Cisco ASR 9000 シリーズ アグリゲーション サービス ルータ システム モニタリング コンフィギュレーション ガイド リリース 4.3.x

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/

Text Part Number: OL-28386-01-J

【注意】シスコ製品をご使用になる前に、安全上の注意（**www.cisco.com/jp/go/safety_warning/**）をご確認ください。本書は、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。また、契約等の記述については、弊社販売パートナー、または、弊社担当者にご確認ください。

目 次

はじめに

『*Cisco ASR 9000* シリーズ アグリゲーション サービス ルータ システム モニタリング コンフィギュレーション ガイド』の「はじめに」には、次の項が含まれています。

マニュアルの変更履歴

表 1：マニュアルの変更履歴に、初版後、このマニュアルに加えられた技術的な変更の履歴を示します。

表 *1*：マニュアルの変更履歴

リビジョン	日付	変更点
OL-28386-01-J	2012 年 12 月	このマニュアルの初回リリース

マニュアルの入手方法およびテクニカル サポート

マニュアルの入手方法、テクニカル サポート、その他の有用な情報について、次の URL で、毎月更新される『*What's New in Cisco Product Documentation*』を参照してください。シスコの新規および改訂版の技術マニュアルの一覧も示されています。

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

『*What's New in Cisco Product Documentation*』は RSS フィードとして購読できます。また、リーダー アプリケーションを使用してコンテンツがデスクトップに直接配信されるように設定することもできます。RSS フィードは無料のサービスです。シスコは現在、RSS バージョン 2.0 をサポートしています。

第 1 章

Cisco IOS XR リリース 4.3.x の新機能および変更された機能の情報

次の表は、『*Cisco ASR 9000* シリーズ アグリゲーション サービス ルータ システム モニタリング コンフィギュレーション ガイド』の新機能および変更された機能の情報を要約し、その参照先を示しています。

追加または変更されたシステム モニタリング機能

機能	説明	導入/変更されたリリース	参照先
非同期 Syslog 通信	この機能が導入されました。	リリース *4.3.0*	アラームおよびアラーム ログ相関の実装とモニタリング 非同期 Syslog 通信 非同期 Syslog 通信を確認するために使用するコマンドの情報については、『*Cisco ASR 9000 Series Aggregation Services Router System Monitoring Command Reference*』の「*Alarm Management and Logging Correlation Commands*」の章を参照してください。

第 2 章

アラームおよびアラーム ログ相関の実装とモニタリング

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 ログおよび相関イベント レコード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、ネットワーク上にロギング相関を実装し、アラームをモニタリングするために必要な、新規および改訂されたタスクについて説明します。

（注） Cisco IOS XR ソフトウェアのシステム ロギングの詳細、およびこのモジュールにリストされているアラーム管理コマンド、ロギング相関コマンドの詳細については、このモジュールの関連資料，（53 ページ）のセクションを参照してください。 設定タスクを実行する手順の中で出現する可能性のあるその他のコマンドについて記載されたマニュアルを特定するには、『*Cisco ASR 9000 Series Aggregation Services Router Commands Master List*』をオンラインで検索してください。

アラームおよびアラーム ログ相関の実装とモニタリングの機能履歴

リリース	変更内容
リリース 3.7.2	この機能が導入されました。
リリース 3.8.0	SNMP アラーム相関機能が追加されました。

アラームおよびアラームログ相関の実装とモニタリングの前提条件

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。

アラームおよびアラーム ログ相関の実装に関する情報

アラームおよびアラーム相関を実装するには、次の概念を十分に理解している必要があります。

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

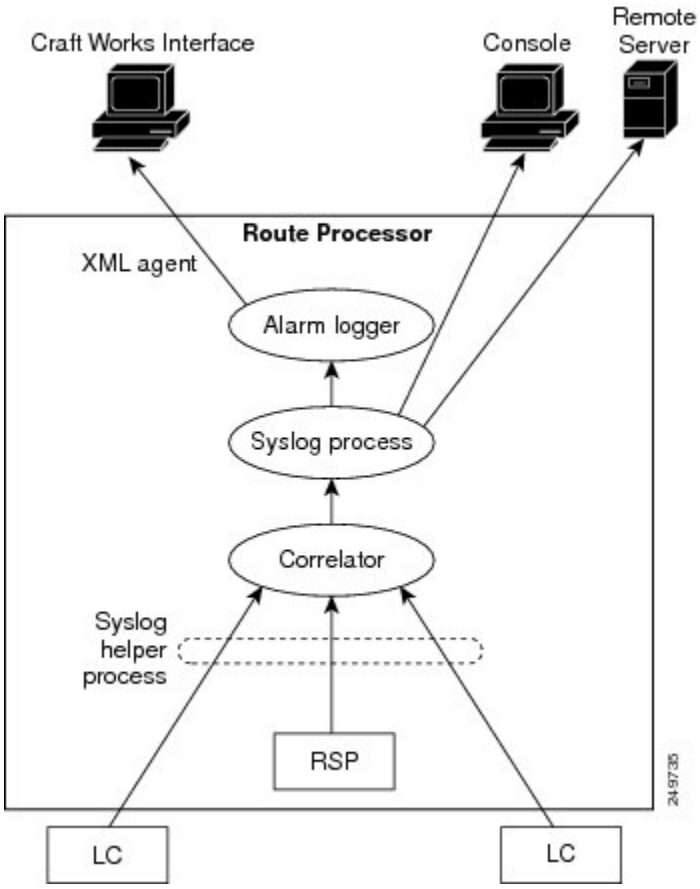
Cisco IOS XR ソフトウェアのアラーム ロギングおよびデバッグ イベント管理システム（ALDEMS）は、システム サーバおよびアプリケーションから転送されるアラーム メッセージをモニタリングし、格納するために使用されます。 また、ALDEMS は、単一の根本原因のために転送されたアラーム メッセージ同士を相関します。

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、高度に分散化されたシステムに必要とされるレベルの、アラーム管理および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, （5 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。

（注） ロギング相関の詳細については、ロギング相関, （6 ページ）を参照してください。

システム ロギング プロセス

ルータはデフォルトで、システム ロギング メッセージをシステム ロギング（syslog）プロセスに送信するように設定されています。 syslog メッセージは、システムのノード全体に分散されている syslog ヘルパー プロセスによって収集されます。 システム ロギング プロセスは、ネットワーク デバイス構成に応じて、システム ロギング バッファ、コンソール、端末回線、syslog サーバなどのさまざまな宛先へのロギング メッセージの配信を制御します。

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。バイステートアラームの詳細については、バイステート アラーム, （9 ページ）を参照してください。

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。

ロギング相関

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、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、不要なメッセージの蓄積が原因となって起こる ALDEMS の不要なストレスを防止します。 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。 ロギング相関ルール設定で適切なメッセージセットを設定して、（目的に応じて）狭いスコープまたは広いスコープでの相関を実現します。

相関のタイプ

根本原因メッセージを分離するためにルールに設定する相関には、2 つのタイプがあります。

非ステートフル相関：この相関は、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：この相関は、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。

ルールおよびルール セットの適用

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。

ルールセットの適用の場合も動作は同じですが、指定されたルール セットの一部であるすべてのルールの設定が適用されます。

show logging correlator rule コマンドを使用すると、**logging correlator apply ruleset** コマンドで設定した設定内容を含め、特定のルールの適用設定が表示されます。

ルート メッセージおよび相関メッセージ

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。 一致が検出されると、コリレータは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。 一致を検出するメッセージ検索は、タイマーの期限が切れるまで継続されます。 相関は、根本原因メッセージを受信した後に発生します。

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 ビット バイトが設定されたもの）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。 根本原因メッセージの転送方法および保存方法については、相関ルール, （7 ページ）を参照してください。

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 *2*：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	Emergencies
1	Alerts
2	Critical

重大度	システムの状態
3	Errors
4	Warnings
5	Notifications
6	Informational

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、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。

- アラーム ステート。SET ステートまたは CLEAR ステートのいずれかです。

アラームの容量しきい値設定

容量しきい値設定は、アラーム システムがしきい値超過アラームのレポートを開始するタイミングを定義します。 通常、警告アラームを生成する容量しきい値はバッファ容量の 80 パーセントに設定しますが、個別の設定では異なる設定が必要になる場合があります。

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。

Rule 1	Category	Group	Code
Root Cause 1	Cat 1	Group 1	Code 1
Non-root Cause 2	Cat 2	Group 2	Code 2
Rule 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 B 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。

非バイステート再発行フラグ

非バイステート再発行フラグは、非バイステート アラーム（イベント）の親バイステート根本原因アラームがクリアされた場合に、その非バイステートアラーム（イベント）がコリレータログから転送されるかどうかを制御します。 この状況では、アクティブなバイステート非根本原因は、条件がまだ存在するため、必ず転送されます。

非バイステート再発行フラグを使用すると、非バイステート アラームを転送するかどうかを制御できるようになります。

内部ルール

内部ルールは Cisco IOS XR ソフトウェアで定義され、Cisco IOS XR ソフトウェアのプロトコルおよびプロセスによって使用されます。 これらのルールはお客様が設定することはできませんが、

show logging correlator ruleコマンドを使用して表示できます。 すべての内部ルール名には、[INTERNAL] というプレフィックスが付きます。

SNMP アラーム相関

Cisco IOS XR マルチシャーシ システムなどの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。 そのため、この機能によりこれらの問題が解決されます。

SNMP アラーム相関機能の目的は、次のとおりです。

- 既存の syslog コリレータからの、相関機能の包括的な抽出
- その他のコンポーネントで機能を再使用に適した、DLL および API の作成
- SNMP トラップ相関を可能にするために、SNMP エージェントと DLL を統合

アラーム管理およびロギング相関の実装およびモニタ方法

ここでは、次のタスクについて説明します。

ロギング相関ルールの設定

このタスクでは、ロギング相関ルールの設定方法について説明します。

ロギング相関ルールを設定する目的は、ロギング相関用の根本原因アラーム メッセージおよび非根本原因アラームメッセージを（メッセージ カテゴリ、グループ、およびコードの組み合わせで）定義することです。 発信元の根本原因アラーム メッセージはsyslogプロセスに転送され、後続のすべての（非根本原因）アラーム メッセージはロギング相関バッファに送信されます。

相関ルールの設定用に使用できるメッセージ内のフィールドは、次のとおりです。

- メッセージカテゴリ（例：PKT_INFRA、MGBL、OS）
- メッセージ グループ（例：LINK、LINEPROTO、OIR）
- メッセージ コード（例：UPDOWN、GO_ACTIVE）。

アクティブなルートプロセッサ上で実行されているロギング相関ルールメカニズムにより、相関ルールの指定に一致するメッセージのキューイングが開始され、相関ルールのタイムアウト間隔で指定された時間だけ継続されます。

タイムアウト間隔は、コリレータが特定の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。

手順の概要

1. **configure**
2. **logging correlator rule** *correlation-rule* { **type** { **stateful** | **nonstateful** }}
3. **timeout** [*milliseconds*]
4. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**
5. **show logging correlator rule** {**all** | *correlation-rule1 ... correlation-rule14* } [**context** *context1 ... context 6*] [**location** *node-id1...node-id6*] [**rulesource** { **internal** | **user** }] [**ruletype** { **nonstateful** | **stateful** }] [**summary** | **detail**]

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**logging correlator rule** *correlation-rule* { **type** { **stateful** | **nonstateful** }} 例： `RP/0/RSP0/CPU0:router(config)# logging correlator rule rule_stateful`	ロギング相関ルールを設定します。 • 根本原因アラームがバイステートの場合、ステートフル相関が明確に変わる場合があります。 • 非ステート相関は変わりません。非根本原因アラームはすべて相関バッファに残ります。
ステップ 3	**timeout** [*milliseconds*] 例： `RP/0/RSP0/CPU0:router(config-corr-rule-st)# timeout 60000`	ロギング コリレータ ルール メッセージの収集期間を指定します。 • タイムアウトは、相関ルールで特定されたアラーム メッセージが最初にログに記録されたときに開始されます。
ステップ 4	次のいずれかのコマンドを使用します。 • **end** • **commit**	設定変更を保存します。 • **end** コマンドを実行すると、変更をコミットするように要求されます。 `Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`

	コマンドまたはアクション	目的

	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーションファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。

ステップ 5	**show logging correlator rule** {**all** | *correlation-rule1 ... correlation-rule14* } [**context** *context1 ... context 6*] [**location** *node-id1...node-id6*] [**rulesource** { **internal** | **user** }] [**ruletype** { **nonstateful** | **stateful** }] [**summary** | **detail**] 例： `RP/0/RSP0/CPU0:router# show logging correlator rule all`	（任意）定義済みの相関ルールを表示します。 • 出力には、メッセージ カテゴリ、グループ、およびコードの情報を含めて、各ルール名の設定が表示されます。

ロギング相関ルール セットの設定

このタスクでは、ロギング相関ルール セットの設定方法について説明します。

手順の概要

1. **configure**
2. **logging correlator ruleset** *ruleset*
3. **rulename** *rulename*
4. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**
5. **show logging correlator ruleset** { **all** | *correlation-ruleset1...correlation-ruleset14* } [**detail** | **summary**]

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**logging correlator ruleset** *ruleset* 例： `RP/0/RSP0/CPU0:router(config)# logging correlator ruleset ruleset1`	ロギング相関ルール セットを設定します。
ステップ 3	**rulename** *rulename* 例： `RP/0/RSP0/CPU0:router(config-corr-ruleset)# rulename stateful_rule`	ルール名を設定します。

ステップ 4	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。

	コマンドまたはアクション	目的
		◦ no と入力すると、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。 ◦ cancel と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。 • 設定変更を実行コンフィギュレーションファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。
ステップ 5	**show logging correlator ruleset** { **all** | *correlation-ruleset1...correlation-ruleset14* } [**detail** | **summary**] 例： `RP/0/RSP0/CPU0:router# show logging correlator ruleset all`	（任意）定義された相関ルール セットを表示します。

根本原因アラームおよび非根本原因アラームの設定

根本原因を 1 つ以上の非根本原因アラームに相関させ、それらをルールとして設定するには、相関ルールに対して指定されている **rootcause** および **nonrootcause** コマンドを使用します。

手順の概要

1. **configure**
2. **logging correlator rule** *correlation-rule* { **type** { **stateful** | **nonstateful** }}
3. **rootcause** { *msg-category group-name msg-code* }
4. **nonrootcause**
5. **alarm** *msg-category group-name msg-code*
6. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**
7. **show logging correlator rule** { **all** | *correlation-rule1...correlation-rule14* } [**context** *context1...context 6*] [**location** *node-id1...node-id6*] [**rulesource** { **internal** | **user** }] [**ruletype** { **nonstateful** | **stateful** }] [**summary** | **detail**]

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**logging correlator rule** *correlation-rule* { **type** { **stateful** | **nonstateful** }} 例： `RP/0/RSP0/CPU0:router(config)# logging correlator rule rule_stateful`	ロギング相関ルールを設定し、ステートフルまたは非ステートフル ルール タイプのサブモードを開始します。 • 根本原因アラームがバイステートの場合、ステートフル相関が明確に変わる場合があります。 • 非ステート相関は変わりません。非根本原因アラームはすべて相関バッファに残ります。
ステップ 3	**rootcause** { *msg-category group-name msg-code* } 例： `RP/0/RSP0/CPU0:router(config-corr-rule-st)# rootcause CAT_BI_1 GROUP_BI_1 CODE_BI_1`	根本原因アラーム メッセージを設定します。 • この例では、ステートフル コンフィギュレーション モードで根本原因アラームを指定しています
ステップ 4	**nonrootcause** 例： `RP/0/RSP0/CPU0:router(config-corr-rule-st)# nonrootcause`	非根本原因コンフィギュレーション モードを開始します
ステップ 5	**alarm** *msg-category group-name msg-code* 例： `RP/0/RSP0/CPU0:router(config-corr-rule-st-nonrc)# alarm CAT_BI_2 GROUP_BI_2 CODE_BI_2`	非根本原因アラーム メッセージを指定します。 • このコマンドは、次のように **nonrootcause** コマンドと一緒に発行できます **nonrootcause alarm** *msg-category group-name msg-code*
ステップ 6	次のいずれかのコマンドを使用します。 • **end** • **commit**	設定変更を保存します。 • **end** コマンドを実行すると、変更をコミットするように要求されます。 `Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`

	コマンドまたはアクション	目的

	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

ステップ 7	**show logging correlator rule** { **all** | *correlation-rule1...correlation-rule14* } [**context** *context1...context 6*] [**location** *node-id1...node-id6*] [**rulesource** { **internal** | **user** }] [**ruletype** { **nonstateful** | **stateful** }] [**summary** | **detail**] 例： `RP/0/RSP0/CPU0:router# show logging correlator rule all`	（任意）定義されているコリレータ ルールを表示します。

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。 このタスクでは、これらのフラグの実装方法について説明します。

これらのフラグに関する詳細については、再配置フラグ，（11 ページ）および非バイステート再発行フラグ，（11 ページ）を参照してください。

手順の概要

1. **configure**
2. **logging correlator rule** *correlation-rule* { **type** { **stateful** | **nonstateful** }}
3. **reissue-nonbistate**
4. **reparent**
5. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**
6. **show logging correlator rule** { **all** | *correlation-rule1...correlation-rule14* } [**context** *context1...context 6*] [**location** *node-id1...node-id6*] [**rulesource** { **internal** | **user** }] [**ruletype** { **nonstateful** | **stateful** }] [**summary** | **detail**]

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**logging correlator rule** *correlation-rule* { **type** { **stateful**	**nonstateful** }} 例： `RP/0/RSP0/CPU0:router(config)# logging correlator rule rule_stateful type nonstateful`	ロギング相関ルールを設定します。 • 根本原因アラームがバイステートの場合、ステートフル相関が明確に変わる場合があります。 • 非ステートフル相関は変わりません。 非根本原因アラームはすべて相関バッファに残ります。
ステップ 3	**reissue-nonbistate** 例： `RP/0/RSP0/CPU0:router(config-corr-rule-st)# reissue-nonbistate`	根本原因アラームがクリアされた後で、コリレータ ログから非バイステート アラーム メッセージ（イベント）を発行します。
ステップ 4	**reparent** 例： `RP/0/RSP0/CPU0:router(config-corr-rule-st)# reparent`	親となる根本原因がクリアされた後の、非根本原因アラームの動作を指定します。
ステップ 5	次のいずれかのコマンドを使用します。	設定変更を保存します。

	コマンドまたはアクション	目的

	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

ステップ 6	**show logging correlator rule** { **all** | *correlation-rule1...correlation-rule14* } [**context** *context1...context 6*] [**location** *node-id1...node-id6*] [**rulesource** { **internal** | **user** }] [**ruletype** { **nonstateful** | **stateful** }] [**summary** | **detail**] 例： `RP/0/RSP0/CPU0:router# show logging correlator rule all`	（任意）定義されているコリレータ ルールを表示します。

次の作業

定義されている相関ルールおよびルール セットをアクティブにするには、**logging correlator apply rule** コマンドおよび **logging correlator apply ruleset** コマンドを使用して、ルールおよびルール セットを適用する必要があります。

ロギング相関ルールの適用

このタスクでは、ロギング相関ルールの適用方法について説明します。

相関ルールを適用すると、そのルールがアクティブになり、スコープが与えられます。 1 つの相関ルールを、ルータ上の複数のスコープに適用できます。つまり、1 つのルールをルータ全体、または複数のロケーション、または複数のコンテキストに適用することが可能です。

（注） ルールが適用される場合、またはこのルールを含むルール セットが適用される場合は、そのルールまたはルール セットの適用が解除されるまで、設定によってルール定義を変更できません。

（注） 適用設定は、ルールおよびそのルールを含むルール セットの両方について、同時に設定できます。 この場合、ルールに対する適用設定は、すべての適用設定が結合されたものになります。

手順の概要

1. **configure**
2. **logging correlator apply rule** *correlation-rule*
3. 次のいずれかを実行します。
   - **all-of-router**
   - **location** *node-id*
   - **context** *name*
4. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**
5. **show logging correlator rule** { **all** | *correlation-rule1...correlation-rule14* } [**context** *context1...context 6*] [**location** *node-id1...node-id6*] [**rulesource** { **internal** | **user** }] [**ruletype** { **nonstateful** | **stateful** }] [**summary** | **detail**]

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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。

	コマンドまたはアクション	目的
ステップ 2	**logging correlator apply rule** *correlation-rule* 例： `RP/0/RSP0/CPU0:router(config)# logging correlator apply-rule rule1`	相関ルールを適用およびアクティブにし、相関適用ルールのコンフィギュレーション モードを開始します。

ステップ 3	次のいずれかを実行します。
	• **all-of-router**
	• **location** *node-id*
	• **context** *name*
	
	例：
	`RP/0/RSP0/CPU0:router(config-corr-apply-rule)# all-of-router`
	または
	`RP/0/RSP0/CPU0:router(config-corr-apply-rule)# location 0/2/CPU0`
	または
	`RP/0/RSP0/CPU0:router(config-corr-apply-rule)# logging correlator apply-rule rule2 context POS_0_0_0_0`
		• ロギング相関ルールをルータ上のすべてのノードに適用します。
		• ロギング相関ルールをルータ上の特定のノードに適用します。
		◦ ノードの場所は、*rack/slot/module* の形式で指定されます。
		• ロギング相関ルールを特定のコンテキストに適用します。

ステップ 4	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。

	コマンドまたはアクション	目的
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。
ステップ 5	**show logging correlator rule** { **all** | *correlation-rule1...correlation-rule14* } [**context** *context1...context 6*] [**location** *node-id1...node-id6*] [**rulesource** { **internal** | **user** }] [**ruletype** { **nonstateful** | **stateful** }] [**summary** | **detail**] 例： `RP/0/RSP0/CPU0:router# show logging correlator rule all`	（任意）定義されているコリレータ ルールを表示します。

ロギング相関ルール セットの適用

このタスクでは、ロギング相関ルール セットの適用方法について説明します。

相関ルール セットを適用すると、そのルールがアクティブになり、スコープが与えられます。 1 つのルールセット設定が適用されると、設定はただちにそのルールセットの一部であるルールに影響します。

（注） 以前に（単独または別のルール セットの一部として）適用したルールの定義は、そのルールまたはルール セットの適用を解除するまで、変更できません。 コマンドの **no** 形式を使用してルールの使用を無効にしてから、ルール セットの再適用を試行してください。

