いハードウェア タイムスタンプ機能はサポートされていません。
- ポート番号が 57343 より大きい値の場合、Twamp サーバ制御セッションはルータ上で作成されません。

ルータでの TWAMP の設定

セッション サーバの設定

```
Router# configure
Router(config)# ipsla server twamp
Router(config-ipsla-server-twamp)# port 862
Router(config-ipsla-server-twamp)# commit
```

セッション リフレクタの設定

```
Router# configure
Router(config)# ipsla responder twamp
Router(config-twamp-ref)# commit
```

実行コンフィギュレーション

```
ipsla
 responder
  twamp
  !
 !
 server twamp
  port 862
 !
!
```

TWAMP の確認

TWAMP 機能のステータスは、show ipsla twamp status コマンドを使用して確認できます。

```
Router# show ipsla twamp status
Thu Aug 17 12:42:38.923 IST
TWAMP Server is enabled
TWAMP Server port : 862
TWAMP Reflector is enabled
```

TWAMP セッションは、show ipsla twamp session コマンドを使用して検証できます。

```
Router# show ipsla twamp session
IP SLAs Responder TWAMP is: Enabled
```

```
Recvr Addr: 10.5.139.11
Recvr Port: 7222
Sender Addr: 172.27.111.233
Sender Port: 33243
Session Id: 10.5.139.11:70929508:88F7A620
Connection Id: 0
```

送信元 IP アドレスに基づく TWAMP テスト セッションは、**show ipsla twamp session source-ip <source ip-address> source-port <source port-number>** コマンドを使用して検証できます。

```
RP/0/0/CPU0:ios# show ipsla twamp session source-ip 172.27.111.233 source-port 33286
IP SLAs Responder TWAMP is: Enabled
Recvr Addr: 10.5.139.11
Recvr Port: 6198
Sender Addr: 172.27.111.233
Sender Port: 33286
Session Id: 10.5.139.11:71804476:F2721505
Connection Id: D
Mode: UnAuthorized
DSCP: 0
Pad Length: 0
Number of Packets Received: 8867
```