手順の概要

1. **configure**
2. **logging correlator apply ruleset** *correlation-rule*
3. 次のいずれかを実行します。
   - **all-of-router**
   - **location** *node-id*
   - **context** *name*
4. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**
5. **show logging correlator ruleset** { **all** | *correlation-ruleset1 ... correlation-ruleset14* } [**detail** | **summary**]

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**logging correlator apply ruleset** *correlation-rule* 例： `RP/0/RSP0/CPU0:router(config)# logging correlator apply ruleset ruleset2`	ルールセットを適用およびアクティブにし、相関適用ルールセットのコンフィギュレーション モードを開始します。
ステップ 3	次のいずれかを実行します。 • **all-of-router** • **location** *node-id* • **context** *name* 例： `RP/0/RSP0/CPU0:router(config-corr-ruleset)# all-of-router`	• ロギング相関ルールセットをルータ上のすべてのノードに適用します。 • ロギング相関ルール セットをルータ上の特定のノードに適用します。 ◦ ノードの場所は、*rack/slot/module* の形式で指定されます。 • ロギング相関ルール セットを特定のコンテキストに適用します。

	コマンドまたはアクション	目的

	または
	
	`RP/0/RSP0/CPU0:router(config-corr-ruleset)# location 0/2/CPU0`
	または
	
	`RP/0/RSP0/CPU0:router(config-corr-ruleset)# context`	

ステップ 4	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

ステップ 5	**show logging correlator ruleset** { **all** | *correlation-ruleset1 ... correlation-ruleset14* } [**detail** | **summary**] 例： `RP/0/RSP0/CPU0:router# show logging correlator ruleset all`	（任意）定義されているコリレータ ルールを表示します。

ロギング イベント バッファ設定の変更

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。

このタスクには次の手順が含まれます。

- ロギング イベント バッファ サイズの変更
- アラーム生成しきい値の設定
- アラーム フィルタ（重大度）の設定

注意 アラーム レポート用の重大度を下げる、または容量警告アラーム生成のしきい値を下げるようなアラーム設定の変更は、システムのパフォーマンスを低下させるおそれがあります。

注意 ロギング イベント バッファ サイズを変更すると、SET ステートのバイステート アラームを除く、すべてのイベント レコードのバッファがクリアされます。

手順の概要

1. **show logging events info**
2. **configure**
3. **logging events buffer-size** *bytes*
4. **logging events threshold** *percent*
5. **logging events level** *severity*
6. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**
7. **show logging events info**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**show logging events info** 例： `RP/0/RSP0/CPU0:router# show logging events info`	（任意）ロギング イベント バッファのサイズ（バイト単位）、アラーム イベント レコードに占められているバッファのパーセンテージ、アラームをレポートする容量しきい値、バッファ内のレコードの総数、および重大度フィルタ（設定されている場合）を表示します。
ステップ 2	**configure** 例： `RP/0/RSP0/CPU0:router# 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 3	**logging events buffer-size** *bytes* 例： `RP/0/RSP0/CPU0:router(config)# logging events buffer-size 50000`	アラーム レコード バッファのサイズを指定します。 • この例では、バッファ サイズは50000バイトに設定されています。
ステップ 4	**logging events threshold** *percent* 例： `RP/0/RSP0/CPU0:router(config)# logging events threshold 85`	到達するとアラーム ロガーによりしきい値超過アラームが生成される、ロギング イベント バッファのパーセンテージを指定します。 • この例では、イベント バッファが容量の 85 パーセントに到達すると、アラーム ロガーによりしきい値超過通知が生成されます。
ステップ 5	**logging events level** *severity* 例： `RP/0/RSP0/CPU0:router(config)# logging events level warnings`	表示されるロギング イベントを決定する重大度を設定します。（重大度のリストについては、アラーム重大度とフィルタリング，（8 ページ）の表 2：イベント ロギングのアラーム重大度，（8 ページ）を参照してください）。 • キーワード オプションは、**emergencies**、**alerts**、**critical**、**errors**、**warnings**、**notifications**、および **informational** です。 • この例では、重大度が警告（レベル4）以上のメッセージがアラーム ログに書き込まれます。 重大度がこれよりも低いメッセージ（通知および情報メッセージ）は、記録されません。
ステップ 6	次のいずれかのコマンドを使用します。 • **end** • **commit**	設定変更を保存します。 • **end** コマンドを実行すると、変更をコミットするように要求されます。 `Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`

	コマンドまたはアクション	目的

	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。

ステップ 7	**show logging events info** 例： `RP/0/RSP0/CPU0:router# show logging events info`	（任意）ロギング イベント バッファのサイズ（バイト単位）、アラーム イベント レコードに占められているバッファのパーセンテージ、アラームをレポートする容量しきい値、バッファ内のレコードの総数、および重大度フィルタ（設定されている場合）を表示します。 • このコマンドは、すべての設定が変更され、システムがその変更を受け付けたことを確認するために使用します。

ロギング コリレータ バッファ設定の変更

このタスクでは、ロギング コリレータ バッファの変更方法について説明します。

ロギングコリレータ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。 相関 ID 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。

手順の概要

1. **configure**
2. **logging correlator buffer-size** *bytes*
3. **exit**
4. **show logging correlator info**
5. **clear logging correlator delete** *correlation-id*
6. **clear logging correlator delete all-in-buffer**
7. **show logging correlator buffer** { **all-in-buffer** [**ruletype** [**nonstateful** | **stateful**]] | [**rulesource** [**internal** | **user**]] | **rule-name** *correlation-rule1...correlation-rule14* | **correlationID** *correlation-id1..correlation-id14* }

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**logging correlator buffer-size** *bytes* 例： `RP/0/RSP0/CPU0:router(config)# logging correlator buffer-size 100000`	ロギング コリレータ バッファのサイズを指定します。 • この例では、ロギング コリレータ バッファのサイズは 100,000 バイトに設定されています。
ステップ 3	**exit** 例： `RP/0/RSP0/CPU0:router(config)# exit`	グローバル コンフィギュレーション モードを終了して、ルータを EXEC モードに戻します。
ステップ 4	**show logging correlator info** 例： `RP/0/RSP0/CPU0:router# show logging correlator info`	（任意）ロギング コリレータ バッファのサイズについての情報、および相関メッセージに占められたバッファのパーセンテージについての情報を表示します
ステップ 5	**clear logging correlator delete** *correlation-id* 例： `RP/0/RSP0/CPU0:router# clear logging correlator delete 48 49 50`	（任意）ロギング コリレータ バッファから特定の相関イベント レコード（1 つまたは複数）を削除します。 • 相関 ID の範囲を指定して削除することもできます（スペース区切りで最大 32 個の相関 ID を指定可能）。

	コマンドまたはアクション	目的
ステップ 6	**clear logging correlator delete all-in-buffer** 例： `RP/0/RSP0/CPU0:router# clear logging correlator delete all-in-buffer`	（任意）ロギング コリレータ バッファからすべての相関イベント メッセージをクリアします。
ステップ 7	**show logging correlator buffer** { **all-in-buffer** [**ruletype** [**nonstateful** | **stateful**]] | [**rulesource** [**internal** | **user**]] | **rule-name** *correlation-rule1...correlation-rule14* | **correlationID** *correlation-id1..correlation-id14* } 例： `RP/0/RSP0/CPU0:router# show logging correlator buffer all-in-buffer`	（任意）相関イベント レコードの内容を表示します。 • このステップは、相関イベント ログから特定の相関 ID が削除されたことを確認する場合に使用します。

重大度および重大度範囲によるアラームの表示

このタスクでは、アラームを重大度ごとおよび重大度範囲ごとに表示する方法について説明します。

アラームは、重大度または重大度の範囲に基づいて表示できます。 重大度および該当するシステム状態のリストについては、アラーム重大度とフィルタリング，（8 ページ）の表 2：イベント ロギングのアラーム重大度，（8 ページ）を参照してください。

（注） コマンドは任意の順序で入力できます。

手順の概要

1. **show logging events buffer severity-lo-limit** *severity*
2. **show logging events buffer severity-hi-limit** *severity*
3. **show logging events buffer severity-hi-limit** *severity* **severity-lo-limit** *severity*
4. **show logging events buffer severity-hi-limit** *severity* **severity-lo-limit** *severity* **timestamp-lo-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*]

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**show logging events buffer severity-lo-limit** *severity* 例： `RP/0/RSP0/CPU0:router# show logging events buffer severity-lo-limit notifications`	（任意）重大度が指定した数値以下のロギング イベントを表示します。 • この例では、重大度が通知（重大度 5）以下のアラームが表示されます。 情報（重大度 6）のメッセージは省略されます。 （注） 重大度に割り当てられた数値ではなく、重大度の *description* を指定するには、**severity-lo-limit** キーワードおよび *severity* 引数を使用します。
ステップ 2	**show logging events buffer severity-hi-limit** *severity* 例： `RP/0/RSP0/CPU0:router# show logging events buffer severity-hi-limit critical`	（任意）重大度が指定した数値以上のロギング イベントを表示します。 • この例では、重大度が重要（重大度 2）以上のアラームが表示されます。 重大度がアラート（重大度 1）および緊急（重大度 0）のものは省略されます。 （注） 重大度に割り当てられた数値ではなく、重大度の *description* を指定するには、**severity-hi-limit** キーワードおよび *severity* 引数を使用します。
ステップ 3	**show logging events buffer severity-hi-limit** *severity* **severity-lo-limit** *severity* 例： `RP/0/RSP0/CPU0:router# show logging events buffer severity-hi-limit alerts severity-lo-limit critical`	（任意）重大度範囲内のロギング イベントを表示します。 • この例では、重大度が重要（重大度2）およびアラート（重大度 1）のアラームが表示されます。 その他のイベント重大度はすべて省略されます。
ステップ 4	**show logging events buffer severity-hi-limit** *severity* **severity-lo-limit** *severity* **timestamp-lo-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*] 例： `RP/0/RSP0/CPU0:router# show logging events buffer severity-lo-limit warnings severity-hi-limit critical timestamp-lo-limit 22:00:00 may 07 04`	（任意）指定されたタイム スタンプ以降に発生した、重大度範囲内のロギング イベントが表示されます。 *month*、*day*、および *year* の各引数を指定していない場合は、デフォルトで現在の月、日、および年になります。 • この例では、2004 年 5 月 7 日 22:00:00 以降に発生した重大度が警告（重大度 4）、エラー（重大度 3）、および重要（重大度 2）のアラームが表示されます。 タイム スタンプよりも前に発生したその他のメッセージはすべて省略されます。

タイム スタンプ範囲に応じたアラームの表示

タイム スタンプ範囲に応じてアラームを表示できます。 特定の始点および終点を指定することは、特定の既知のシステム イベントの間に発生したアラームの分離に役立ちます。

このタスクでは、タイム スタンプ範囲に応じて、アラームを表示する方法について説明します。

（注） コマンドは任意の順序で入力できます。

手順の概要

1. **show logging events buffer timestamp-lo-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*]
2. **show logging events buffer timestamp-hi-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*]
3. **show logging events buffer timestamp-hi-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*] **timestamp-lo-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*]

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**show logging events buffer timestamp-lo-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*] 例： `RP/0/RSP0/CPU0:router# show logging events buffer timestamp-lo-limit 21:28:00 april 18 04`	（任意）指定した日時以降のタイム スタンプ付きのロギング イベントを表示します。 • *month*、*day*、および *year* の各引数を指定していない場合は、デフォルトで現在の月、日、および年になります。 • この出力例では、2004 年 4 月 18 日 21:28:00 以降にロギングされたイベントが表示されます。
ステップ 2	**show logging events buffer timestamp-hi-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*] 例： `RP/0/RSP0/CPU0:router# show logging events buffer timestamp-hi-limit 21:28:03 april 18 04`	（任意）指定した日時よりも前のタイム スタンプ付きのロギング イベントを表示します。 • *month*、*day*、および *year* の各引数を指定していない場合は、デフォルトで現在の月、日、および年になります。 • この出力例では、2004 年 4 月 18 日 21:28:03 よりも前にロギングされたイベントが表示されます。
ステップ 3	**show logging events buffer timestamp-hi-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*] **timestamp-lo-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*]	（任意）指定した日時の範囲のタイム スタンプ付きのロギング イベントを表示します。

	コマンドまたはアクション	目的
	例： `RP/0/RSP0/CPU0:router# show logging events buffer timestamp-hi-limit 21:28:00 april 18 04 timestamp-lo-limit 21:16:00 april 18 03`	• *month*、*day*、および *year* の各引数を指定していない場合は、デフォルトで現在の月、日、および年になります。 • この出力例では、2003 年 4 月 18 日 21:16:00 から 2004 年 4 月 18 日 21:28:00 の間にロギングされたイベントが表示されます。

メッセージ グループおよびメッセージ コードに応じたアラームの表示

このタスクでは、ロギング イベント バッファ内のアラームを、メッセージ コードおよびメッセージ グループに応じて表示する方法について説明します。

メッセージ グループおよびメッセージ コードごとにアラームを表示することは、関連するイベントの分離に役立ちます。

（注） コマンドは任意の順序で入力できます。

手順の概要

1. **show logging events buffer group** *message-group*
2. **show logging events buffer message** *message-code*
3. **show logging events buffer group** *message-group* **message** *message-code*

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**show logging events buffer group** *message-group* 例： `RP/0/RSP0/CPU0:router# show logging events buffer group SONET`	（任意）指定したメッセージ グループに一致するロギング イベントを表示します。 • この例では、メッセージ グループ SONET を含むすべてのイベントが表示されます。

	コマンドまたはアクション	目的
ステップ 2	**show logging events buffer message** *message-code* 例： `RP/0/RSP0/CPU0:router# show logging events buffer message ALARM`	（任意）指定したメッセージ コードに一致するロギング イベントを表示します。 • この例では、メッセージ コード ALARM を含むすべてのイベントが表示されます。
ステップ 3	**show logging events buffer group** *message-group* **message** *message-code* 例： `RP/0/RSP0/CPU0:router# show logging events buffer group SONET message ALARM`	（任意）指定したメッセージ グループおよびメッセージ コードに一致するロギング イベントを表示します。 • この例では、メッセージ グループ SONET およびメッセージ コード ALARM を含むすべてのイベントが表示されます。

最初と最後の範囲に応じたアラームの表示

このタスクでは、ロギングイベントバッファ内の最初と最後のアラームの範囲に応じてアラームを表示する方法について説明します。

ロギングイベントバッファ内の最初または最後を始点とした任意の範囲のアラームを表示できます。

（注） コマンドは任意の順序で入力できます。

手順の概要

1. **show logging events buffer first** *event-count*
2. **show logging events buffer last** *event-count*
3. **show logging events buffer first** *event-count* **last** *event-count*

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**show logging events buffer first** *event-count* 例： `RP/0/RSP0/CPU0:router# show logging events buffer first 15`	（任意）ロギング イベント バッファ内の最初のイベントから順にロギング イベントを表示します。 • *event-count* 引数には、表示するイベントの数を指定します。 • この例では、ロギング イベント バッファの最初の 15 個のイベントが表示されます。
ステップ 2	**show logging events buffer last** *event-count* 例： `RP/0/RSP0/CPU0:router# show logging events buffer last 20`	（任意）ロギング イベント バッファ内の最後のイベントから順にロギング イベントを表示します。 • *event-count* 引数には、表示するイベントの数を指定します。 • この例では、ロギング イベント バッファの最後の 20 個のイベントが表示されます。
ステップ 3	**show logging events buffer first** *event-count* **last** *event-count* 例： `RP/0/RSP0/CPU0:router# show logging events buffer first 20 last 20`	（任意）ロギング イベント バッファの最初と最後のイベントを表示します。 • *event-count* 引数には、表示するイベントの数を指定します。 • この例では、ロギング イベント バッファの最初の 20 個のイベントと最後の 20 個のイベントの両方が表示されます。

ロケーションによるアラームの表示

このタスクでは、ロケーションごとにアラームを表示する方法について説明します。

（注） コマンドは任意の順序で入力できます。

手順の概要

1. **show logging events buffer location** *node-id*
2. **show logging events buffer location** *node-id* **event-hi-limit** *event-id* **event-lo-limit** *event-id*

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**show logging events buffer location** *node-id* 例： `RP/0/RSP0/CPU0:router# show logging events buffer 0/2/CPU0`	（任意）発生イベント ID の範囲を特定のノードに限定します。 • ノードの場所は、*rack/slot/module* の形式で指定されます。
ステップ 2	**show logging events buffer location** *node-id* **event-hi-limit** *event-id* **event-lo-limit** *event-id* 例： `RP/0/RSP0/CPU0:router# show logging events buffer location 0/2/CPU0 event-hi-limit 100 event-lo-limit 1`	（任意）発生イベント ID の範囲を特定のノードに限定し、表示するイベント ID の上限および下限を指定することで、範囲を絞り込みます。 • ノードの場所は、*rack/slot/module* の形式で指定されます。

イベント レコード ID によるアラームの表示

このタスクでは、イベント レコード ID ごとにアラームを表示する方法について説明します。

（注） コマンドは任意の順序で入力できます。

手順の概要

1. **show logging events buffer all-in-buffer**
2. **show logging events buffer event-hi-limit** *event-id* **event-lo-limit** *event-id*

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**show logging events buffer all-in-buffer** 例： `RP/0/RSP0/CPU0:router# show logging events buffer all-in-buffer`	（任意）ロギング イベント バッファ内のすべてのメッセージを表示します。 **注意** アラームの重大度の設定によっては、このコマンドで大量の出力が作成される場合があります。

	コマンドまたはアクション	目的
ステップ 2	**show logging events buffer event-hi-limit** *event-id* **event-lo-limit** *event-id* 例： `RP/0/RSP0/CPU0:router# show logging events buffer event-hi-limit 100 event-lo-limit 1`	（任意）表示するイベントIDの上限および下限を指定することで、範囲を絞り込みます。

ロギング相関バッファのサイズ、メッセージ、ルールの表示

このタスクでは、ロギング相関バッファのサイズ、ロギング相関バッファ内のメッセージ、および相関ルールの表示方法について説明します。

（注） コマンドは任意の順序で入力できます。

手順の概要

1. **show logging correlator info**
2. **show logging correlator buffer all-in-buffer**
3. **show logging correlator buffer correlationID** *correlation-id*
4. **show logging correlator buffer rule-name** *correlation-rule*
5. **show logging correlator rule all**
6. **show logging correlator rule** *correlation-rule*
7. **show logging correlator ruleset all**
8. **show logging correlator ruleset** *ruleset-name*

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**show logging correlator info** 例： `RP/0/RSP0/CPU0:router# show logging correlator info`	（任意）ロギング相関バッファのサイズ（バイト単位）および相関メッセージが占めているパーセンテージを表示します。

	コマンドまたはアクション	目的
ステップ 2	**show logging correlator buffer all-in-buffer** 例： `RP/0/RSP0/CPU0:router# show logging correlator buffer all-in-buffer`	（任意）ロギング相関バッファ内のすべてのメッセージを表示します。
ステップ 3	**show logging correlator buffer correlationID** *correlation-id* 例： `RP/0/RSP0/CPU0:router# show logging correlator buffer correlationID 37`	（任意）相関バッファ内の特定の相関 ID に一致する特定のメッセージを表示します。
ステップ 4	**show logging correlator buffer rule-name** *correlation-rule* 例： `RP/0/RSP0/CPU0:router# show logging correlator buffer rule-name rule7`	（任意）相関バッファ内の特定のルールに一致する特定のメッセージを表示します。
ステップ 5	**show logging correlator rule all** 例： `RP/0/RSP0/CPU0:router# show logging correlator rule all`	（任意）定義された相関ルールをすべて表示します。
ステップ 6	**show logging correlator rule** *correlation-rule* 例： `RP/0/RSP0/CPU0:router# show logging correlator rule rule7`	（任意）指定された相関ルールを表示します。
ステップ 7	**show logging correlator ruleset all** 例： `RP/0/RSP0/CPU0:router# show logging correlator ruleset all`	（任意）定義された相関ルール セットをすべて表示します。
ステップ 8	**show logging correlator ruleset** *ruleset-name* 例： `RP/0/RSP0/CPU0:router# show logging correlator ruleset ruleset_static`	（任意）指定された相関ルール セットを表示します。

アラーム イベント レコードのクリアおよびバイステート アラームのリセット

このタスクでは、アラーム イベント レコードおよびバイステート アラームのクリア方法について説明します。

不要なメッセージおよび古くなったメッセージをクリアして、イベントロギングバッファのサイズを小さくすることで検索を容易にし、より移動しやすくします。

ロギングイベントバッファ内のイベントのクリアに使用できるフィルタリング機能（**clear logging events delete** コマンドを使用）は、ロギング イベント バッファ内のイベントの表示にも使用できます（**show logging events buffer** コマンドを使用）。

（注） コマンドは任意の順序で入力できます。

手順の概要

1. **show logging events buffer all-in-buffer**
2. **clear logging events delete timestamp-lo-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*]
3. **clear logging events delete event-hi-limit** *severity* **event-lo-limit** *severity*
4. **clear logging events delete location** *node-id*
5. **clear logging events delete first** *event-count*
6. **clear logging events delete last** *event-count*
7. **clear logging events delete message** *message-code*
8. **clear logging events delete group** *message-group*
9. **clear logging events reset all-in-buffer**
10. **show logging events buffer all-in-buffer**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**show logging events buffer all-in-buffer** 例： `RP/0/RSP0/CPU0:router# show logging events buffer all-in-buffer`	（任意）ロギングイベントバッファ内のすべてのメッセージを表示します。
ステップ 2	**clear logging events delete timestamp-lo-limit** *hh* **:** *mm* **:** *ss* [*month*] [*day*] [*year*] 例： `RP/0/RSP0/CPU0:router# clear logging`	（任意）ロギング イベント バッファから、指定した日時よりも前に発生したロギング イベントを削除します。

	コマンドまたはアクション	目的
	`events delete timestamp-lo-limit 20:00:00 april 01 2004`	• *month*、*day*、および *year* の各引数を指定していない場合は、デフォルトで現在の月、日、および年になります。 • この例では、2004 年 4 月 1 日よりも前に発生したすべてのイベントが削除されます。
ステップ 3	**clear logging events delete event-hi-limit** *severity* **event-lo-limit** *severity* 例： `RP/0/RSP0/CPU0:router# clear logging events delete event-hi-limit warnings event-lo-limit informational`	（任意）アラーム メッセージのロギング用の重大度範囲内のロギング イベントを削除します。 • この例では、重大度が警告、通知、情報のイベントがすべて削除されます。
ステップ 4	**clear logging events delete location** *node-id* 例： `RP/0/RSP0/CPU0:router# clear logging events delete location 0/2/CPU0`	（任意）ロギング イベント バッファから、特定のノード上で発生したロギング イベントを削除します。 • ノードの場所は、*rack/slot/module* の形式で指定されます。
ステップ 5	**clear logging events delete first** *event-count* 例： `RP/0/RSP0/CPU0:router# clear logging events delete first 10`	（任意）ロギング イベント バッファ内の最初のイベントから順にロギング イベントを削除します。 • この例では、ロギング イベント バッファの最初の 10 個のイベントがクリアされます。
ステップ 6	**clear logging events delete last** *event-count* 例： `RP/0/RSP0/CPU0:router# clear logging events delete last 20`	（任意）ロギング イベント バッファ内の最後のイベントから順にロギング イベントを削除します。 • この例では、ロギング イベント バッファの最後の 20 個のイベントがクリアされます。
ステップ 7	**clear logging events delete message** *message-code* 例： `RP/0/RSP0/CPU0:router# clear logging events delete message sys`	（任意）指定したメッセージ コードを含むロギング イベントを削除します。 • この例では、メッセージ コード SYS が含まれるすべてのイベントがロギング イベント バッファから削除されます。
ステップ 8	**clear logging events delete group** *message-group* 例： `RP/0/RSP0/CPU0:router# clear logging events delete group config_i`	（任意）指定したメッセージ グループを含むロギング イベントを削除します。 • この例では、メッセージ グループ CONFIG_I が含まれるすべてのイベントがロギング イベント バッファから削除されます。

	コマンドまたはアクション	目的
ステップ 9	**clear logging events reset all-in-buffer** 例： `RP/0/RSP0/CPU0:router# clear logging events reset all-in-buffer`	（任意）ロギング イベント バッファから、SET ステートのバイステート アラームがすべて削除されます。
ステップ 10	**show logging events buffer all-in-buffer** 例： `RP/0/RSP0/CPU0:router# show logging events buffer all-in-buffer`	（任意）ロギングイベントバッファ内のすべてのメッセージを表示します。

SNMP 相関バッファ サイズの定義

このタスクでは、SNMP トラップの相関バッファ サイズを定義する方法について説明します。

手順の概要

1. **configure**
2. **snmp-server correlator buffer-size** *bytes*
3. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**snmp-server correlator buffer-size** *bytes* 例： `RP/0/RSP0/CPU0:router(config)# snmp-server correlator buffer-size 600`	SNMP 相関トラップを保存できるバッファ サイズを定義します。 デフォルトのサイズは 64 KB です。 相関バッファは手動でクリアできます。手動でクリアしない場合はバッファは自動的に一巡し、新しい相関を収容できるように一番古い相関から破棄されます。

	コマンドまたはアクション	目的

ステップ 3	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

SNMP ルールセットの定義

このタスクでは、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定義します。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。

手順の概要

1. **configure**
2. **snmp-server correlator ruleset** *name* **rulename** *name*
3. 次のいずれかのコマンドを使用します。
   • **end**
   • **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**snmp-server correlator ruleset** *name* **rulename** *name*
	
	例：
	`RP/0/RSP0/CPU0:router(config)#`
	
	`snmp-server correlator ruleset rule1`
	` rulename rule2`
	` host ipv4 address 1.2.3.4`
	` host ipv4 address 2.3.4.5 port 182`
		複数のルールを 1 つのグループにグループ化し、そのグループをホストのセットに適用できるルールセットを指定します。

ステップ 3	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。
		
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。

SNMP 相関ルールの設定

このタスクでは、SNMP 相関ルールの設定方法について説明します。

SNMP トラップ相関ルールを設定する目的は、相関ルールまたは非相関ルールを定義して、特定のトラップ宛先に適用することです。

手順の概要

1. **configure**
2. **snmp-server correlator rule** *rule_name* { **nonrootcause trap** *trap_oid* **varbind** *vbind_OID* { **index** | **value** } **regex** *line* | **rootcause trap** *trap_oid* **varbind** *vbind_OID* { **index** | **value** } **regex** *line* | **timeout** }
3. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**snmp-server correlator rule** *rule_name* { **nonrootcause trap** *trap_oid* **varbind** *vbind_OID* { **index** | **value** } **regex** *line* | **rootcause trap** *trap_oid* **varbind** *vbind_OID* { **index** | **value** } **regex** *line* | **timeout** }
	
	例：
	`RP/0/RSP0/CPU0:router(config)# snmp-server correlator rule test`
	`  rootcause A`
	`  varbind A1 value regex RA1`
	`  varbind A2 index regex RA2`
	`  timeout 5000`
	`  nonrootcause`
	`  trap B`
	`  varbind B1 index regex RB1`
	`  varbind B2 value regex RB2`
	`  trap C`
	`  varbind C1 value regex RC1`
	`  varbind C2 value regex RC2`
		SNMP 相関ルールを設定します。 根本原因トラップ OID 数値または非根本原因トラップ一致定義を指定できます。
		• 非根本原因トラップ OID 数値および（任意で）非根本原因トラップに固有の 1 つ以上の変数バインド数値を指定します。この非根本原因トラップは、このルールに対して有効な非根本原因を検出するためにすべて照会される必要があります。 POSIX regexp は、vbind インデックスまたは値が一致する必要のある正規表現を指定します。
		• 根本原因トラップ OID 数値および（任意で）根本原因トラップに固有の 1 つ以上の変数バインド数値を指定します。この根本原因トラップは、このルールに対して有効な根本原因を検出するためにすべて照会される必要があります。 POSIX regexp は、vbind インデックスまたは値が一致する必要のある正規表現を指定します。

	コマンドまたはアクション	目的
		（注） この指定されたルールで根本原因または非根本原因を最初に受信した後に相関を検出するタイムアウトを指定できます。 範囲は 1 ～ 600000 ミリ秒です。 （注） トラップのすべてのOID値および変数バインドは検証され、IOS XR でサポートされている有効なOIDと一致しない場合は拒否されます。

ステップ 3	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

SNMP 相関ルールの適用

このタスクの目的は、SNMP トラップ相関ルールを特定のトラップ宛先に適用することです。

手順の概要

1. **configure**
2. **snmp-server correlator apply rule** *rule-name* [**all-hosts** | **host ipv4 address** *address* [*port*]
3. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**snmp-server correlator apply rule** *rule-name* [**all-hosts** | **host ipv4 address** *address* [*port*]
	
	例：
	`RP/0/RSP0/CPU0:router# snmp-server correlator apply rule ifupdown`
	`host ipv4 address 1.2.3.4`
	`host ipv4 address 2.3.4.5 port 182`
		SNMP トラップ相関ルールを特定のトラップ宛先に適用します。宛先がすべてのトラップ ホストであるトラップにルールを適用するか、または個別の IP アドレスおよびオプションのポートを指定することで特定のサブセットにルールを適用できます。

ステップ 3	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。

	コマンドまたはアクション	目的
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

SNMP 相関ルールセットの適用

このタスクの目的は、複数の SNMP トラップ相関ルールをグループとして特定のトラップ宛先に適用することです。

手順の概要

1. **configure**
2. **snmp-server correlator apply ruleset** *ruleset-name* [**all-hosts** | **host ipv4 address** *address* [*port*]
3. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**snmp-server correlator apply ruleset** *ruleset-name* [**all-hosts**	**host ipv4 address** *address* [*port*] 例： `RP/0/RSP0/CPU0:router# snmp-server correlator apply ruleset ruleset_1 host ipv4 address 1.2.3.4 host ipv4 address 2.3.4.5 port 182`	SNMP トラップ相関ルールセットを特定のトラップ宛先に適用します。宛先がすべてのトラップ ホストであるトラップに複数の SNMP トラップ相関ルール セットを適用するか、または個別の IP アドレスおよびオプションのポートを指定することで特定のサブセットに複数の SNMP トラップ相関ルール セットを適用できます。
ステップ 3	次のいずれかのコマンドを使用します。	設定変更を保存します。

	コマンドまたはアクション	目的

	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

非同期 Syslog 通信

非同期Syslog通信機能では、増分レートでログメッセージを送信することにより、各ノード（LC、RP）でテストするメッセージを適切に順序付け、各ノード（LC、RP）上の複数のクライアントから生成されるメッセージをドロップせず、パフォーマンス、スケーラビリティ、および遅延を確認することができます。

この機能により次のことが可能になります。

- MC min 4+1 でテストするメッセージの正しい順序付け。
- MC min 4+1 の複数のクライアントから生成されるメッセージの非ドロップ。
- Syslogd_helper メッセージ処理キャパシティ。テスト クライアント（ロガー）を使用して、大量の Syslog メッセージでフラッディングし、Syslog メッセージが失われないかどうかを確認します（新規設計による指定のレート）。
- 各ノードからの 1200 または 1500 メッセージ/秒。LC および RC での restart restart/crash syslogd_helper、RP での correlatord および syslogd。

- ルーティング プロトコル OSPF の設定。 サブ インターフェイスを使用して 5k ネイバーを設定します。インターフェイス フラッピングを実行してログ メッセージを生成し、syslogd_helper パフォーマンスを確認します。
- 少数の重いプロセスのデバッグをイネーブルにします。sysdb/GSP

アラーム管理およびロギング相関の設定例

ここでは、次の設定例について説明します。

表示されるイベント数を削減するためのアラームフィルタリング用重大度の増加およびアラーム バッファ サイズと容量しきい値の変更：例

次の設定例は、容量しきい値を 90 パーセントに設定して、ロギング イベント バッファをデフォルトから 10,000 バイトに縮小し、重大度をエラーに増加する方法を示します。

```
!
logging events threshold 90
logging events buffer-size 10000
logging events level errors
!
```

重大度をエラーに増やすと、重大度がエラー以上のアラームのみが表示されるようになるため、ロギング イベント バッファに表示されるアラームの数が減ります。 容量しきい値を 90 パーセントに増やすと、しきい値超過イベントとラップアラウンド イベント間の間隔が短くなります。そのため、ロギング イベント バッファの容量が 90 パーセントに到達するまで、ロギング イベント バッファからしきい値超過アラームは生成されません。 ロギング イベント バッファのサイズを 10,000 バイトに縮小すると、ロギング イベント バッファに表示されるアラームの数が減り、コンポーネントのメモリ要件が小さくなります。

ノード ステータス メッセージを永続的に抑制するための非ステートフル相関ルールの設定：例

次に、ノード ステータス メッセージを永続的に抑制するように非ステートフル相関ルールを設定する方法の例を示します。

```
logging correlator rule node_status type nonstateful
timeout 4000
  rootcause PLATFORM INVMGR NODE_STATE_CHANGE
  nonrootcause
    alarm PLATFORM SYSLDR LC_ENABLED
```

```
    alarm PLATFORM ALPHA_DISPLAY CHANGE
  !
!
logging correlator apply rule node_status
  all-of-router
!
```

この例では、カードのブート後に syslog プロセスに同時に転送されるものとして 3 つの類似するメッセージが特定されます。

PLATFORM-INVMGR-6-NODE_STATE_CHANGE : Node: 0/1/CPU0, state: IOS XR RUN

PLATFORM-SYSLDR-5-LC_ENABLED : LC in slot 1 is now running IOX

PLATFORM-ALPHA_DISPLAY-6-CHANGE : Alpha display on node 0/1/CPU0 changed to IOX RUN in state default

これらのメッセージは類似しています。 ログに 1 つのメッセージのみが表示されるようにするには、これらのメッセージのいずれか1つを根本原因メッセージとして（ログに表示されるように）指定し、その他のメッセージは非根本原因メッセージと見なされるようにします。

通常、根本原因メッセージは一番最初に着信したメッセージですが、これは要件ではありません。

```
logging correlator rule node_status type nonstateful
  timeout 4000
  rootcause PLATFORM INVMGR NODE_STATE_CHANGE
  nonrootcause
    alarm PLATFORM SYSLDR LC_ENABLED
    alarm PLATFORM ALPHA_DISPLAY CHANGE
  !
!
```

この例では、PLATFORM INVMGR NODE_STATE_CHANGE アラーム（根本原因メッセージ）を PLATFORM SYSLDR LC_ENABLED アラームおよび PLATFORM ALPHA_DISPLAY CHANGE アラームと相関するように node_status という名前の相関ルールを設定します。 UPDOWN 相関ルールは、ルータ全体に適用されます。

```
logging correlator apply rule node_status
  all-of-router
!
```

カードのブート後に、次のメッセージが送信されます。

PLATFORM-INVMGR-6-NODE_STATE_CHANGE : Node: 0/1/CPU0, state: IOS XR RUN

PLATFORM-SYSLDR-5-LC_ENABLED : LC in slot 1 is now running IOX

PLATFORM-ALPHA_DISPLAY-6-CHANGE : Alpha display on node 0/1/CPU0 changed to IOX RUN in state default

コリレータは、PLATFORM-INVMGR-6-NODE_STATE_CHANGE メッセージを syslog プロセスに転送し、残りの 2 つのメッセージはロギング コリレータ バッファに保持されます。

この例では、**show logging events buffer all-in-buffer** コマンドの show 出力例には、node_status 相関ルールの 4 秒間の期限が切れた後、ロギング イベント バッファに保存されたアラームが表示されます。

```
RP/0/RSP0/CPU0:router# show logging events buffer all-in-buffer

#ID :C_id:Source :Time :%CATEGORY-GROUP-SEVERITY-MESSAGECODE: Text

#76 :12  :RP/0/0/CPU0:Aug  2 22:32:43 : invmgr[194]:


%PLATFORM-INVMGR-6-NODE_STATE_CHANGE : Node: 0/1/CPU0, state: IOS XR RUN
```

show logging correlator buffer 相関 ID コマンドは、1 分間の間隔が経過した後、次の出力を作成します。 出力には、ロギング相関バッファ内の相関 ID 12 が割り当てられたアラームが表示されます。

```
RP/0/RSP0/CPU0:router# show logging correlator buffer correlationID 46

#C_id.id:Rule Name:Source :Time : Text

#12.1 :nodestatus:RP/0/0/CPU0:Aug  2 22:32:43 : invmgr[194]:
%PLATFORM-INVMGR-6-NODE_STATE_CHANGE : Node: 0/1/CPU0, state: IOS XR RUN
#12.2 :nodestatus:RP/0/0/CPU0:Aug  2 22:32:43 : sysldr[336]: %PLATFORM-SYSLDR-5-LC_ENABLED
 : LC in slot 1 is now running IOX
#12.3 :nodestatus:RP/0/0/CPU0:Aug  2 22:32:44 : alphadisplay[102]:
%PLATFORM-ALPHA_DISPLAY-6-CHANGE : Alpha display on node 0/1/CPU0 changed to IOX RUN  in
state default
```

このルールは非ステートフルとして定義されたため、これらのメッセージは無期限でバッファ内に保持されます。

LINK UPDOWN アラームおよび SONET ALARM アラーム用ステートフル相関ルールの設定：例

次に、LINK UPDOWN メッセージおよび SONET ALARM メッセージ用の相関ルールを設定する方法の例を示します。

```
!
logging correlator rule updown type stateful
  timeout 10000
  rootcause PKT_INFRA LINK UPDOWN
  nonrootcause
    alarm L2 SONET ALARM
  !
!
logging correlator apply rule updown
  all-of-router
```

```
!
```

この例では、2 つのルータが接続されているとします。 コリレータが根本原因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 syslog プロセスに直接送信します。 ルールに一致する後続の PKT_INFRA-LINK- UPDOWN メッセージまたは L2-SONET-ALARM メッセージはリーフ メッセージと見なされ、ロギング コリレータ バッファに保存されます。 何らかの理由で、リーフ メッセージ（この例の L2-SONET-ALARM アラームなど）を最初に受信した場合、コリレータはそのメッセージをロギング イベント バッファへただちには送信せず、タイムアウト間隔が経過するまで待機します。 タイムアウト後、ルート メッセージが受信されなかった場合は、ロギング コリレータ バッファ内のタイムアウト間隔中に受信したすべてのメッセージが syslog プロセスに転送されます。

この例では、updown という名前の相関ルールが、PKT_INFRA-LINK-UPDOWN アラーム（ルート メッセージ）と L2-SONET-ALARM アラーム（PKT_INFRA-LINK-UPDOWN アラームに関連付けられたリーフ メッセージ）を相関するように設定されています。

```
logging correlator rule updown type stateful
  timeout 10000
  rootcause PKT_INFRA LINK UPDOWN
  nonrootcause
    alarm L2 SONET ALARM
In this example, the updown correlation rule is applied to the entire router:
logging correlator apply rule updown
  all-of-router
```

次に、**show logging events buffer all-in-buffer** コマンドの出力例を示します。 この出力例では、設定されている updown 相関ルールの 1 分間の期限が切れた後にロギング イベント バッファに保存されたアラームが表示されます。

```
RP/0/RSP0/CPU0:router# show logging events buffer all-in-buffer
#ID :C_id:Source :Time :%CATEGORY-GROUP-SEVERITY-MESSAGECODE: Text
#144 :46 :LC/0/7/CPU0:Jan 30 16:35:39 2004:ifmgr[130]: %PKT_INFRA-LINK-3-UPDOWN :
Interface POS0/7/0/0, changed state to Down
```

（注） タイムアウト間隔中の最初の LINK UPDOWN ルート メッセージのみが syslog プロセスに転送されます。

次に、1 分間の間隔が経過した後に作成された **show logging correlator buffer correlationID** コマンドの出力例を示します。 出力には、ロギング相関バッファ内の相関 ID 46 が割り当てられたアラームが表示されます。 この例では、タイムアウト間隔中に生成された、相関 ID 46 が割り当てられた PKT_INFRA-LINK-UPDOWN 根本原因メッセージと、L2-SONET-ALARM リーフ メッセージが表示されます。

```
RP/0/RSP0/CPU0:router# show logging correlator buffer correlationID 46
```

```
#C_id.id:Rule Name:Source :Time : Text
#46.1 :updown :LC/0/7/CPU0:Jan 30 16:35:39 2004:ifmgr[130]: %PKT_INFRA-LINK-3-UPDOWN :
Interface POS0/7/0/0, changed state to Down
#46.2 :updown :LC/0/7/CPU0:Jan 30 16:35:41 2004:DI_Partner[50]: %L2-SONET-4-ALARM :
SONET0_7_0_0: SLOS
```

（注） タイムアウト間隔中に生成された後続の PKT_INFRA-LINK-UPDOWN リーフ メッセージおよび L2-SONET-ALARM リーフ メッセージは、リーフ メッセージであるため、ロギング コリレータ バッファに残ります。

次に、**show logging correlator buffer correlationID** コマンドの出力例を示します。 出力には、相関 ID 46、47（PKT_INFRA-LINK-UPDOWN 根本原因メッセージおよび L2-SONET-ALARM 根本原因メッセージに関連付けられた相関 ID）が割り当てられたアラームが表示されます。

```
RP/0/RSP0/CPU0:router# show logging correlator buffer correlationID 46
NO records matching query found
```

その他の関連資料

次の項では、Cisco ASR 9000 シリーズ ルータ でのアラーム ログおよびロギング相関の実装とモニタリングに関連する参考資料を示します。

関連資料

関連項目	マニュアル タイトル
アラームおよびロギング相関コマンド	『*Cisco ASR 9000 Series Aggregation Services Router System Monitoring Command Reference*』の「*Alarm Management and Logging Correlation Commands*」モジュール
ロギング サービス コマンド	『*Cisco ASR 9000 Series Aggregation Services Router System Monitoring Command Reference*』の「*Logging Services Commands*」モジュール
オンボード障害ロギング（OBFL）設定タスク	『*Cisco ASR 9000 Series Aggregation Services Router System Monitoring Command Reference*』の「*Implementing Logging Services*」モジュール

関連項目	マニュアル タイトル
オンボード障害ロギング（OBFL）コマンド	『*Cisco ASR 9000 Series Aggregation Services Router System Monitoring Command Reference*』の「*Onboard Failure Logging Commands*」モジュール
Cisco IOS XR ソフトウェアの XML API 参考資料	『*Cisco IOS XR XML API Guide*』
Cisco IOS XR ソフトウェアのスタートアップ参考資料	『*Cisco ASR 9000 Series Aggregation Services Router Getting Started Guide*』
ユーザ グループとタスク ID に関する情報	『*Cisco ASR 9000 Series Aggregation Services Router System Security Configuration Guide*』の「*Configuring AAA Services*」の章

標準

標準	タイトル
この機能でサポートされる新規の標準または変更された標準はありません。また、既存の標準のサポートは変更されていません。	—

MIB

MIB	MIB のリンク
—	Cisco IOS XR ソフトウェアを使用して MIB を検索およびダウンロードするには、http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml にある Cisco MIB Locator を使用し、[Cisco Access Products] メニューからプラットフォームを選択します。

RFC

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

シスコのテクニカル サポート

説明	リンク
シスコのテクニカル サポート Web サイトでは、製品、テクノロジー、ソリューション、技術的なヒント、およびツールへのリンクなどの、数千ページに及ぶ技術情報が検索可能です。Cisco.com に登録済みのユーザは、このページから詳細情報にアクセスできます。	*http://www.cisco.com/cisco/web/support/index.html*

第 3 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、Cisco ASR 9000 シリーズ ルータで 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Cisco IOS XR ソフトウェアの障害とイベントを処理するために必要な新規および改訂されたタスクについて説明します。

（注） このモジュールでリストされているイベント管理コマンドの詳細については、このモジュールの関連資料，（122 ページ）を参照してください。

Embedded Event Manager ポリシーの設定および管理の機能履歴

リリース	変更内容
リリース 4.0.0	この機能が導入されました。

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

EEM ポリシーを実装するには、次の概念を理解する必要があります。

Event Management

Cisco IOS XR ソフトウェア システムの Embedded Event Management（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 EEM は、これらのイベントを検出して、適切な応答を実行します。 また、EEM を使用して、障害を防止または包含、およびディザスタ リカバリを支援することができます。

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。

また、EEM はシステムの各プロセスの信頼性メトリックも維持します。

システム イベント検出

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。 EEMは、syslogメッセージとのパターンマッチを使用します。 また、タイマー イベント ディテクタにも依存して、特定の日時が発生したとこを検出します。

ポリシーベースのイベント応答

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、*policy handlers* という名前のルーチンに含まれています。 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。

信頼性メトリック

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 ハンドラが存在するか確認します。
  - 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。
  - ポリシー ハンドラが存在しない場合、EEM は何も実行しません。
- イベント通知に加入しているプロセスに通知します。

（注） ポリシー アクションが含まれるスクリプトと、イベントを受信するように加入しているスクリプトの間には違いがあります。 ポリシー アクションが含まれるスクリプトは、ポリシーを実装するものと想定されます。 これらは、再帰を防止するルールによってバインドされています。 通知に加入しているスクリプトは、このようなルールにバインドされることはありません。

- 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 管理ポリシー

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。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。 （Embedded Event Manager スクリプトとスクリプティング インターフェイス（Tcl）， （60 ページ）を参照）。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。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。

Embedded Event Manager スクリプトとスクリプティング インターフェイス（Tcl）

EEM スクリプトは、EEM イベントがパブリッシュされるときに、ポリシーを実装するために使用されます。 EEM スクリプトおよびポリシーは、**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。

EEM は、次の 2 つのタイプの EEM スクリプトを使用します。

- レギュラー EEM スクリプトは、**eem script** CLI コマンドを使用して EEM で識別されます。レギュラー EEM スクリプトはスタンドアロン スクリプトであり、このスクリプトが処理するイベントの定義を包含します。
- *EEM* コールバック スクリプトは、プロセスまたは EEM スクリプトがイベントを処理するように登録されたときに、EEM で識別されます。 EEM コールバック スクリプトは、基本的に名前付き関数で、C 言語 API を使用して EEM で識別されます。

次に、スクリプトでの CLI の使用例を示します。

```
sjc-cde-010:/tftpboot/cnwei/fm> cat test_cli_eem.tcl
::cisco::eem::event_register_syslog occurs 1 pattern $_syslog_pattern maxrun 90

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

set errorInfo ""

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

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

set msg $arr_einfo(msg)
set config_cmds ""

# 2. execute the user-defined config commands
if [catch {cli_open} result] {
    error $result $errorInfo
} else {
    array set cli1 $result
}

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

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

    append config_cmds $_config_cmd1
}

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

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

action_syslog priority info msg "Ran config command $_config_cmd1 $_config_cmd2
```

スクリプト言語

スクリプト言語は、Cisco IOS XR ソフトウェアに実装されている Tool Command Language（Tcl）です。 すべての Embedded Event Manager スクリプトは、Tcl で記述されます。 この Tcl のフル実装はシスコによって拡張され、 **eem** コマンドが追加されて Tcl スクリプトと EEM の間にインターフェイスが提供されました。

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

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

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

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

レギュラー Embedded Event Manager スクリプト

レギュラー EEM スクリプトは、EEM イベントがパブリッシュされるときに、ポリシーを実装するために使用されます。 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。

EEM スクリプト内の最初コード実行可能行は、**eem event register** キーワードである必要があります。 このキーワードは、スクリプトのスケジュール対象となる EEM イベントを特定します。 このキーワードは、指定された EEM イベントを処理するように登録するために、**event manager policy** コンフィギュレーション コマンドにより使用されます。

EEM スクリプトでは、Embedded Event Manager ポリシー Tcl コマンド拡張カテゴリ，（63 ページ）にリストされている任意の EEM スクリプト サービスが使用される場合があります。

EEM スクリプトが存在する場合、この EEM イベントのデフォルト アクション（存在する場合）を実行するか、その他のアクションを実行しないかを EEM に通知するために使用される戻りコードの設定を担当します。 特定のイベントに対して複数のイベント ハンドラがスケジューリングされている場合、前のハンドラからの戻りコードが次のハンドラに渡されます。これにより、値をそのままにするか、更新できます。

（注） EEM スクリプトは、スケジュールの対象となるイベント以外のイベントを処理するように登録することはできません。

Embedded Event Manager コールバック スクリプト

EEM コールバック スクリプトは、eem_handler_spec でこのスクリプトの名前を指定している、以前に登録された EEM イベントに対して発生している EEM イベントの結果として入力されます。

EEM コールバック スクリプトが存在する場合、この EEM イベントのデフォルト アクション（存在する場合）を実行するかどうかを EEM に通知するために使用される戻りコードの設定を担当します。特定のイベントに対して複数のイベント ハンドラがスケジューリングされている場合、前のハンドラからの戻りコードが次のハンドラに渡されます。これにより、値をそのままにするか、更新できます。

（注） EEM コールバック スクリプトは、表 3：Embedded Event Manager Tcl コマンド拡張カテゴリ，（63 ページ）にリストされている EEM スクリプトを自由に使用できます。ただし、**eem event register** キーワードは EEM コールバック スクリプトでは許可されていないため、使用できません。

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。

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

表 *3*：*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 Tcl コマンド拡張 **event_publish** もあります。

カテゴリ	定義
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、または、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 4：2 文字の省略形の指定，（65 ページ） 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 *4* : *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 *5* : *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。

Embedded Event Manager の組み込みアクション	説明
手動による EEM ポリシーの実行	**event manager run** コマンドが EXEC 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。
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. CLI コンフィギュレーション コマンド **event manager policy** *scriptfilename* **username** *username* が入力されます。
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

イベントは、イベント ディテクタと呼ばれるルーチンによって検出されます。 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は、必要に応じてイベントをパブリッシュするために使用可能な情報を処理します。

以下のイベント ディテクタがサポートされています。

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。

EEM イベントの検出および回復の一般的なフロー

EEM は、イベント ディテクタと呼ばれるソフトウェア エージェントを使用してシステム内の異なるコンポーネントのモニタリングをサポートする、柔軟でポリシードリブンのフレームワークです。 EEM サーバ、コア イベント パブリッシャ（イベント ディテクタ）、およびイベント サブスクライバ（ポリシー）の間にな関係があります。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。

イベントまたは障害が検出されると、Embedded Event Manager によって、たとえば OIR イベント パブリッシャなどのイベント パブリッシャから、検出された障害またはイベントの登録が発生しているかどうかが判断されます。 EEM によって、イベント登録情報が、イベント データそのものと、照会されます。 ポリシーによって、検出されたイベントが Tcl コマンド拡張 event_register_xxx

で登録されます。 イベント情報 Tcl コマンド拡張 event_reqinfo は、検出されたイベントに関する情報について Embedded Event Manager に問い合わせるために、ポリシーで使用されます。

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。 *、?、または [...] を使用した POSIX ワイルドカード ファイル名パターンがロード モジュールのパス名でサポートされています。 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 *6*：*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。 メッセージ テキスト フィールドでは POSIX 正規表現パターンがサポートされています。

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 *7*：*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 *8*：*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。

Embedded Event Manager イベント	説明
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

プロセッサ コンプレックス全体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。

ハードウェア カードの信頼性メトリック データ

信頼性メトリック データは、プロセッサ コンプレックスの各ハードウェア カードについて保持されます。 データは、ディスク ID でインデックスが作成されたテーブルに記録されます。

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。
- QNX による異常終了：Neutrino オペレーティング システムがプロセスを異常終了させます。
- 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

ここでは、次の手順について説明します。

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。

環境変数

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 コマンドの *var-value* 引数ではスペースを使用できます。 このコマンドは、*var-name* 引数の後から行の最後までのすべての文字列を *var-value* 引数の一部として解釈します。

event manager environment コマンドを使用して設定された後に、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。

手順の概要

1. **show event manager environment**
2. **configure**
3. **event manager environment** *var-name var-value*
4. リセットするすべての環境変数について手順 3 を繰り返します。
5. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**
6. **show event manager environment**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**show event manager environment** 例： `RP/0/RSP0/CPU0:router# show event manager environment`	すべての EEM 環境変数の名前と値を表示します。
ステップ 2	**configure** 例： `RP/0/RSP0/CPU0:router# 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 3	**event manager environment** *var-name var-value* 例： `RP/0/RSP0/CPU0:router(config)# event manager environment _cron_entry 0-59/2 0-23/1 * * 0-7`	環境変数を新しい値にリセットします。 • *var-name* 引数は、EEM 環境設定変数に割り当てる名前です。 • *var-value* 引数は、環境変数 *var-name* に格納する、スペースを含む文字列です。

	コマンドまたはアクション	目的
		• 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）。 • *var-value* 引数では、スペースを使用できます。 このコマンドは、*var-name* 引数の後から行の最後までのすべての文字列を *var-value* 引数の一部として解釈します。
ステップ 4	リセットするすべての環境変数について手順 3 を繰り返します。	—

ステップ 5	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

ステップ 6	**show event manager environment** 例： `RP/0/RSP0/CPU0:router# show event manager environment`	すべての EEM 環境変数のリセットされた名前と値を表示します。手順 3 で設定した環境変数名と値を確認できます。

次の作業

EEM 環境変数を設定した後、Embedded Event Manager ポリシーの登録, （76 ページ）に従って、登録できるポリシーを確認し、ポリシーを登録します。

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 ポリシーを登録します。

Embedded Event Manager ポリシー

EEM ポリシーを登録するには、グローバル コンフィギュレーション モードで **event manager policy** コマンドを使用します。 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。

ユーザ名

EEM ポリシーを登録するには、スクリプトを実行するために使用するユーザ名を指定する必要があります。 この名前は、現在ログインしているユーザと異なる名前でもかまいません。ただし、登録するユーザは、スクリプトを実行するユーザ名のスーパーセットである権限を所有している必要があります。所有していない場合、スクリプトは登録されず、コマンドは拒否されます。また、スクリプトを実行するユーザ名は、登録される EEM ポリシーが実行するコマンドへのアクセス権を所有している必要があります。

（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。

持続時間

ユーザ名に対するオプションの **persist-time** キーワードも定義できます。 **persist-time** キーワードは、ユーザ名認証が有効な時間（秒数）を定義します。 スクリプトの初回登録時は、スクリプトに対して設定された username が認証されます。 スクリプトの登録後は、スクリプトの実行ごとにユーザ名が再度認証されます。 AAA サーバがダウンしている場合は、ユーザ名の認証をメモリから読み取れます。 このユーザ名の認証をメモリに保持する秒数は、**persist-time** キーワードによって決定します。

- AAA サーバがダウンしていて **persist-time** キーワードが期限切れになっていない場合、ユーザ名はメモリから認証され、スクリプトが実行されます。
- AAA サーバがダウンしていて **persist-time** キーワードが期限切れの場合、ユーザ認証が失敗して、スクリプトは実行されません。

persist-time キーワードには、次の値を使用できます。

- デフォルトの **persist-time** は、3600 秒（1 時間）です。 **persist-time** を 1 時間に設定するには、**persist-time** キーワードを指定せずに **event manager policy** コマンドを入力します。
- ユーザ名の認証がキャッシュされないようにするには、0 を指定します。 AAA サーバがダウンしている場合、ユーザ名は認証されず、スクリプトは実行されません。
- ユーザ名が無効としてマーキングされないようにするには、**infinite** を指定します。 キャッシュに保持されたユーザ名の認証は、期限切れになりません。 AAA サーバがダウンしている場合、ユーザ名はキャッシュから認証されます。

system または user キーワード

system または **user** キーワードを指定せずに **event manager policy** コマンドを入力すると、指定されたポリシー ファイルが、まずシステム ポリシー ディレクトリで検索されます。 システム ポリシーディレクトリ内でファイルが見つかった場合、そのポリシーがシステム ポリシーとして登録されます。 指定されたポリシー ファイルがシステム ポリシー ディレクトリ内で見つからなかった場合、ユーザ ポリシー ディレクトリが検索されます。 指定されたファイルがユーザ ポリシーディレクトリ内で見つかった場合、そのポリシー ファイルがユーザ ポリシーとして登録されます。 同じ名前を持つポリシー ファイルがシステム ポリシー ディレクトリとユーザ ポリシー ディレクトリの両方で見つかった場合、システム ポリシー ディレクトリ内のポリシー ファイルが優先され、システム ポリシーとして登録されます。

ポリシーを登録した後でその登録内容を確認するには、**show event manager policy registered** コマンドを使用します。 出力では、登録済みポリシーの情報が 2 つの部分にわかれて表示されます。各ポリシーの説明の最初の行には、ポリシーに割り当てられているインデックス番号、ポリシーのタイプ（システムまたはユーザ）、登録済みイベントのタイプ、ポリシーの登録日時、およびポリシーファイルの名前が表示されます。各ポリシーの説明の残りの行には、登録済みイベントとイベントの処理方法に関する情報が表示されます。この情報は、ポリシー ファイルを構成する Tcl コマンドの引数から直接取得されます。

手順の概要

1. **show event manager policy available** [**system** | **user**]
2. **configure**
3. **event manager policy** *policy-name* **username** *username* [**persist-time** { *seconds* | **infinite** }] | **type** { **system** | **user** }
4. 登録するすべての EEM ポリシーについて手順 3 を繰り返します。
5. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**
6. **show event manager policy registered**

手順の詳細

	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/RSP0/CPU0:router# show event manager policy available`	登録可能なすべての EEM ポリシーを表示します。 • オプションの **system** キーワードを入力すると、使用可能なすべてのシステム ポリシーが表示されます。 • オプションの **user** キーワードを入力すると、使用可能なすべてのユーザ ポリシーが表示されます。
ステップ 2	**configure** 例： `RP/0/RSP0/CPU0:router# 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 3	**event manager policy** *policy-name* **username** *username* [**persist-time** { *seconds* | **infinite** }] | **type** { **system** | **user** } 例： `RP/0/RSP0/CPU0:router(config)# event manager policy cron.tcl username tom type user`	EEM ポリシーを EEM に登録します。 • 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。 • 必要な **username** キーワードと引数を入力します。ここで、*username* はスクリプトを実行するユーザ名です。 • ユーザ名認証をメモリに保持する時間を指定するには、オプションの **persist-time** キーワードを入力します。 ◦ **persist-time** キーワードに *seconds*（秒数）を入力します。 ◦ 認証を永続的なものにするには、**infinite** キーワードを入力します（認証は期限切れになりません）。 • オプションの **type system** キーワードを入力すると、シスコによって定義されたシステム ポリシーが登録されます。 • オプションの **type user** キーワードを入力すると、ユーザ定義ポリシーが登録されます。 （注） EEM ポリシーを登録する前に、AAA 認可（aaa authorization eventmanage 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、『*Cisco ASR 9000 Series Aggregation Services Router System Security Configuration Guide*』の「*Configuring AAA Services* 」モジュールを参照してください。
ステップ 4	登録するすべての EEM ポリシーについて手順 3 を繰り返します。	—

	コマンドまたはアクション	目的

ステップ 5	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。

ステップ 6	**show event manager policy registered** 例： `RP/0/RSP0/CPU0:router# show event manager policy registered`	登録済みのすべての EEM ポリシーを表示します。手順 3 の確認を行うことができます。

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。

ここでは、次のタスク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。

はじめる前に

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。 ポリシーの例については、EEM サンプル ポリシー, （88 ページ）を参照してください。 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。

手順の概要

1. **show event manager environment** [**all** | *environment-name*]
2. **configure**
3. **event manager environment** *var-name* [*var-value*]
4. ステップ 3, （80 ページ） を繰り返して、ステップ 5, （81 ページ） で登録されるポリシーに必要なすべての環境変数を設定します。
5. **event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]]
6. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**show event manager environment** [**all**	*environment-name*] 例： `RP/0/RSP0/CPU0:router# show event manager environment all`	（任意）EEM 環境変数の名前と値を表示します。 • **all** キーワードは、すべての EEM 環境変数を表示します。 • *environment-name* 引数は、指定された環境変数に関する情報を表示します。
ステップ 2	**configure** 例： `RP/0/RSP0/CPU0:router# 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 3	**event manager environment** *var-name* [*var-value*] 例： `RP/0/RSP0/CPU0:router(config)# event manager environment _cron_entry 0-59/2 0-23/1 * * 0-7`	環境変数を新しい値にリセットします。 • *var-name* 引数は、EEM 環境設定変数に割り当てる名前です。 • *var-value* 引数は、環境変数 *var-name* に格納する、スペースを含む文字列です。

	コマンドまたはアクション	目的
		• 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）。 • *var-value* 引数では、スペースを使用できます。 このコマンドは、*var-name* 引数の後から行の最後までのすべての文字列を *var-value* 引数の一部として解釈します。
ステップ 4	ステップ 3, （80 ページ） を繰り返して、ステップ 5, （81 ページ） で登録されるポリシーに必要なすべての環境変数を設定します。	—
ステップ 5	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]] 例： `RP/0/RSP0/CPU0:router(config)# event manager policy tm_cli_cmd.tcl username user_a type system`	ポリシー内で定義された指定イベントが発生した場合に、EEM ポリシーを実行するよう、定義します。 • シスコによって定義されたシステム ポリシーを登録するには、**system** キーワードを使用します。 • ユーザ定義のシステムポリシーを登録するには、**user** キーワードを使用します。 • ユーザ名認証が有効な期間を指定するには、**persist-time** キーワードを使用します。 この例では、tm_cli_cmd.tcl という名前の EEM サンプルポリシーが、システム ポリシーとして定義されます。

ステップ 6	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーションファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュ

	コマンドまたはアクション	目的
		レーションセッションは終了せず、設定変更もコミットされません。 • 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

登録済みの EEM ポリシーの表示

登録済みの EEM ポリシーを表示するには、次の任意の作業を実行します。

手順の概要

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

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**show event manager policy registered** [**event-type** *type*] [**system** | **user**] [**time-ordered** | **name-ordered**] 例： `RP/0/RSP0/CPU0:router# show event manager policy registered system`	現在登録されているポリシーについての情報を表示します。 • **event-type** キーワードを指定すると、特定のイベント タイプで登録されているポリシーが表示されます。 • **time-ordered** キーワードを指定すると、現在登録されているポリシーの情報が、時間でソートされて表示されます。 • **name-ordered** キーワードを指定すると、ポリシー名順（アルファベット順）にポリシーが表示されます。

EEM ポリシーの登録解除

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

手順の概要

1. **show event manager policy registered** [**event-type** *type*] [**system** | **user**] [**time-ordered** | **name-ordered**]
2. **configure**
3. **no event manager policy** *policy-name*
4. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**
5. ステップ 1, （83 ページ） を繰り返して、ポリシーが削除されたことを確認します。

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**show event manager policy registered** [**event-type** *type*] [**system** | **user**] [**time-ordered** | **name-ordered**] 例： `RP/0/RSP0/CPU0:router# show event manager policy registered system`	現在登録されているポリシーについての情報を表示します。 • **event-type** キーワードを指定すると、特定のイベント タイプで登録されているポリシーが表示されます。 • **time-ordered** キーワードを指定すると、現在登録されているポリシーの情報が、時間でソートされて表示されます。 • **name-ordered** キーワードを指定すると、ポリシー名順（アルファベット順）にポリシーが表示されます。
ステップ 2	**configure** 例： `RP/0/RSP0/CPU0:router# 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 3	**no event manager policy** *policy-name* 例： `RP/0/RSP0/CPU0:router(config)# no event manager policy tm_cli_cmd.tcl`	ポリシーを登録解除するためにEEMポリシーを設定から削除します。
ステップ 4	次のいずれかのコマンドを使用します。 • **end** • **commit** 例： `RP/0/RSP0/CPU0:router(config)# end`	設定変更を保存します。 • **end** コマンドを実行すると、変更をコミットするように要求されます。 `Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`

	コマンドまたはアクション	目的
	または `RP/0/RSP0/CPU0:router(config)# commit`	◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。 ◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。 ◦ cancel と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。 • 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。
ステップ 5	ステップ 1, （83 ページ） を繰り返して、ポリシーが削除されたことを確認します。	—

EEM ポリシー実行の一時停止

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

手順の概要

1. **show event manager policy registered** [**event-type** *type*] [**system** | **user**] [**time-ordered** | **name-ordered**]
2. **configure**
3. **event manager scheduler suspend**
4. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**show event manager policy registered** [**event-type** *type*] [**system** | **user**] [**time-ordered** | **name-ordered**] 例： `RP/0/RSP0/CPU0:router# show event manager policy registered system`	現在登録されているポリシーについての情報を表示します。 • **event-type** キーワードを指定すると、特定のイベント タイプで登録されているポリシーが表示されます。 • **time-ordered** キーワードを指定すると、現在登録されているポリシーの情報が、時間でソートされて表示されます。 • **name-ordered** キーワードを指定すると、ポリシー名順（アルファベット順）にポリシーが表示されます。
ステップ 2	**configure** 例： `RP/0/RSP0/CPU0:router# 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 3	**event manager scheduler suspend** 例： `RP/0/RSP0/CPU0:router(config)# event manager scheduler suspend`	すべての EEM ポリシーの実行がすぐに一時停止されます。

ステップ 4	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

	コマンドまたはアクション	目的

EEM ポリシーの管理

ユーザ ライブラリ ファイルまたはユーザ定義 EEM ポリシーの保存に使用するディレクトリを指定するには、この作業を実行します。

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

手順の概要

1. **show event manager directory user** [**library** | **policy**]
2. **configure**
3. **event manager directory user** {**library** *path* | **policy** *path*}
4. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**show event manager directory user** [**library** | **policy**] 例： `RP/0/RSP0/CPU0:router# show event manager directory user library`	EEM ユーザ ライブラリまたはポリシー ファイルの保存に使用するディレクトリを表示します。 • オプションの **library** キーワードによって、ユーザ ライブラリ ファイルに使用されるディレクトリが表示されます。 • オプションの **policy** キーワードによって、ユーザ定義 EEM ポリシーに使用されるディレクトリが表示されます。
ステップ 2	**configure** 例： `RP/0/RSP0/CPU0:router# 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。

	コマンドまたはアクション	目的
ステップ 3	**event manager directory user** {**library** *path* | **policy** *path*} 例： `RP/0/RSP0/CPU0:router(config)# event manager directory user library disk0:/usr/lib/tcl`	ユーザ ライブラリ ファイルまたはユーザ定義 EEM ポリシーの保存に使用するディレクトリを指定します。 • ユーザ ディレクトリへの絶対パス名を指定するには、*path* 引数を指定します。

ステップ 4	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

EEM を使用したソフトウェア モジュール方式プロセスの信頼性メトリック

Cisco IOS XR ソフトウェア プロセスの信頼性メトリックを表示するには、この省略可能なタスクを実行します。

手順の概要

1. **show event manager metric process** {**all** | *job-id* | *process-name*} **location** {**all** | *node-id*}

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**show event manager metric process** {**all** | *job-id* | *process-name*} **location** {**all** | *node-id*} 例： `RP/0/RSP0/CPU0:router# show event manager environment`	プロセスの信頼性メトリック データを表示します。 システムでは、プロセスの開始時と終了時にレコードが保存され、このデータが、信頼性分析の基本データとして使用されます。

EEM サンプル ポリシー

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

Cisco IOS XR ソフトウェアには、サンプル ポリシーが含まれています（表「*EEM* サンプル ポリシーの説明」を参照してください）。 サンプル ポリシーは、ユーザ ディレクトリにコピーして変更できます。 Tcl は、現在ポリシー作成のためにシスコでサポートされている唯一のスクリプト言語です。 Tcl ポリシーは、Emacs などのテキスト エディタを使用して変更できます。 ポリシーは、定義されている経過時間の秒数以内で実行する必要があり、時間変数はポリシー内で設定できます。 デフォルトは 20 秒です。

EEM サンプル ポリシーは、CLI を使用してルータに表示される場合があります。
`Show event manager policy available system`

この表ではサンプル EEM ポリシーについて説明します。

表 *9*：*EEM* サンプル ポリシーの説明

ポリシーの名前	説明
periodic_diag_cmds.tcl	このポリシーは、cron エントリ _cron_entry_diag の期限が切れたときにトリガーされます。その後、この固定セットの出力が固定のコマンドセットに対して収集され、出力が電子メールで送信されます。

ポリシーの名前	説明
periodic_proc_avail.tcl	このポリシーは、cron エントリ _cron_entry_procavail の期限が切れたときにトリガーされます。その後、この固定セットの出力が固定のコマンドセットに対して収集され、出力が電子メールで送信されます。
periodic_sh_log.tcl	このポリシーは、cron エントリ _cron_entry_log の期限が切れたときにトリガーされ、show log コマンドとその他いくつかのコマンドの出力が収集されます。環境変数 _log_past_hours が設定されている場合、最後の _log_past_hours 時間に生成されたログ メッセージが収集されます。そうでない場合、ログ全体が収集されます。
sl_sysdb_timeout.tcl	このポリシーは、スクリプトが sysdb タイムアウト ios_msgs を探すときにトリガーされ、show コマンドの出力が取得されます。出力は、ブロックしているプロセスの名前が付けられたファイルに書き込まれます。
tm_cli_cmd.tcl	このポリシーは、設定可能な CRON エントリを使用して実行されます。設定可能な CLI コマンドが実行され、結果が電子メールで送信されます。
tm_crash_hist.tcl	このポリシーは、毎日夜中に実行され、指定された電子メールアドレスにプロセス クラッシュ履歴レポートが電子メールで送信されます。

使用可能なサンプル ポリシーおよびその実行方法についての詳細は、EEM イベント ディテクタのデモ：例，（111 ページ）を参照してください。

手順の概要

1. **show event manager policy available** [**system** | **user**]
2. **configure**
3. **event manager directory user** {**library** *path* | **policy** *path*}
4. **event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]]
5. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	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/RSP0/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	**configure** 例： `RP/0/RSP0/CPU0:router# 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 3	**event manager directory user** {**library** *path* | **policy** *path*} 例： `RP/0/RSP0/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 4	**event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]] 例： `RP/0/RSP0/CPU0:router(config)# event manager policy test.tcl username user_a type user`	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。

ステップ 5	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレー

	コマンドまたはアクション	目的
		ションセッションは終了せず、設定変更もコミットされません。 • 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。

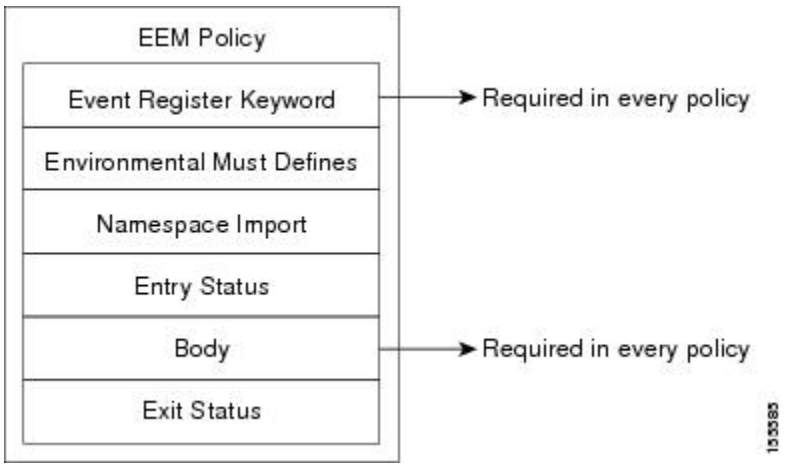
Tcl を使用した EEM ポリシーのプログラミング

Tcl コマンド拡張を使用してポリシーをプログラムするには、この作業を実行します。 既存のポリシーをコピーし、変更することを推奨します。 EEM Tcl ポリシーには、event_register Tcl コマンド拡張と本体の 2 つの必須部分が存在する必要があります。 Tcl ポリシーの構造と要件, （91 ページ） にある他のすべてのセクションは、オプションです。

Tcl ポリシーの構造と要件

すべての EEM ポリシーでは、図 2：Tcl ポリシーの構造と要件 , （91 ページ） 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 *2*：*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。 使用可能な EEM **event_register** Tcl コマンド拡張のリストについては、Embedded Event

Manager イベント登録 Tcl コマンド拡張, （124 ページ）を参照してください。 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、次のものを含める必要があります。

- 検出されたイベントに関する情報の EEM への問い合わせに使用される **event_reqinfo** イベント情報の Tcl コマンド拡張。 使用可能な EEM イベント情報の Tcl コマンド拡張のリストについては、Embedded Event Manager イベント情報 Tcl コマンド拡張, （162 ページ）を参照してください。
- 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。 使用可能な EEM アクションの Tcl コマンド拡張のリストについては、Embedded Event Manager アクション Tcl コマンド拡張, （187 ページ）を参照してください。
- 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。 使用可能な EEM システム情報の Tcl コマンド拡張のリストについては、Embedded Event Manager システム情報 Tcl コマンド拡張, （205 ページ）を参照してください。
- 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。 使用可能な SMTP ライブラリの Tcl コマンド拡張のリストについては、SMTP ライブラリのコマンド拡張, （218 ページ）を参照してください。 使用可能な CLI ライブラリの Tcl コマンド拡張のリストについては、CLI ライブラリのコマンド拡張, （221 ページ）を参照してください。
- 他のポリシーによって使用される Tcl 変数の保存に使用される **context_save** および **context_retrieve** の Tcl コマンド拡張。

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

```
# Query the event info and log a message.
```

```
array set arr_einfo [event_reqinfo]
if {$_cerrno != 0} {
  set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
    $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
  error $result
}
global timer_type timer_time_sec
set timer_type $arr_einfo(timer_type)
  set timer_time_sec $arr_einfo(timer_time_sec)
# Log a message.
set msg [format "timer event: timer type %s, time expired %s" \
$timer_type [clock format $timer_time_sec]]
action_syslog priority info msg $msg
if {$_cerrno != 0} {
  set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
    $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
  error $result
}
```

EEM 開始ステータス

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

開始ステータスの指定には、次の 3 種類の値のいずれかを使用できます。

- 0（以前のポリシーは成功した）
- 0 以外（以前のポリシーは失敗した）
- Undefined（以前実行されたポリシーはない）

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 が設定されるたびに、他の 4 つの Tcl グローバル変数が _cerrno から分岐し、それとともに設定されます（_cerr_sub_num、_cerr_sub_err、_cerr_posix_err、および _cerr_str）。

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

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

```
}
```

_cerrno：32 ビット エラー戻り値

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

```
XYSSSSSSSSSSSSSEEEEEEEEPPPPPPPPP
```

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

```
862439AE
```

この番号は、次の 32 ビット値として解釈されます。

```
10000110001001000011100110101110
```

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

表 *10*：*_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 11：エラー クラス エンコーディング，（95 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。
PPPPPPPP	パススルー POSIX エラー コード（9 ビット = 値 512）。 これは、32 ビット シーケンスの最後で、このエラー コードに対応する文字列は、$_cerr_posix_err に含まれています。

XY のエラー クラス エンコーディング

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

表 *11*：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
01	CERR_CLASS_INFO
10	CERR_CLASS_WARNING
11	CERR_CLASS_FATAL

ゼロのエラー戻り値は、SUCCESS を示します。

手順の概要

1. **show event manager policy available** [**system** | **user**]
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。
3. 必要な event_register Tcl コマンド拡張を定義します。
4. 適切な名前空間を、::cisco 階層構造に追加します。
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。
6. スクリプトの本体をプログラムします。
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。
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。
9. Cisco エラー番号（_cerrno）の Tcl グローバル変数を設定します。
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。
11. **configure**
12. **event manager directory user** {**library** *path* | **policy** *path*}
13. **event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]]
14. 次のいずれかのコマンドを使用します。
    - **end**
    - **commit**
15. ポリシーを実行し、ポリシーを観察します。
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。

手順の詳細

	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/RSP0/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。	—

	コマンドまたはアクション	目的
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

	コマンドまたはアクション	目的
		（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） EXEC モードで **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。 • 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。

	コマンドまたはアクション	目的
		• 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、_cerr_posix_err、および _cerr_str）。
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 4：2 文字の省略形の指定, （65 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, （64 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。

	コマンドまたはアクション	目的
ステップ 11	**configure** 例： `RP/0/RSP0/CPU0:router# 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/RSP0/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/RSP0/CPU0:router(config)# event manager policy test.tcl username user_a type user`	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。

ステップ 14	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

	コマンドまたはアクション	目的
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。

手順の概要

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。
2. **tclsh**
3. **auto_mkindex** *directory_name* ***.tcl**
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, （102 ページ） から Tcl ライブラリ ファイルをコピーし、ステップ 3, （102 ページ） から tclIndex ファイルをコピーします。
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。
6. **configure**
7. **event manager directory user library** *path*
8. **event manager directory user policy** *path*
9. **event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]]
10. **event manager run** *policy* [*argument*]
11. 次のいずれかのコマンドを使用します。
    - **end**
    - **commit**

手順の詳細

	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。

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.`

	コマンドまたはアクション	目的
		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, （102 ページ）から Tcl ライブラリ ファイルをコピーし、ステップ 3, （102 ページ） から tclIndex ファイルをコピーします。	—

ステップ 5	Tcl で記述されたユーザ定義 EEM ポリシー ファイルを、ターゲット 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, （103 ページ） 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** 例： RP/0/RSP0/CPU0:router# 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 7	**event manager directory user library** *path* 例： RP/0/RSP0/CPU0:router(config)# event manager directory user library disk2:/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, （103 ページ）のファイルがコピーされたディレクトリです。

	コマンドまたはアクション	目的
ステップ 8	**event manager directory user policy** *path* 例： `RP/0/RSP0/CPU0:router(config)# event manager directory user policy disk2:/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, （103 ページ）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/RSP0/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。
ステップ 10	**event manager run** *policy* [*argument*] 例： `RP/0/RSP0/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	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。

	コマンドまたはアクション	目的

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。

手順の概要

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。
2. **tclsh**
3. **pkg_mkindex** *directory_name* ***.tcl**
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, （106 ページ） から Tcl パッケージ ファイルをコピーし、ステップ 3, （106 ページ） から pkgIndex ファイルをコピーします。
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。
6. **configure**
7. **event manager directory user library** *path*
8. **event manager directory user policy** *path*
9. **event manager policy** *policy-name* **username** *username* [**persist-time** [*seconds* | **infinite**] | **type** [**system** | **user**]]
10. **event manager run** *policy* [*argument*]
11. 次のいずれかのコマンドを使用します。
    - **end**
    - **commit**

手順の詳細

	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。
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

	コマンドまたはアクション	目的

		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, （106 ページ） から Tcl パッケージ ファイルをコピーし、ステップ 3, （106 ページ） から pkgIndex ファイルをコピーします。	—

ステップ 5	Tcl で記述されたユーザ定義 EEM ポリシー ファイルを、ターゲット 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, （107 ページ） 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?"`

	コマンドまたはアクション	目的
ステップ 6	**configure** 例： `RP/0/RSP0/CPU0:router# 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 7	**event manager directory user library** *path* 例： `RP/0/RSP0/CPU0:router(config)# event manager directory user library disk2:/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, （107 ページ）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/RSP0/CPU0:router(config)# event manager directory user policy disk2:/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, （107 ページ）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/RSP0/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： `RP/0/RSP0/CPU0:router(config)# event manager run packagetest.tcl`	手動で EEM ポリシーを実行します。

ステップ 11	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。

	コマンドまたはアクション	目的
		◦ no と入力すると、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。 ◦ cancel と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。 • 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

イベント管理ポリシーの設定例

ここでは、次の設定例を示します。

環境変数の設定：例

次の設定は、環境変数 cron_entry を設定します。

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router#(config)# event manager environment _cron_entry 0-59/2 0-23/1 * * 0-7
```

ユーザ定義 Embedded Event Manager ポリシーの登録：例

次の設定では、ユーザ定義イベント管理ポリシーを登録します。

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# event manager policy cron.tcl username tom user
```

使用可能なポリシーの表示：例

使用可能なポリシーを表示する **show event manager policy available** コマンドの出力例は、次のとおりです。

```
RP/0/RSP0/CPU0:router# show event manager policy available

No.  Type    Time Created                 Name
```

```
1    system  Mon Mar 15 21:32:14 2004      periodic_diag_cmds.tcl
2    system  Mon Mar 15 21:32:14 2004      periodic_proc_avail.tcl
3    system  Mon Mar 15 21:32:16 2004      periodic_sh_log.tcl
4    system  Mon Mar 15 21:32:16 2004      tm_cli_cmd.tcl
5    system  Mon Mar 15 21:32:16 2004      tm_crash_hist.tcl
```

Embedded Event Manager プロセスの表示：例

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。 信頼性メトリックデータを表示する **show event manager metric process** コマンドの出力例は、次のとおりです。

```
RP/0/RSP0/CPU0:router# show event manager metric process all location 0/1/CPU0

=====================================
job id: 78, node name: 0/1/CPU0
process name: wd-critical-mon, instance: 1
--------------------------------
last event type: process start
recent start time: Mon Sep 10 21:36:49 2007
recent normal end time: n/a
recent abnormal end time: n/a
number of times started: 1
number of times ended normally: 0
number of times ended abnormally: 0
most recent 10 process start times:
--------------------------
Mon Sep 10 21:36:49 2007
--------------------------

most recent 10 process end times and types:

cumulative process available time: 59 hours 33 minutes 42 seconds 638 milliseconds
cumulative process unavailable time: 0 hours 0 minutes 0 seconds 0 milliseconds
process availability:  1.000000000
number of abnormal ends within the past 60 minutes (since reload): 0
number of abnormal ends within the past 24 hours (since reload): 0
number of abnormal ends within the past 30 days (since reload): 0
=====================================
job id: 56, node name: 0/1/CPU0
process name: dllmgr, instance: 1
--------------------------------
last event type: process start
recent start time: Mon Sep 10 21:36:49 2007
recent normal end time: n/a
recent abnormal end time: n/a
number of times started: 1
number of times ended normally: 0
number of times ended abnormally: 0
most recent 10 process start times:
--------------------------
Mon Sep 10 21:36:49 2007
--------------------------

most recent 10 process end times and types:

cumulative process available time: 59 hours 33 minutes 42 seconds 633 milliseconds
cumulative process unavailable time: 0 hours 0 minutes 0 seconds 0 milliseconds
process availability:  1.000000000
number of abnormal ends within the past 60 minutes (since reload): 0
number of abnormal ends within the past 24 hours (since reload): 0
number of abnormal ends within the past 30 days (since reload): 0
```

```
====================================
.
.
.
```

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

ここでは、次の設定例を示します。

EEM イベント ディテクタのデモ：例

この例では、サンプル ポリシーを使用して、Embedded Event Manager ポリシーの使用方法を示します。 後述のセクションで、サンプル ポリシーの使用方法について説明します。

EEM サンプル ポリシーの説明

設定例では、1 つのサンプル EEM ポリシーを取り上げています。 tm_cli_cmd.tcl は、設定可能な CRON エントリを使用して実行されます。 このポリシーでは、設定可能な CLI コマンドが実行され、結果が電子メールで送信されます。

サンプル ポリシーのイベント マネージャ環境変数

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

次の表に、電子メール変数の一覧を示します。

表 *12*：サンプル ポリシーで使用される電子メール特有の環境変数

環境変数	説明	例
_domainname	デフォルト ドメイン名。	example.com
_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。	mailserver.example.com
_email_to	電子メールの送信先アドレス。	engineering@example.com
_email_from	電子メールの送信元アドレス。	devtest@example.com

環境変数	説明	例
_email_cc	電子メールのコピーの送信先アドレス。	manager@example.com

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

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

環境変数	説明	例
_config_cmd1	実行される最初のコンフィギュレーション コマンド。	**interface gigabitEthernet1/0/5/0**
_config_cmd2	実行される 2 番目のコンフィギュレーション コマンド。 この変数は任意で、指定する必要はありません。	**no shutdown**
_syslog_pattern	ポリシー実行時を決定するために syslog メッセージを比較するために使用する正規表現パターン マッチ文字列。	.*UPDOWN.*FastEthernet0/0.*

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

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

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

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

表 *15*：*tm_crash_reporter.tcl* ポリシーで使用される環境変数

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

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

表 *16*：*tm_fsys_usage.tcl* ポリシーで使用される環境変数

環境変数	説明	例
_tm_fsys_usage_cron	event_register TCL コマンド拡張で使用される CRON 仕様。 指定されない場合、tm_fsys_usage.tcl ポリシーが 1 分に 1 回トリガーされます。この変数は任意で、指定する必要はありません。	0-59/1 0-23/1 * * 0-7
_tm_fsys_usage_debug	この変数が値 1 に設定された場合、システムのすべてのエントリのディスク使用率情報が表示されます。 この変数は任意で、指定する必要はありません。	1
_tm_fsys_usage_freebytes	システムまたは特定のプレフィックスの空きバイト数しきい値。 空きスペースが所定の値を下回ると、警告が表示されます。 この変数は任意で、指定する必要はありません。	disk2:98000000

環境変数	説明	例
_tm_fsys_usage_percent	システムまたは特定のプレフィックスのディスク使用割合しきい値。 ディスク使用割合が所定の割合を超えると、警告が表示されます。 指定されない場合、すべてのシステムのデフォルトのディスク使用割合は、80% です。 この変数は任意で、指定する必要はありません。	nvram:25 disk2:5

一部の EEM ポリシーの登録

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

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

すべてのサンプル ポリシーの基本設定の詳細

Embedded Event Manager（EEM）から電子メールを送信できるようにするには、**hostname** コマンドと **ip domain-name** コマンドを設定する必要があります。 EEM 環境変数も設定する必要があります。 Cisco IOS XR ソフトウェア イメージのブート後、次の初期設定を使用し、ネットワークで適切な値を置き換えます。 tm_fsys_usage サンプル ポリシーの環境変数（表 16： tm_fsys_usage.tcl ポリシーで使用される環境変数, （113 ページ） を参照）はすべて任意で、ここではそのリストは示されていません。

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

サンプル ポリシーの使用

ここでは、4 つの Tcl サンプル ポリシーを使用する方法を示す次の設定シナリオについて説明します。

sl_intf_down.tcl サンプル ポリシーの実行

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

次に、このポリシーの使用方法を示すサンプル設定について説明します。 EXEC モードで、**show event manager policy registered** コマンドを使用し、現在ポリシーが登録されていないことを確認します。 次のコマンドは **show event manager policy available** コマンドで、インストールできるポリシーが表示されます。 **configure** コマンドを入力してグローバル コンフィギュレーション モードを開始後に、**event manager policy** コマンドを使用して、EEM で sl_intf_down.tcl ポリシーを登録できます。 グローバル コンフィギュレーション モードを終了後、**show event manager policy registered** コマンドを再度入力し、ポリシーが登録されたことを確認します。

インターフェイスがダウンするときに、ポリシーが実行されます。 **show event manager environment** コマンドを入力し、現在の環境変数の値を表示します。 _syslog_pattern EEM 環境変数で指定されたインターフェイスのケーブルを取り外します（またはシャットダウンを設定します）。 インターフェイスがダウンし、インターフェイスがダウンしていることについての Syslog メッセージを記録する Syslog デーモンのプロンプトが表示されて、Syslog イベント ディテクタが呼び出されます。

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

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

tm_cli_cmd.tcl サンプル ポリシーの実行

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

次に、このポリシーの使用方法を示すサンプル設定について説明します。 EXEC モードで、**show event manager policy registered** コマンドを入力し、現在ポリシーが登録されていないことを確認します。 次のコマンドは **show event manager policy available** コマンドで、インストールできるポリシーが表示されます。 **configure** コマンドを入力してグローバル コンフィギュレーション モードを開始後に、**event manager policy** コマンドを使用して、EEM で tm_cli_cmd.tcl ポリシーを登録できます。 グローバル コンフィギュレーション モードを終了後、**show event manager policy registered** コマンドを入力し、ポリシーが登録されたことを確認します。

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

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

tm_crash_reporter.tcl サンプル ポリシーの実行

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

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

次に、このポリシーの使用方法を示すサンプル設定について説明します。 EXEC モードで、**show event manager policy registered** コマンドを入力し、現在ポリシーが登録されていないことを確認します。 次に、**show event manager policy available** コマンドを入力し、インストールできるポリシーを表示します。 **configure** コマンドを入力してグローバル コンフィギュレーション モードを開始後に、**event manager policy** コマンドを使用して、EEM で tm_crash_reporter.tcl ポリシーを登

録できます。 グローバル コンフィギュレーション モードを終了後、**show event manager policy registered** コマンドを入力し、ポリシーが登録されたことを確認します。

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

tm_fsys_usage.tcl サンプル ポリシーの実行

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

次に、このポリシーの使用方法を示すサンプル設定について説明します。 ユーザ EXEC モードで、**show event manager policy registered** コマンドを入力し、現在ポリシーが登録されていないことを確認します。 次に、**show event manager policy available** コマンドを入力し、インストールできるポリシーを表示します。 **configure** コマンドを入力してグローバル コンフィギュレーション モードを開始後に、**event manager policy** コマンドを使用して、EEM で tm_fsys_usage.tcl ポリシーを登録できます。 グローバル コンフィギュレーション モードを終了後、**show event manager policy registered** コマンドを再度入力し、ポリシーが登録されたことを確認します。 tm_fsys_usage.tcl ポリシーで使用される任意の環境変数のいずれかを設定した場合、**show event manager environment** コマンドによって、設定された変数が表示されます。

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

Tcl でポリシーをプログラミングするサンプル スクリプト例

ここでは、EEM システム ポリシーとして含まれている 2 つのサンプル ポリシーについて説明します。 これらのポリシーの詳細については、EEM イベント ディテクタのデモ：例，（111 ページ）を参照してください。

tm_cli_cmd.tcl サンプル ポリシー

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

```
::cisco::eem::event_register_timer cron name crontimer2 cron_entry $_cron_entry maxrun 240
#-------------------------------------------------------------------
# EEM policy that will periodically execute a cli command and email the
# results to a user.
```

```
#
# July 2005, Cisco EEM team
#
# Copyright (c) 2005 by cisco Systems, Inc.
# All rights reserved.
#------------------------------------------------------------------
### The following EEM environment variables are used:
###
### _cron_entry (mandatory)            - A CRON specification that determines
###                                      when the policy will run. See the
###                                      IOS XR Embedded Event Manager
###                                      documentation for more information
###                                      on how to specify a cron entry.
### Example: _cron_entry                 0-59/1 0-23/1 * * 0-7
###
### _log_file (mandatory without _email_....)
###                                    - A filename to append the output to.
###                                      If this variable is defined, the
###                                      output is appended to the specified
###                                      file with a timestamp added.
### Example: _log_file                   disk0:/my_file.log
###
### _email_server (mandatory without _log_file)
###                                    - A Simple Mail Transfer Protocol (SMTP)
###                                      mail server used to send e-mail.
### Example: _email_server               mailserver.example.com
###
### _email_from (mandatory without _log_file)
###                                    - The address from which e-mail is sent.
### Example: _email_from                 devtest@example.com
###
### _email_to (mandatory without _log_file)
###                                    - The address to which e-mail is sent.
### Example: _email_to                   engineering@example.com
###
### _email_cc (optional)               - The address to which the e-mail must
###                                      be copied.
### Example: _email_cc                   manager@example.com
###
### _show_cmd (mandatory)              - The CLI command to be executed when
###                                      the policy is run.
### Example: _show_cmd                   show version
###
# check if all required environment variables exist
# If any required environment variable does not exist, print out an error msg and quit
if {![info exists _log_file]} {
    if {![info exists _email_server]} {
set result \
"Policy cannot be run: variable _log_file or _email_server has not been set"
error $result $errorInfo
    }
    if {![info exists _email_from]} {
set result \
"Policy cannot be run: variable _log_file or _email_from has not been set"
error $result $errorInfo
    }
    if {![info exists _email_to]} {
set result \
"Policy cannot be run: variable _log_file ore _email_to has not been set"
error $result $errorInfo
    }
    if {![info exists _email_cc]} {
#_email_cc is an option, must set to empty string if not set.
set _email_cc ""
    }
}
if {![info exists _show_cmd]} {
    set result \
        "Policy cannot be run: variable _show_cmd has not been set"
    error $result $errorInfo
}
namespace import ::cisco::eem::*
namespace import ::cisco::lib::*
```

```
# query the event info and log a message
array set arr_einfo [event_reqinfo]
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
global timer_type timer_time_sec
set timer_type $arr_einfo(timer_type)
set timer_time_sec $arr_einfo(timer_time_sec)
# log a message
set msg [format "timer event: timer type %s, time expired %s" \
        $timer_type [clock format $timer_time_sec]]
action_syslog priority info msg $msg
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
$_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
# 1. execute the command
if [catch {cli_open} result] {
    error $result $errorInfo
} else {
    array set cli1 $result
}

# save exact execution time for command
set time_now [clock seconds]
# execute command
if [catch {cli_exec $cli1(fd) $_show_cmd} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
    # format output: remove trailing router prompt
    regexp {\n*(.*\n)([^\n]*)$} $result dummy cmd_output
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
    error $result $errorInfo
}
# 2. log the success of the CLI command
set msg [format "Command \"%s\" executed successfully" $_show_cmd]
action_syslog priority info msg $msg
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
# 3. if _log_file is defined, then attach it to the file
if {[info exists _log_file]} {
    # attach output to file
    if [catch {open $_log_file a+} result] {
        error $result
    }
    set fileD $result
    # save timestamp of command execution
    #      (Format = 00:53:44 PDT Mon May 02 2005)
    set time_now [clock format $time_now -format "%T %Z %a %b %d %Y"]
    puts $fileD "%%% Timestamp = $time_now"
    puts $fileD $cmd_output
    close $fileD
}
# 4. if _email_server is defined send the email out
if {[info exists _email_server]} {
    set routername [info hostname]
    if {[string match "" $routername]} {
error "Host name is not configured"
    }
    if [catch {smtp_subst [file join $tcl_library email_template_cmd.tm]} \
    result] {
error $result $errorInfo
    }
    if [catch {smtp_send_email $result} result] {
error $result $errorInfo
```

```
    }
}
```

sl_intf_down.tcl サンプル ポリシー

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

```
::cisco::eem::event_register_syslog occurs 1 pattern $_syslog_pattern maxrun 90
#------------------------------------------------------------------
# EEM policy to monitor for a specified syslog message.
# Designed to be used for syslog interface-down messages.
# When event is triggered, the given config commands will be run.
#
# July 2005, Cisco EEM team
#
# Copyright (c) 2005 by cisco Systems, Inc.
# All rights reserved.
#------------------------------------------------------------------
### The following EEM environment variables are used:
###
### _syslog_pattern (mandatory)        - A regular expression pattern match string
###                                      that is used to compare syslog messages
###                                      to determine when policy runs
### Example: _syslog_pattern             .*UPDOWN.*FastEthernet0/0.*
###
### _email_server (mandatory)          - A Simple Mail Transfer Protocol (SMTP)
###                                      mail server used to send e-mail.
### Example: _email_server               mailserver.example.com
###
### _email_from (mandatory)            - The address from which e-mail is sent.
### Example: _email_from                 devtest@example.com
###
### _email_to (mandatory)              - The address to which e-mail is sent.
### Example: _email_to                   engineering@example.com
###
### _email_cc (optional)               - The address to which the e-mail must
###                                      be copied.
### Example: _email_cc                   manager@example.com
###
### _config_cmd1 (optional)            - The first configuration command that
###                                      is executed.
### Example: _config_cmd1                interface Ethernet1/0
###
### _config_cmd2 (optional)            - The second configuration command that
###                                      is executed.
### Example: _config_cmd2                no shutdown
###
# check if all the env variables we need exist
# If any of them doesn't exist, print out an error msg and quit
if {![info exists _email_server]} {
    set result \
        "Policy cannot be run: variable _email_server has not been set"
    error $result $errorInfo
}
if {![info exists _email_from]} {
    set result \
        "Policy cannot be run: variable _email_from has not been set"
    error $result $errorInfo
}
if {![info exists _email_to]} {
    set result \
        "Policy cannot be run: variable _email_to has not been set"
    error $result $errorInfo
}
if {![info exists _email_cc]} {
     #_email_cc is an option, must set to empty string if not set.
     set _email_cc ""
}
```

```
namespace import ::cisco::eem::*
namespace import ::cisco::lib::*
# 1. query the information of latest triggered eem event
array set arr_einfo [event_reqinfo]
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
      $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
set msg $arr_einfo(msg)
set config_cmds ""
# 2. execute the user-defined config commands
if [catch {cli_open} result] {
    error $result $errorInfo
} else {
    array set cli1 $result
}

if [catch {cli_exec $cli1(fd) "config t"} result] {
    error $result $errorInfo
}
if {[info exists _config_cmd1]} {
    if [catch {cli_exec $cli1(fd) $_config_cmd1} result] {
        error $result $errorInfo
    }
    append config_cmds $_config_cmd1
}
if {[info exists _config_cmd2]} {
    if [catch {cli_exec $cli1(fd) $_config_cmd2} result] {
        error $result $errorInfo
    }
    append config_cmds "\n"
    append config_cmds $_config_cmd2
}
if [catch {cli_exec $cli1(fd) "end"} result] {
    error $result $errorInfo
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
    error $result $errorInfo
}
after 60000
# 3. send the notification email
set routername [info hostname]
if {[string match "" $routername]} {
    error "Host name is not configured"
}
if [catch {smtp_subst [file join $tcl_library email_template_cfg.tm]} result] {
    error $result $errorInfo
}
if [catch {smtp_send_email $result} result] {
    error $result $errorInfo
}
```

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

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

Tcl set コマンド操作のトレース：例

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

```
rename set _set
proc set {var args} {
   puts [list setting $var $args]
   uplevel _set $var $args
};
When this is placed in a policy, a message is displayed anytime a scalar variable is set,
for example:

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

その他の関連資料

次の項では、Embedded Event Manager ポリシーの設定および管理についての関連資料を示します。

関連資料

関連項目	マニュアル タイトル
Embedded Event Manager コマンド	『*Cisco ASR 9000 Series Aggregation Services Router System Monitoring Command Reference*』の「*Embedded Event Manager Commands*」モジュール
ルート プロセッサ フェールオーバー コマンド	『*Cisco ASR 9000 Series Aggregation Services Router Interface and Hardware Component Command Reference*』の「Hardware Redundancy and Node Administration Commands」モジュール
Cisco IOS XR XML API に関する資料	『*Cisco IOS XR XML API Guide*』
Cisco IOS XR のスタートアップ マニュアル	『*Cisco ASR 9000 Series Aggregation Services Router Getting Started Guide*』
ユーザ グループとタスク ID に関する情報	『*Cisco ASR 9000 Series Aggregation Services Router System Security Configuration Guide*』の「*Configuring AAA Services*」の章

標準

標準	タイトル
この機能でサポートされる新規の標準または変更された標準はありません。また、既存の標準のサポートは変更されていません。	—

MIB

MIB	MIB のリンク
—	Cisco IOS XR ソフトウェアを使用している MIB を特定してダウンロードするには、次の URL にある Cisco MIB Locator を使用し、[Cisco Access Products] メニューからプラットフォームを選択します。http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFC

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

シスコのテクニカル サポート

説明	リンク
シスコのテクニカル サポート Web サイトでは、製品、テクノロジー、ソリューション、技術的なヒント、およびツールへのリンクなどの、数千ページに及ぶ技術情報が検索可能です。Cisco.com に登録済みのユーザは、このページから詳細情報にアクセスできます。	*http://www.cisco.com/cisco/web/support/index.html*

Embedded Event Manager ポリシー Tcl コマンド拡張リファレンス

ここでは、次の EEM ポリシーの Tcl コマンド拡張カテゴリについて説明します。

（注） すべての EEM Tcl コマンド拡張について、エラーがあった場合、戻される Tcl 結果文字列には、エラー情報が含まれます。

（注） 数値範囲が指定されていない引数は、–2147483648 から 2147483647 までの整数から取得されます。

次の表記法が、Tcl コマンド拡張ページで説明されている構文に使用されます。

- 任意の引数は、たとえば次の例のように、角カッコ内に示されます。

```
[type ?]
```

- 疑問符（?）は 入力する変数を表します。
- 引数間の選択肢は、たとえば次の例のように、パイプ文字で示されます。

```
[queue_priority low|normal|high]
```

Embedded Event Manager イベント登録 Tcl コマンド拡張

次の EEM イベント登録 Tcl コマンド拡張がサポートされています。

event_register_appl

アプリケーション イベントの登録を行います。 この Tcl コマンド拡張を使用すると、アプリケーション イベントがトリガーされ、続いて event_publish Tcl コマンド拡張の別のポリシーが実行されるときに、ポリシーが実行されます。event_publish コマンド拡張によって、アプリケーション イベントがパブリッシュされます。

アプリケーション イベントを登録するには、サブシステムを指定する必要があります。 Tcl ポリシーまたは内部 EEM API のいずれかによって、アプリケーション イベントをパブリッシュできます。イベントがポリシーによってパブリッシュされている場合、ポリシーで予約される *sub_system* 引数は 798 です。

構文

```
event_register_appl [sub_system ?] [type ?] [queue_priority low|normal|high] [maxrun ?]
[nice 0|1]
```

引数

sub_system	（任意）アプリケーションイベントをパブリッシュしたEEMポリシーに割り当てられる番号。他のすべての番号はCiscoで使用するために予約されており、番号は798に設定されます。この引数が指定されない場合、すべてのコンポーネントが照会されます。
type	（任意）指定されたイベント内のイベントサブタイプ。*sub_system*引数および*type*引数によって、アプリケーションイベントが一意に識別されます。この引数が指定されない場合、すべてのタイプが照会されます。この引数を指定する場合、1～4294967295の整数を選択する必要があります。 パブリッシュと登録を動作させるには、**event_publish**コマンド拡張と**event_register_appl**コマンド拡張との間のコンポーネントとタイプが一致する必要があります。
queue_priority	（任意）スクリプトのキューイングに使用されるプライオリティ レベル。normalは、lowプライオリティよりも高く、highプライオリティよりも低いプライオリティです。このプライオリティは実行プライオリティではなく、キューイング プライオリティです。 この引数が指定されない場合、デフォルトのプライオリティはnormal です。
maxrun	（任意）スクリプトの最大ランタイム（SSSSSSSSSS[.MMM] 形式で指定します。SSSSSSSSSSは、0～4294967295の秒数を表す整数で、MMMは0～999のミリ秒数を表す整数である必要があります）。この引数が指定されない場合、デフォルトの20秒ランタイム制限が使用されます。

nice	（任意）ポリシー実行時間のプライオリティ設定。 *nice* 引数が 1 に設定されている場合、ポリシーは、デフォルトプライオリティよりも低い実行時プライオリティで実行されます。デフォルト値は 0 です

複数の条件が存在する場合、すべての条件が満たされたときに、アプリケーション イベントが発生します。

結果文字列

なし

_cerrno を設定

No

event_register_cli

CLI イベントの登録を行います。 この Tcl コマンドを使用すると、拡張 CLI コマンドに対して実行されるパターン マッチに基づいて、特定パターンの CLI コマンドが入力されるときに、ポリシーが実行されます。 これは、IOS-XR の新しいプロセス（dlrsc_tracker）として実装されます。この ED は XR の admin コマンドでパターン マッチしません。

（注） **sh mem summary** などの省略形の CLI コマンドを入力でき、パーサーによってコマンドが **show memory summary** に拡張され、照会が実行されます。 CLI イベント ディテクタによる機能は、有効な XR CLI コマンドでの正規表現パターン比較機能だけです。 これには、リダイレクションが使用される場合のパイプ記号（|）以降のテキストは含まれません。

構文

```
event_register_cli [tag ?]
[occurs ?] [period ?] pattern ? [default ?] [queue_priority low|normal|high|last] [maxrun
?] [nice 0|1]
```

引数

tag	（任意）Tcl スクリプト内で複数のイベント文をサポートするため、Tcl コマンド拡張のトリガーとともに使用できるタグを指定する文字列。

occurs	（任意）イベントが発生する前の発生回数。この引数が指定されない場合、イベントは1回目から発生します。この引数が指定される場合は、1～4294967295の範囲の整数である必要があります。
period	（任意）イベントがパブリッシュされるようにするために、すべてのCLIイベントが発生する必要がある（occurs句を満たす必要がある）逆方向検索時間ウィンドウを指定します（SSSSSSSSSS[.MMM]形式で指定します。SSSSSSSSSSは、0～4294967295の秒数を表す整数で、MMMは0～999のミリ秒数を表す整数である必要があります）。この引数が指定されない場合は、最新のイベントが使用されます。
pattern	（必須）CLIコマンドパターンマッチの実行に使用される正規表現を指定します。
default	（任意）CLIイベントディテクタがポリシーの終了を待つ時間（SSSSSSSSSS[.MMM]形式で指定します。SSSSSSSSSSは、0～4294967295の秒数を表す整数で、MMMは0～999のミリ秒数を表す整数である必要があります）。ポリシーが終了する前にデフォルトの時間の期限が切れると、デフォルトアクションが実行されます。デフォルトアクションによって、コマンドが実行されます。この引数が指定されない場合、デフォルトの時間は30秒に設定されます。

複数の条件が存在する場合、すべての条件が一致したときに、CLIイベントが発生します。

結果文字列

なし

_cerrno を設定

No

event_register_config

実行コンフィギュレーションの変更の登録を行います。コンフィギュレーションが変更されるときに、ポリシーをトリガーするためにこのTclコマンド拡張を使用します。これは、IOS-XRの新

しいプロセス（dlrsc_tracker）として実装されます。 この ED は XR の admin 設定の変更を確認しません。

構文

```
event_register_config
[queue_priority low|normal|high|last]
[maxrun ?] [nice 0|1]
```

引数

queue_priority	（任意）次のような、スクリプトがキューに入れられるプライオリティ レベル。 • queue_priority low：3 つのプライオリティ レベルの最も低いレベルで、スクリプトがキューに入れられるよう、指定します。 • queue_priority normal：low プライオリティよりも高く、high プライオリティよりも低いプライオリティ レベルで、スクリプトがキューに入れられるよう、指定します。 • queue_priority high：3 つのプライオリティ レベルの最も高いレベルで、スクリプトがキューに入れられるよう、指定します。 • queue_priority last：最も低いプライオリティ レベルで、スクリプトがキューに入れられるよう、指定します。 「queue_priority_last」引数が設定された状態で複数のスクリプトが登録されている場合、これらのスクリプトは、イベントのパブリッシュ順に実行されます。 （注） queue_priority 引数によって、登録されているスクリプトの実行プライオリティではなく、キューイングのプライオリティが指定されます。 この引数が指定されない場合、デフォルトのキューイング プライオリティは normal です。
maxrun	（任意）スクリプトの最大ランタイム（SSSSSSSSSS[.MMM] 形式で指定します。SSSSSSSSSS は、0 ～ 4294967295 の秒数を表す整数で、MMM は 0 ～ 999 のミリ秒数を表す整数である必要があります）。 この引数が指定されない場合、デフォルトの 20 秒ランタイム制限が使用されます。

nice	（任意）ポリシー実行時間のプライオリティ設定。 nice 引数が 1 に設定されている場合、ポリシーは、デフォルトプライオリティよりも低い実行時プライオリティで実行されます。デフォルト値は 0 です

複数の条件が存在する場合、すべての条件が一致したときに、Syslog イベントが発生します。

結果文字列

なし

_cerrno を設定

No

event_register_counter

パブリッシャとサブスクライバの両方として、カウンタ イベントの登録を行います。 この Tcl コマンド拡張を使用すると、しきい値に近くなった名前付きカウンタに基づいて、ポリシーが実行されます。サブスクライバとして、このイベントカウンタによって、登録に必要なカウンタの名前が指定され、別のポリシーまたは別のプロセスに依存して、カウンタが実際に操作されます。たとえば、policyB がカウンタ ポリシーとして動作し、policyA（カウンタ ポリシーは不要ですが）では、register_counter、counter_modify、または unregister_counter の各 Tcl コマンド拡張を使用して、policyB で定義されているカウンタが操作されます。

構文

```
event_register_counter name ? entry_op gt|ge|eq|ne|lt|le entry_val ?
exit_op gt|ge|eq|ne|lt|le exit_val ? [queue_priority low|normal|high]
[maxrun ?] [nice 0|1]
```

引数

name	（必須）カウンタの名前。
entry_op	（必須）現在のカウンタの値を開始値と比較するために使用される開始比較演算子。真の場合、イベントが発生し、終了基準を満たすまでイベントモニタリングがディセーブルにされます。
entry_val	（必須）カウンタイベントを発生させる必要があるかどうかを判断するために、現在のカウンタの値を比較する必要がある値。

exit_op	（必須）現在のカウンタの値を終了値と比較するために使用される終了比較演算子。真の場合、このイベントのイベントモニタリングが再度イネーブルにされます。
exit_val	（必須）終了基準を満たすかどうかを判断するために、現在のカウンタの値を比較する必要がある値。
queue_priority	（任意）スクリプトのキューイングに使用されるプライオリティ レベル。normal は、low プライオリティよりも高く、high プライオリティよりも低いプライオリティです。このプライオリティは実行プライオリティではなく、キューイング プライオリティです。 この引数が指定されない場合、デフォルトのプライオリティは normal です。
maxrun	（任意）スクリプトの最大ランタイム（SSSSSSSSSS[.MMM] 形式で指定します。SSSSSSSSSS は、0 ～ 4294967295 の秒数を表す整数で、MMM は 0 ～ 999 のミリ秒数を表す整数である必要があります）。 この引数が指定されない場合、デフォルトの 20 秒ランタイム制限が使用されます。
nice	（任意）ポリシー実行時間のプライオリティ設定。 *nice* 引数が 1 に設定されている場合、ポリシーは、デフォルトプライオリティよりも低い実行時プライオリティで実行されます。デフォルト値は 0 です

結果文字列

なし

_cerrno を設定

No

event_register_hardware

ハードウェア イベントと条件で指定される環境モニタリング ハードウェア デバイス用に登録します。

構文

```
event_register_hardware env_device ? env_cond ?
[priority normal|low|high] [maxrun_sec ?] [maxrun_nsec ?] [nice 0|1]
```

引数

env_device	（必須）モニタリング用に使用する環境デバイス。1 ～ 2147483647 の範囲の整数であることが必要です。 これは、複数のタイプの環境デバイスをモニタするビットマスクです。 サポートされるデバイスとその対応するビットマスクの一覧を次に示します。 • 0x0001 シャーシ • 0x0002 バックプレーン • 0x0004 スロット • 0x0008 カード • 0x0010 ポート • 0x0020 ファン • 0x0040 電源のグループ • 0x0080 電源 • 0x0100 センサー 複数のデバイスをモニタするには、ビット単位の OR を行います。

env_cond	（必須）モニタする環境条件。これは、複数のタイプの環境条件をモニタするビットマスクです。サポートされる環境条件とその対応するビットマスクの一覧を次に示します。 • 0x0001 低警告 • 0x0002 高警告 • 0x0004 警告 • 0x0010 低クリティカル • 0x0020 高クリティカル • 0x0040 クリティカル • 0x0100 プレシャットダウン • 0x0200 シャットダウン
priority	（任意）スクリプトをキューに格納する優先順位レベル。指定しない場合、デフォルトでは通常の優先順位が使用されます。
maxrun_sec、maxrun_nsec	（任意）秒またはナノ秒単位で指定される最大実行時間。0 ～ 2147483647 の範囲の整数であることが必要です。指定しない場合、デフォルトの 20 秒の実行時間制限が使用されます。
nice	（任意）秒またはナノ秒単位で指定される最大実行時間。0 ～ 2147483647 の範囲の整数であることが必要です。指定しない場合、デフォルトの 20 秒の実行時間制限が使用されます。

結果文字列

なし

_cerrno を設定

No

event_register_none

event manager run コマンドによってトリガーされるイベントの登録を行います。これらのイベントは、このイベントをスクリーニングする None イベント ディテクタによって処理されます。

構文

```
event_register_none [queue_priority low|normal|high] [maxrun ?] [nice 0|1]
```

引数

queue_priority	（任意）スクリプトのキューイングに使用されるプライオリティ レベル。normal は、low プライオリティよりも高く、high プライオリティよりも低いプライオリティです。 このプライオリティは実行プライオリティではなく、キューイング プライオリティです。 この引数が指定されない場合、デフォルトのプライオリティは normal です。
maxrun	（任意）スクリプトの最大ランタイム（SSSSSSSSSS[.MMM] 形式で指定します。SSSSSSSSSS は、0 ～ 4294967295 の秒数を表す整数で、MMM は 0 ～ 999 のミリ秒数を表す整数である必要があります）。 この引数が指定されない場合、デフォルトの 20 秒ランタイム制限が使用されます。
nice	（任意）ポリシー実行時間のプライオリティ設定。 *nice* 引数が 1 に設定されている場合、ポリシーは、デフォルト プライオリティよりも低い実行時プライオリティで実行されます。デフォルト値は 0 です

結果文字列

なし

_cerrno を設定

No

event_register_oir

活性挿抜（OIR）イベントの登録を行います。 この Tcl コマンド拡張を使用すると、ハードウェア カード OIR イベントの発生時に発生するイベントに基づいて、ポリシーが実行されます。 これらのイベントは、このイベントをスクリーニングする OIR イベント ディテクタによって処理されます。

構文

```
event_register_oir [queue_priority low|normal|high] [maxrun ?] [nice 0|1]
```

引数

queue_priority	（任意）スクリプトのキューイングに使用されるプライオリティ レベル。normal は、low プライオリティよりも高く、high プライオリティよりも低いプライオリティです。このプライオリティは実行プライオリティではなく、キューイング プライオリティです。 この引数が指定されない場合、デフォルトのプライオリティは normal です。
maxrun	（任意）スクリプトの最大ランタイム（SSSSSSSSSS[.MMM] 形式で指定します。SSSSSSSSSS は、0 ～ 4294967295 の秒数を表す整数で、MMM は 0 ～ 999 のミリ秒数を表す整数である必要があります）。この引数が指定されない場合、デフォルトの 20 秒ランタイム制限が使用されます。
nice	（任意）ポリシー実行時間のプライオリティ設定。 *nice* 引数が 1 に設定されている場合、ポリシーは、デフォルト プライオリティよりも低い実行時プライオリティで実行されます。デフォルト値は 0 です

結果文字列

なし

_cerrno を設定

No

event_register_process

プロセス イベントの登録を行います。 この Tcl コマンド拡張を使用すると、Cisco IOS XR ソフトウェア モジュール方式プロセスの開始時と停止時に発生するイベントに基づいて、ポリシーが実行されます。 これらのイベントは、このイベントをスクリーニングする System Manager イベント ディテクタによって処理されます。 この Tcl コマンド拡張は、ソフトウェア モジュール方式イメージでのみサポートされます。

構文

```
event_register_process abort|term|start
[job_id ?] [instance ?] [path ?] [node ?]
[queue_priority low|normal|high] [maxrun ?] [nice 0|1] [tag?]
```

引数

abort	（必須）プロセスの異常な終了。ゼロではない終了ステータスでの終了、カーネル生成信号の受信、またはユーザ要求のために送信されない SIGTERM 信号または SIGKILL 信号の受信のため、プロセスが強制終了されることがあります。
term	（必須）プロセスの正常な終了。
start	（必須）プロセスの開始。
job_id	（任意）プロセスイベントをパブリッシュした EEM ポリシーに割り当てられる番号。 他のすべての番号は Cisco での使用のために予約されているため、番号は 798 に設定されます。
instance	（任意）プロセス インスタンス ID。 指定される場合、この引数は、1 ～ 4294967295 の範囲の整数である必要があります。
path	（任意）プロセスパス名（正規表現文字列）。
node	（任意）ノード名は、「node」という語句と、それに続く、次の形式を使用してスラッシュ（/）で区切られた 2 つのフィールドで構成される、文字列です。 node<slot-number>/<cpu-number> slot-number は、ハードウェア スロット番号です。 cpu-number は、ハードウェア CPU 番号です。 たとえば、スロット 0 にある Cisco Catalyst 6500 シリーズ スイッチのスーパーバイザ カードの SP CPU は、node0/0 と指定されます。 たとえば、スロット 0 にある Cisco Catalyst 6500 シリーズ スイッチのスーパーバイザ カードの RP CPU は、node0/1 と指定されます。 *node* 引数が指定されない場合、デフォルトのノード指定は、常に、すべての該当するノードを表す正規表現パターン マッチ * です。

queue_priority	（任意）スクリプトのキューイングに使用されるプライオリティ レベル。normal は、low プライオリティよりも高く、high プライオリティよりも低いプライオリティです。このプライオリティは実行プライオリティではなく、キューイング プライオリティです。 この引数が指定されない場合、デフォルトのプライオリティは normal です。
maxrun	（任意）スクリプトの最大ランタイム（SSSSSSSSSS[.MMM] 形式で指定します。SSSSSSSSSS は、0 ～ 4294967295 の秒数を表す整数で、MMM は 0 ～ 999 のミリ秒数を表す整数である必要があります）。 この引数が指定されない場合、デフォルトの 20 秒ランタイム制限が使用されます。
nice	（任意）ポリシー実行時間のプライオリティ設定。 *nice* 引数が 1 に設定されている場合、ポリシーは、デフォルトプライオリティよりも低い実行時プライオリティで実行されます。デフォルト値は 0 です
tag	タグは指定できますが無視されます。 タグ オプションを指定した Cisco IOS EEM スクリプトは、エラーになることなく Cisco IOS XR ソフトウェア環境で実行できます。 Cisco IOS XR ソフトウェアは複数のイベントをサポートしていないため、タグの効果はありません。

任意の引数が指定されない場合、イベントは、引数のすべての可能な値に対して照会されます。複数の引数が存在する場合、すべての条件が一致したときに、プロセス イベントが発生します。

結果文字列

なし

_cerrno を設定

No

event_register_snmp

簡易ネットワーク管理プロトコル（SNMP）統計イベントの登録を行います。 この Tcl コマンド拡張を使用すると、SNMP オブジェクト ID（OID）によって指定されたカウンタが、定義された

しきい値に近くなったときに、ポリシーが実行されます。 snmpポリシーが登録されると、ポーリング タイマーが指定されます。 イベントの一致は、登録されたイベントのポーリング タイマーが期限切れになるときに発生します。 SNMP 通知が Tcl ポリシーを使用して動作するようにするには、**snmp-server manager** CLI コマンドをイネーブルにする必要があります。

構文

```
event_register_snmp [tag ?] oid ? get_type exact|next
entry_op gt|ge|eq|ne|lt|le entry_val ?
entry_type value|increment|rate
[exit_comb or|and]
[exit_op gt|ge|eq|ne|lt|le] [exit_val ?]
[exit_type value|increment|rate]
[exit_time ?] poll_interval ? [average_factor ?]
[queue_priority low|normal|high|last]
[maxrun ?] [nice 0|1]
```

引数

tag	（任意）Tcl スクリプト内で複数のイベント文をサポートするため、Tcl コマンド拡張のトリガーとともに使用できるタグを指定する文字列。
entry_op	（必須）現在のOIDデータの値を開始値と比較するために使用される開始比較演算子。真の場合、イベントが発生し、終了基準を満たすまでイベントモニタリングがディセーブルにされます。
get_type	（必須）指定されたOIDに適用する必要があるSNMP 取得操作のタイプ。 get_type 引数が「exact」の場合、指定された OID の値が取得されます。get_type 引数が「next」の場合、指定されたOIDの辞書順での後続値が取得されます。
entry_val	（必須）SNMPイベントを発生させる必要があるかどうかを判断するために、現在のOIDデータの値を比較する必要がある値。

entry-type	entry-val 引数によって指定されたオブジェクト ID に適用される操作のタイプを指定します。 値は、entry-val 引数の実際の値として定義されます。 増分では、entry-val フィールドは増分差異として使用され、entry-val は、現在のカウンタの値と、イベントが最後に真であったとき（これが新しいイベントの場合は最初にポーリングされたサンプル）の値との間の差と、比較されます。負の値によって、減少しているカウンタの増分差異がチェックされます。 レートは、ある期間の変更の平均レートとして定義されます。期間は、average-factor の値に、poll-interval の値を乗じたものです。ポーリング間隔ごとに、現在のサンプルと前のサンプルとの間の差が取得され、絶対値として記録されます。前の average-factor 値サンプルの平均は、変更のレートとして取得されます。
exit_comb	（任意）イベントモニタリングが再度イネーブルにされるよう、終了基準が満たされているかどうかを判断するために必要な、終了条件テストの組み合わせを示す、終了組み合わせ演算子を使用します。「and」の場合は、終了基準を満たすために、終了値と終了時間テストの両方を渡す必要があります。「or」の場合は、終了基準を満たすために、終了値または終了時間テストのいずれかを渡します。 exit_comb が「and」の場合、exit_op と exit_val（exit_time）が存在する必要があります。 exit_comb が「or」の場合、（exit_op と exit_val）または（exit_time）が存在する必要があります。
exit_op	（任意）現在の OID データの値を終了値と比較するために使用される終了比較演算子。真の場合、このイベントのイベントモニタリングが再度イネーブルにされます。
exit_val	（任意）終了基準を満たすかどうかを判断するために、現在の OID データの値を比較する必要がある値。

exit-type	（任意）exit-val 引数によって指定されたオブジェクト ID に適用される操作のタイプを指定します。 指定されない場合、値が仮定されます。 値は、exit-val 引数の実際の値として定義されます。 増分では、exit-val フィールドは増分差異として使用され、exit-val は、現在のカウンタの値と、イベントが最後に真であったとき（これが新しいイベントの場合は最初にポーリングされたサンプル）の値との間の差と、比較されます。 負の値によって、減少しているカウンタの増分差異がチェックされます。 レートは、ある期間の変更の平均レートとして定義されます。 期間は、average-factor の値に、poll-interval の値を乗じたものです。 ポーリング間隔ごとに、現在のサンプルと前のサンプルとの間の差が取得され、絶対値として記録されます。 前の average-factor 値サンプルの平均は、変更のレートとして取得されます。
exit_time	（任意）イベントモニタリングが再度イネーブルにされるときに発生するイベントの後の、POSIX タイマー ユニットの数。SSSSSSSSSS[.MMM] 形式で指定します。SSSSSSSSSS は、0 ～ 4294967295 の秒数を表す整数である必要があります。 MMM はミリ秒を表し、0 ～ 999 の整数である必要があります。
poll_interval	（必須）POSIX タイマー ユニットの連続的なポーリング間の間隔。 間隔は、現在、最小で 1 秒に設定されます（SSSSSSSSSS[.MMM] 形式で指定します。SSSSSSSSSS は、0 ～ 4294967295 の秒数を表す整数で、MMM は 0 ～ 999 のミリ秒数を表す整数である必要があります）。
average-factor	（任意）レートベースの計算に使用される期間の計算に使用される 1 から 64 の範囲の数。average-factor の値は、poll-interval の値を乗じた値で、ミリ秒単位で導き出される期間です。 最少平均係数値は 1 です。

結果文字列

なし

_cerrno を設定

No

event_register_snmp_notification

簡易ネットワーク管理プロトコル（SNMP）通知トラップ イベントの登録を行います。 この Tcl コマンド拡張を使用すると、特定のインターフェイスまたはアドレスで、指定された SNMP オブジェクト ID（OID）で SNMP トラップが検出されるときに、ポリシーが実行されます。 SNMP 通知が Tcl ポリシーを使用して動作するようにするには、**snmp-server manager** CLI コマンドをイネーブルにする必要があります。

構文

```
event_register_snmp_notification [tag ?] oid ? oid_val ?
op {gt|ge|eq|ne|lt|le}
[src_ip_address ?]
[dest_ip_address ?]
[queue_priority {normal|low|high|last}]
[maxrun ?]
[nice {0|1}]
[default ?]
[direction {incoming|outgoing}]
[msg_op {drop|send}]
```

引数

tag	（任意）Tcl スクリプト内で複数のイベント文をサポートするため、Tcl コマンド拡張のトリガーとともに使用できるタグを指定する文字列。
oid	（必須）SNMP ドット付き表記でのデータ エレメントの OID 番号（たとえば、1.3.6.1.2.1.2.1.0）。 指定された OID がドット（.）で終わっている場合、ドットの前の OID 番号で始まっているすべての OID が、照会されます。 これは、XR の SNMP がサポートするすべての OID をサポートします。
oid_val	（必須）SNMP イベントを発生させる必要があるかどうかを判断するために、現在の OID データの値を比較する必要がある OID 値。

op	（必須）現在の OID データの値を、SNMP プロトコル データ ユニット（PDU）の OID データ値と比較するために使用される、比較演算子。真の場合、イベントが発生します。
src_ip_address	（任意）SNMP 通知トラップが発信される発信元 IP アドレス。 デフォルトは all です。すべての IP アドレスから SNMP 通知トラップを受信するよう、設定されます。 このオプションは、XR ではサポートされません。これは、src_ip_address が着信トラップ用であり、EEM XR でサポートされていないためです。
dest_ip_address	（任意）SNMP 通知トラップが送信される宛先 IP アドレス。 デフォルトは all です。すべての宛先 IP アドレスから SNMP トラップを受信するよう、設定されます。
default	（任意）SNMP 通知イベントディテクタがポリシーの終了を待つ、秒単位での時間を指定します。 時間は、sssssssss[.mmm] 形式で指定します。sssssssss は、0 ～ 4294967295 の秒数を表す整数で、mmm は 0 ～ 999 のミリ秒数を表す整数である必要があります。
direction	（任意）発着信 SNMP トラップまたは通知 PDU がフィルタリングする方向。デフォルト値は発信です。 着信の XR 方向はサポートされておらず、ユーザが方向を着信として提供する場合は、ポリシーの登録に失敗します。
msg_op	（任意）イベントが一度トリガーされると、SNMP PDU（廃棄または送信）で行われるアクション。 デフォルトは send です。 XR の msg_op のドロップはサポートされておらず、ユーザが msg_op をドロップとして提供する場合は、ポリシーの登録に失敗します。

結果文字列

なし

_cerrno を設定

No

event_register_stat

統計情報イベントの登録を行います。 この Tcl コマンド拡張を使用すると、特定の統計カウンタが定義されたしきい値を超えた場合にポリシーを実行できます。

EEM キーワードが監視する統計カウンタを一意に識別するために、次の 3 つのフィールドがあります。

- 引数名に対応するデータ要素名。 たとえば、ifstats-generic 名がインターフェイス汎用統計として定義されています。
- データ要素の最初の修飾子は、*modifier_1* 引数に対応します。 たとえば、Ethernet1_0 は ifstats-generic の最初の修飾子として定義されており、インターフェイス汎用統計情報がイーサネット インターフェイス固有であることを修飾します。
- データ要素の 2 番目の修飾子は、*modifier_2* 引数に対応します。 たとえば、input-ptks は、ifstats-generic の 2 番目の修飾子として定義されており、特定のイーサネット インターフェイスのインターフェイス統計情報を、受信パケットの数としてさらに修飾します。

構文

```
event_register_stat name ? [modifier_1 ?] [modifier_2 ?]

entry_op gt|ge|eq|ne|lt|le entry_val ? [exit_comb or|and]

[exit_op gt|ge|eq|ne|lt|le] [exit_val ?] [exit_time_sec ?] [exit_time_nsec ?]
[poll_interval_sec ?] [poll_interval_nsec ?] [priority normal|low|high]
[maxrun_sec ?] [maxrun_nsec ?] [nice 0|1] [tag ?]
```

引数

name	（必須）統計データ要素名。
modifier_1	インターフェイス統計情報では必須ですが、それ以外ではオプションです。インターフェイス統計情報の場合、この変数はインターフェイス名です。インターフェイス名を取得するには、**show interface brief** コマンドを使用します。 このコマンドは、スラッシュ（/）で指定された、現在設定されているすべてのインターフェイス名の一覧を表示します（たとえば Ethernet 1/0）。 このインターフェイスを *modifier_1* 引数に対して設定するには、スラッシュをアンダースコアに変更します。

modifier_2	インターフェイス統計情報では必須ですが、それ以外ではオプションです。インターフェイス統計情報の場合、この変数はインターフェイス統計情報名です。インターフェイス統計情報名を取得するには、**show event manager statistics -table** コマンドを **all** キーワードとともに使用して、すべての統計情報のクラスを表示します。その後、**show event manager statistics -table** コマンドを *name* 引数とともに使用して、modifier_2 の特定の統計情報名を取得します。
entry_op	（必須）現在の統計値をエントリ値と比較するために使用するエントリ比較演算子。 true の場合イベントが発生し、終了条件が満たされるまでイベント監視が無効になります。
entry_val	（必須）統計情報イベントを発生させるかどうかを判断するために、現在の統計情報カウンタの値を比較する値。
exit_comb	（必須）イベント監視を再度イネーブルにできるように、終了条件が満たされているかどうかを判断するために必要な、終了条件テストの組み合わせを示す終了組み合わせ演算子。 条件が満たされている場合、終了条件を満たすためには、終了値テストと終了時間テストに合格する必要があります。または、終了値テストまたは終了時間テストのいずれかが終了条件を満たします。 *exit_comb* および *exit_op*、*exit_val* 引数（*exit_time_sec* 引数または *exit_time_nsec* 引数）が存在する必要があります。 *exit_comb* 引数または（*exit_op* および *exit_val* 引数）または（*exit_time_sec* 引数または *exit_time_nsec* 引数）が存在する必要があります。
exit_op	現在の統計値を終了値と比較するために使用する終了比較演算子。 true の場合、このイベントのイベント監視が再度イネーブルになります。
exit_val	終了条件が満たされているかどうかを判断するために、現在の統計情報カウンタの値を比較する値。

exit_time_sec exit_time_nse	イベント監視を再度イネーブルにする場合に、イベントが発生してからの POSIX タイマー ユニットの数。 整数は、0 ～ 2147483647 の範囲にしてください。
poll_interval_sec poll_interval_nsec	引数 *poll_interval_sec* と *poll_interval_nsec* のいずれかを指定する必要があります。 間隔は、POSIX 時間単位での連続するポーリングの間にある必要があります。 現在 1 秒以上に制限されています。 整数は、0 ～ 2147483647 の範囲にしてください。
priority	（任意）スクリプトに対してキューに格納される優先度。 指定しない場合、デフォルトでは通常の優先順位が使用されます。
maxrun_sec、 maxrun_nsec	（任意）秒またはナノ秒単位で指定される最大実行時間。 指定しない場合、デフォルトとして 20 秒のランタイム制限が使用されます。 整数は、0 ～ 2147483647 の範囲にしてください。
nice	（任意）*nice* 引数が値 1 に設定されている場合、ポリシーは、デフォルト プライオリティよりも低い実行時プライオリティで実行されます。 デフォルト値は 0 です
tag	タグは指定できますが無視されます。タグ オプションを指定した Cisco IOS EEM スクリプトは、エラーになることなく Cisco IOS XR ソフトウェア環境で実行できます。 Cisco IOS XR ソフトウェアは複数のイベントをサポートしていないため、タグの効果はありません。

（注） 終了条件には、時間ベース、値ベース、または両方を指定できます。 終了条件が満たされるまでイベント監視は再度イネーブルになりません。

複数の条件が存在する場合、すべての条件が満たされたときに、統計情報イベントが発生します。

結果文字列

なし

_cerrno を設定

No

event_register_syslog

Syslog イベントの登録を行います。 この Tcl コマンド拡張を使用すると、一定の時間内に一定回数の発生後、特定パターンの Syslog メッセージが記録されるときに、ポリシーがトリガーされます。

構文

```
event_register_syslog [occurs ?] [period ?] pattern ?
[priority all|emergencies|alerts|critical|errors|warnings|notifications|
informational|debugging|0|1|2|3|4|5|6|7]
[queue_priority low|normal|high]
[severity_fatal] [severity_critical] [severity_major]
[severity_minor] [severity_warning] [severity_notification]
[severity_normal] [severity_debugging]
[maxrun ?] [nice 0|1]
```

引数

occurs	（任意）イベントが発生する前の発生回数。この引数が指定されない場合、イベントは 1 回目から発生します。 指定される場合、0 より大きい値を指定する必要があります。
period	（任意）イベントを発生させるために取る必要がある 1 つまたは複数のイベントの間の、秒単位およびミリ秒単位の時間の間隔（SSSSSSSSSS[.MMM] 形式で指定します。SSSSSSSSSS は、0 ～ 4294967295 の秒数を表す整数で、MMM は 0 ～ 999 のミリ秒数を表す整数である必要があります）。 この引数が指定されない場合、期間チェックは適用されません。
pattern	（必須）Syslog メッセージ パターン マッチの実行に使用される正規表現。 この引数は、記録された Syslog メッセージを指定するためにポリシーによって使用されます。
priority	（任意）スクリーニングされるメッセージのプライオリティ。 この引数が指定される場合、指定されたロギング プライオリティ レベルまたはそれ以下メッセージのみがスクリーニングされます。 この引数が指定されない場合、デフォルトのプライオリティは 0 です。

queue_priority	（任意）スクリプトのキューイングに使用されるプライオリティ レベル。normal は、low プライオリティよりも高く、high プライオリティよりも低いプライオリティです。このプライオリティは実行プライオリティではなく、キューイング プライオリティです。 この引数が指定されない場合、デフォルトのプライオリティは normal です。
maxrun	（任意）スクリプトの最大ランタイム（SSSSSSSSSS[.MMM] 形式で指定します。SSSSSSSSSS は、0 ～ 4294967295 の秒数を表す整数で、MMM は 0 ～ 999 のミリ秒数を表す整数である必要があります）。 この引数が指定されない場合、デフォルトの 20 秒ランタイム制限が使用されます。
nice	（任意）ポリシー実行時間のプライオリティ設定。 *nice* 引数が 1 に設定されている場合、ポリシーは、デフォルトプライオリティよりも低い実行時プライオリティで実行されます。デフォルト値は 0 です

複数の条件が存在する場合、すべての条件が一致したときに、Syslog イベントが発生します。

表 *17* : *Syslog* イベントの重大度のマッピング

重大度のキーワード	Syslog のプライオリティ	説明
severity_fatal	LOG_EMERG (0)	システムが使用不可能な状態。
severity_critical	LOG_ALERT (1)	クリティカル条件で、即時対応が必要であることを示す
severity_major	LOG_CRIT (2)	重大な状態。
severity_minor	LOG_ERR (3)	軽微な状態。
severity_warning	LOG_WARNING (4)	警告状態。
severity_notification	LOG_NOTICE (5)	基本的な通知、情報メッセージ
severity_normal	LOG_INFO (6)	正常なイベント、正常な状態に戻ったことを伝える

重大度のキーワード	Syslog のプライオリティ	説明
severity_debugging	LOG_DEBUG (7)	デバッグ メッセージ。

結果文字列

なし

_cerrno を設定

No

event_register_timer

パブリッシャとサブスクライバの両方として、タイマーを作成し、タイマー イベントの登録を行います。 時間特有または時間に基づいたポリシーをトリガーする必要があるときに、この Tcl コマンド拡張を使用します。このイベントタイマーは、イベントのパブリッシャとサブスクライバの両方です。パブリッシャの部分は、名前付きタイマーがオフになるという条件を示します。サブスクライバの部分は、イベントが登録されているタイマーの名前を示します。

（注） CRON および絶対時間の指定は、現地時間で動作します。

構文

```
event_register_timer watchdog|countdown|absolute|cron
[name ?] [cron_entry ?]
[time ?]
[queue_priority low|normal|high] [maxrun ?]
[nice 0|1]
```

引数

watchdog	（必須）ウォッチドッグ タイマー。
countdown	（必須）カウントダウン タイマー。
absolute	（必須）絶対タイマー。
cron	（必須）CRON タイマー。
name	（任意）タイマーの名前。

cron_entry	

（任意）CRON タイマー タイプが指定される場合に、エントリを指定する必要があります。他のいずれかのタイマー タイプが指定される場合には、指定しないでください。 cron_entry は、UNIX CRON デーモンで使用される部分的な UNIX Crontab エントリ（最初の 5 つのフィールド）です。

cron_entry の指定は、5 つのフィールドが使用されるテキスト文字列で構成されます。フィールドは、空白文字で区切られます。 フィールドは、CRON タイマー イベントがトリガーされるときの時刻と日付を表します。フィールドは、表 18：CRON イベントがトリガーされるときの時刻と日付，（151 ページ） で説明されています。

番号の範囲を使用できます。範囲は、ハイフンで区切られる 2 つの数字で表示されます。範囲には、2 つの数字自身も含まれます。 たとえば、時刻に入力される 8-11 は、8 時、9 時、10 時、および 11 時での実行を示します。

フィールドはアスタリスク記号（*）も使用でき、これは常に「first-last」を表します。

リストを使用できます。リストは、カンマで区切られた番号のセット（または範囲）です。例："1,2,5,9" および "0-4,8-12"。

手順の値は、範囲の組み合わせで使用できます。 範囲に続く「/<number>」によって、範囲内での省略値を指定します。 たとえば、2 時間ごとにイベントのトリガーを指定する場合、「0-23/2」を hour フィールドに使用します。アスタリスク記号後にも手順を使用でき、「2 時間ごと」と指定する場合は、「*/2」を使用します。

month フィールドと day of week フィールドには、名前も使用できます。特定の日または月の最初の 3 文字を使用します（ケースは問題ではありません）。名前の範囲またはリストは使用できません。

タイマー イベントがトリガーされる日は、day of month と day of week の 2 つのフィールドで指定できます。 両方のフィールドが制限される（つまり * ではない）場合、いずれかのフィー

	ルドが現在の時刻と一致すると、イベントがトリガーされます。 たとえば、「30 4 1,15 * 5」の場合、各月の 1 日と 15 日に加え、金曜日の午前 4:30 にイベントがトリガーされます。 最初の 5 つのフィールドの代わりに、7 つの特殊文字列の 1 つが表示されることがあります。これらの 7 つの特殊文字列は、表 19：cron_entry の特殊文字列, （152 ページ） で説明します。 例 1：「0 0 1,15 * 1」では、各月の 1 日と 15 日、および月曜日ごとに、真夜中の 0 時に、イベントがトリガーされます。 1 つのフィールドによってのみ日を指定する場合、他のフィールドは * に設定する必要があります。「0 0 * * 1」では、月曜日にのみ、真夜中の 0 時に、イベントがトリガーされます。 例 2：「15 16 1 * *」では、各月の 1 日の午後 4:15 にイベントがトリガーされます。 例 3：「0 12 * * 1-5」では、各週の月曜日から金曜日まで、正午に、イベントがトリガーされます。 例 4：「@weekly」では、1 週間に一度、日曜日の真夜中の 0 時に、イベントがトリガーされます。
time	（任意）CRON 以外のタイマータイプが指定される場合に、時刻を指定する必要があります。CRON タイマータイプが指定される場合には、指定しないでください。 ウォッチドッグ タイマーとカウントダウン タイマーでは、タイマーの期限が切れるまでの秒およびミリ秒の単位での数です。絶対タイマーでは、期限切れ時刻のカレンダー時間です。 時間は、SSSSSSSSSS[.MMM] 形式で指定します。SSSSSSSSSS は、0 ～ 4294967295 の秒数を表す整数で、MMM は 0 ～ 999 のミリ秒数を表す整数である必要があります。 期限の絶対日付は、1970 年 1 月 1 日以降の秒およびミリ秒の単位での数です。 指定された日付がすでに過ぎた場合、タイマーの期限はただちに切れます。

queue_priority	（任意）スクリプトのキューイングに使用されるプライオリティ レベル。normal は、low プライオリティよりも高く、high プライオリティよりも低いプライオリティです。 このプライオリティは実行プライオリティではなく、キューイング プライオリティです。 この引数が指定されない場合、デフォルトのプライオリティは normal です。
maxrun	（任意）スクリプトの最大ランタイム（SSSSSSSSSS[.MMM] 形式で指定します。SSSSSSSSSS は、0 ～ 4294967295 の秒数を表す整数で、MMM は 0 ～ 999 のミリ秒数を表す整数である必要があります）。 この引数が指定されない場合、デフォルトの 20 秒ランタイム制限が使用されます。
nice	（任意）ポリシー実行時間のプライオリティ設定。 *nice* 引数が 1 に設定されている場合、ポリシーは、デフォルト プライオリティよりも低い実行時プライオリティで実行されます。 デフォルト値は 0 です

表 *18*：*CRON* イベントがトリガーされるときの時刻と日付

フィールド	使用可能な値
minute	0 ～ 59
hour	0 ～ 23
day of month	1 ～ 31
month	1 ～ 12（または名前、表 19：cron_entry の特殊文字列, （152 ページ）を参照）
day of week	0 ～ 7（0 または 7 が日曜日、または名前。表 19：cron_entry の特殊文字列, （152 ページ） を参照）

表 *19* : *cron_entry* の特殊文字列

文字列	意味
@yearly	1 年に 1 回トリガーする、「0 0 1 1 *」。
@annually	@yearly と同じ。
@monthly	1 か月に 1 回トリガーする、「0 0 1 * *」。
@weekly	1 週間に 1 回トリガーする、「0 0 * * 0」。
@daily	1 日に 1 回トリガーする、「0 0 * * *」。
@midnight	@daily と同じ。
@hourly	1 時間に 1 回トリガーする、「0 * * * *」。

結果文字列

なし

_cerrno を設定

No

関連項目

event_register_timer_subscriber, （152 ページ）

event_register_timer_subscriber

サブスクライバとしてタイマー イベントの登録を行います。 この Tcl コマンド拡張を使用すると、サブスクライバとして、登録するイベント タイマーの名前が指定されます。 イベント タイマーは、別のポリシーまたは別のプロセスに依存して、カウンタが実際に操作されます。 たとえば、policyB はタイマー加入者ポリシーとして動作しますが、policyA（タイマー ポリシーは不要ですが）では、register_counter、timer_arm、または timer_cancel の各 Tcl コマンド拡張を使用して、policyB で参照されているカウンタが操作されます。

構文

```
event_register_timer_subscriber watchdog|countdown|absolute|cron
name ? [queue_priority low|normal|high] [maxrun ?] [nice 0|1]
```

引数

watchdog	（必須）ウォッチドッグ タイマー。

countdown	（必須）カウントダウン タイマー。
absolute	（必須）絶対タイマー。
cron	（必須）CRON タイマー。
name	（必須）タイマーの名前。
queue_priority	（任意）スクリプトのキューイングに使用されるプライオリティ レベル。normal は、low プライオリティよりも高く、high プライオリティよりも低いプライオリティです。 このプライオリティは実行プライオリティではなく、キューイング プライオリティです。 この引数が指定されない場合、デフォルトのプライオリティは normal です。
maxrun	（任意）スクリプトの最大ランタイム（SSSSSSSSSS[.MMM] 形式で指定します。SSSSSSSSSS は、0 ～ 4294967295 の秒数を表す整数で、MMM は 0 ～ 999 のミリ秒数を表す整数である必要があります）。 この引数が指定されない場合、デフォルトの 20 秒ランタイム制限が使用されます。
nice	（任意）ポリシー実行時間のプライオリティ設定。 nice 引数が 1 に設定されている場合、ポリシーは、デフォルトプライオリティよりも低い実行時プライオリティで実行されます。デフォルト値は 0 です

（注） タイマーイベントまたはカウンタイベントの登録を行う EEM ポリシーは、パブリッシャとサブスクライバの両方として動作できます。

結果文字列

なし

_cerrno を設定

No

関連項目

event_register_timer, （147 ページ）

event_register_track

XR のオブジェクト トラッキング コンポーネントからのレポート イベントに登録します。 この Tcl コマンド拡張を使用すると、指定された追跡に対するオブジェクト トラッキング コンポーネント レポートに基づいて、ポリシーがトリガーされます。 これは、IOS-XR の新しいプロセス（dlrsc_tracker）として実装されます。 追跡 ED が機能するためには、管理パッケージがインストールされている必要があることに注意してください。

構文

```
event_register_track ? [tag ?] [state up|down|any] [queue_priority low|normal|high|last]
[maxrun ?]
[nice 0|1]
```

引数

? （文字列を表す）	（必須）トラッキング対象オブジェクトの名前。
tag	（任意）Tcl スクリプト内で複数のイベント文をサポートするため、Tcl コマンド拡張のトリガーとともに使用できるタグを指定する文字列。
state	（任意）トラックされるオブジェクトの状態遷移によってイベントが発生するよう、指定します。 **up** が指定される場合、トラックされるオブジェクトが down 状態から up 状態に遷移するときに、イベントが発生します。 **down** が指定される場合、トラックされるオブジェクトが up 状態から down 状態に遷移するときに、イベントが発生します。 **any** が指定される場合、トラックされるオブジェクトが任意の状態へ、または任意の状態から遷移するときに、イベントが発生します。

queue_priority	（任意）次のような、スクリプトがキューに入れられるプライオリティ レベル。 • queue_priority low：3 つのプライオリティ レベルの最も低いレベルで、スクリプトがキューに入れられるよう、指定します。 • queue_priority normal：low プライオリティよりも高く、high プライオリティよりも低いプライオリティ レベルで、スクリプトがキューに入れられるよう、指定します。 • queue_priority high：3 つのプライオリティ レベルの最も高いレベルで、スクリプトがキューに入れられるよう、指定します。 • queue_priority last：最も低いプライオリティ レベルで、スクリプトがキューに入れられるよう、指定します。 「queue_priority_last」引数が設定された状態で複数のスクリプトが登録されている場合、これらのスクリプトは、イベントのパブリッシュ順に実行されます。 （注） queue_priority 引数によって、登録されているスクリプトの実行プライオリティではなく、キューイングのプライオリティが指定されます。 この引数が指定されない場合、デフォルトのキューイング プライオリティは normal です。
maxrun	（任意）スクリプトの最大ランタイム（SSSSSSSSSS[.MMM] 形式で指定します。SSSSSSSSSS は、0 ～ 4294967295 の秒数を表す整数で、MMM は 0 ～ 999 のミリ秒数を表す整数である必要があります）。 この引数が指定されない場合、デフォルトの 20 秒ランタイム制限が使用されます。
nice	（任意）ポリシー実行時間のプライオリティ設定。 nice 引数が 1 に設定されている場合、ポリシーは、デフォルト プライオリティよりも低い実行時プライオリティで実行されます。デフォルト値は 0 です

任意の引数が指定されない場合、イベントは、引数のすべての可能な値に対して照会されます。

結果文字列

なし

_cerrno を設定

No

event_register_wdsysmon

Watchdog System Monitor イベントの登録を行います。 この Tcl コマンド拡張を使用すると、いくつかのサブイベントまたは条件の組み合わせである複合イベントの登録が行われます。 たとえば、**event_register_wdsysmon** コマンドを使用して、特定処理の CPU の使用率が 80% を超え、かつ、処理に使用されるメモリが、その初期割り当ての 50% よりも大きいときの、条件の組み合わせの登録を行うことができます。 この Tcl コマンド拡張は、ソフトウェア モジュール方式イメージでのみサポートされます。

構文

```
event_register_wdsysmon [timewin ?]
[sub12_op and|or|andnot]
[sub23_op and|or|andnot]
[sub34_op and|or|andnot]
[sub1 subevent-description]
[sub2 subevent-description]
[sub3 subevent-description]
[sub4 subevent-description] [node ?]
[queue_priority low|normal|high]
[maxrun ?] [nice 0|1]
```

引数

timewin	（任意）イベントが生成されるようにするために、すべてのサブイベントが発生する必要がある時間ウィンドウを SSSSSSSSSS[.MMM] 形式で指定します。 SSSSSSSSSS 形式は、0 ～ 4294967295 の秒数を表す整数である必要があります。 MMM 形式は、0 ～ 999 のミリ秒数を表す整数である必要があります。
sub12_op	（任意）サブイベント 1 とサブイベント 2 とを比較する組み合わせ演算子。
sub34_op	（任意）サブイベント 1、2、サブイベント 3、サブイベント 4 とを比較する組み合わせ演算子。
sub1	（任意）サブイベント 1 を指定します。

subevent-description	（任意）サブイベントの構文。
sub2	（任意）サブイベント 2 を指定します。
sub3	（任意）サブイベント 3 を指定します。
sub4	（任意）サブイベント 4 を指定します。
node	（任意）デッドロック条件がモニタされるノード名は、「node」という語句と、それに続く、次の形式を使用してスラッシュ（/）で区切られた 2 つのフィールドで構成される、文字列です。 `node<slot-number>/<cpu-number>` slot-number は、ハードウェア スロット番号です。 cpu-number は、ハードウェア CPU 番号です。 たとえば、スロット 0 にある Cisco Catalyst 6500 シリーズ スイッチのスーパーバイザ カードの SP CPU は、node0/0 と指定されます。 たとえば、スロット 0 にある Cisco Catalyst 6500 シリーズ スイッチのスーパーバイザ カードの RP CPU は、node0/1 と指定されます。 node 引数が指定されない場合、デフォルトのノード指定は、登録が行われているローカル ノードです。
queue_priority	（任意）スクリプトのキューイングに使用されるプライオリティ レベル。normal は、low プライオリティよりも高く、high プライオリティよりも低いプライオリティです。 このプライオリティは実行プライオリティではなく、キューイング プライオリティです。 この引数が指定されない場合、デフォルトのプライオリティは normal です。
maxrun	（オプション）SSSSSSSSSS[.MMM] 形式で指定される最大実行時間。 SSSSSSSSSS 形式は、0 ～ 4294967295 の秒数を表す整数である必要があります。 MMM 形式は、0 ～ 999 のミリ秒数を表す整数である必要があります。 この引数が指定されない場合、デフォルトの 20 秒ランタイム制限が使用されます。

nice	（任意）ポリシー実行時間のプライオリティ設定。 *nice* 引数が 1 に設定されている場合、ポリシーは、デフォルトプライオリティよりも低い実行時プライオリティで実行されます。デフォルト値は 0 です

サブイベント

subevent description の構文は、7 つのケースのうちの 1 つを使用できます。

subevent descriptions の引数では、number 引数の値に次の制約事項が適用されます。

- dispatch_mgr では、val は、0 ～ 4294967295 の範囲の整数である必要があります。
- cpu_proc および cpu_tot では、val は、0 ～ 100 の整数である必要があります。
- mem_proc、mem_tot_avail、および mem_tot_used では、is_percent が偽の場合、val は、0 ～ 4294967295 の範囲の整数である必要があります。

1 deadlock procname ?

引数

procname	（必須）デッドロック条件をモニタするプロセス名を指定する正規表現。指定された場合、サブイベントによって、時間ウィンドウは無視されます。

1 dispatch_mgr [procname ?] [op gt|ge|eq|ne|lt|le] [val ?] [period ?]

引数

procname	（任意）dispatch_manager ステータスをモニタするプロセス名を指定する正規表現。
op	（任意）収集したイベント数を指定した値と比較するために使用する比較演算子。真の場合、イベントが発生します。
val	（任意）発生したイベントの数を比較する値。

period	（任意）発生したイベント数の時間（SSSSSSSSSS[.MMM] 形式で指定します。SSSSSSSSSS 形式は、0 ～ 4294967295 の秒数を表す整数である必要があります。 MMM 形式は、0 ～ 999 のミリ秒数を表す整数である必要があります。 この引数が指定されない場合は、最新のサンプルが使用されます。

1 cpu_proc [procname ?] [op gt|ge|eq|ne|lt|le] [val ?] [period ?]

引数

procname	（任意）CPU の使用条件をモニタするプロセス名を指定する正規表現。
op	（任意）収集した CPU 使用率サンプル パーセンテージを指定したパーセンテージ値と比較するために使用する比較演算子。 真の場合、イベントが発生します。
val	（任意）サンプル期間中の平均 CPU 使用率を比較するパーセンテージ値。
period	（任意）サンプルの収集を平均するための時間を SSSSSSSSSS[.MMM] 形式で指定します。SSSSSSSSSS 形式は、0 ～ 4294967295 の秒数を表す整数である必要があります。 MMM 形式は、0 ～ 999 のミリ秒数を表す整数である必要があります。 この引数が指定されない場合は、最新のサンプルが使用されます。

1 cpu_tot [op gt|ge|eq|ne|lt|le] [val ?] [period ?]

引数

op	（任意）収集した合計システム CPU 使用率サンプル パーセンテージを指定したパーセンテージ値と比較するために使用する比較演算子。 真の場合、イベントが発生します。
val	（任意）サンプル期間中の平均 CPU 使用率を比較するパーセンテージ値。

period	（任意）サンプルの収集を平均するための時間を SSSSSSSSSS[.MMM] 形式で指定します。 SSSSSSSSSS 形式は、0 ～ 4294967295 の秒数を表す整数である必要があります。 MMM 形式は、0 ～ 999 のミリ秒数を表す整数である必要があります。 この引数が指定されない場合は、最新のサンプルが使用されます。

1 mem_proc [procname ?] [op gt|ge|eq|ne|lt|le] [val ?] [is_percent TRUE|FALSE] [period ?]

引数

procname	（任意）メモリ使用状況をモニタするプロセス名を指定する正規表現。
op	（任意）収集した使用メモリを指定した値と比較するために使用する比較演算子。 真の場合、イベントが発生します。
val	（任意）キロバイト単位で指定される絶対値、またはパーセンテージ。 パーセンテージは、指定された時間内で最も古いサンプルと、最新のサンプルとの違いを表します。 メモリ使用量が時間内で 150 KB から 300 KB に増えた場合、増加パーセンテージは 100 です。 この値と測定された値が比較されます。
is_percent	（任意）TRUE に設定されている場合、パーセンテージの値が収集され、比較されます。 これ以外の場合、絶対値が収集され、比較されます。
period	（任意）is_percent が TRUE に設定される場合、時間のパーセンテージが計算されます。 これ以外の場合、収集サンプルの期間は平均化され、SSSSSSSSSS[.MMM] 形式で指定されます。 SSSSSSSSSS 形式は、0 ～ 4294967295 の秒数を表す整数である必要があります。 MMM 形式は、0 ～ 999 のミリ秒数を表す整数である必要があります。 この引数が指定されない場合は、最新のサンプルが使用されます。

1 mem_tot_avail [op gt|ge|eq|ne|lt|le] [val ?] [is_percent TRUE|FALSE] [period ?]

引数

op	（任意）収集した使用可能メモリを指定した値と比較するために使用する比較演算子。真の場合、イベントが発生します。
val	（任意）キロバイト単位で指定される絶対値、またはパーセンテージ。パーセンテージは、指定された時間内で最も古いサンプルと、最新のサンプルとの違いを表します。使用可能なメモリ使用量が時間内で 300 KB から 150 KB に減った場合、減少パーセンテージは 50 です。 この値と測定された値が比較されます。
is_percent	（任意）TRUE に設定されている場合、パーセンテージの値が収集され、比較されます。 これ以外の場合、絶対値が収集され、比較されます。
period	（任意）is_percent が TRUE に設定される場合、時間のパーセンテージが計算されます。 これ以外の場合、収集サンプルの期間は平均化され、SSSSSSSSSS[.MMM] 形式で指定されます。SSSSSSSSSS 形式は、0 ～ 4294967295 の秒数を表す整数である必要があります。 MMM 形式は、0 ～ 999 のミリ秒数を表す整数である必要があります。 この引数が指定されない場合は、最新のサンプルが使用されます。

1 mem_tot_used [op gt|ge|eq|ne|lt|le] [val ?] [is_percent TRUE|FALSE] [period ?]

引数

op	（任意）収集した使用メモリを指定した値と比較するために使用する比較演算子。真の場合、イベントが発生します。
val	（任意）キロバイト単位で指定される絶対値、またはパーセンテージ。パーセンテージは、指定された時間内で最も古いサンプルと、最新のサンプルとの違いを表します。メモリ使用量が時間内で 150 KB から 300 KB に増えた場合、増加パーセンテージは 100 です。 この値と測定された値が比較されます。

is_percent	（任意）TRUE に設定されている場合、パーセンテージの値が収集され、比較されます。 これ以外の場合、絶対値が収集され、比較されます。
period	（任意）is_percent が TRUE に設定される場合、時間のパーセンテージが計算されます。 これ以外の場合、収集サンプルの期間は平均化され、SSSSSSSSSS[.MMM] 形式で指定されます。 SSSSSSSSSS 形式は、0 ～ 4294967295 の秒数を表す整数である必要があります。 MMM 形式は、0 ～ 999 のミリ秒数を表す整数である必要があります。 この引数が指定されない場合は、最新のサンプルが使用されます。 （注） is_percent が真に設定されている場合、この引数は必須です。これ以外の場合、この引数は任意です。

結果文字列

なし

_cerrno を設定

No

（注） サブイベントの説明内部では、各引数は、位置に依存しません。

Embedded Event Manager イベント情報 Tcl コマンド拡張

次の EEM イベント情報 Tcl コマンド拡張がサポートされています。

event_reqinfo

現在のポリシーを実行させる原因のイベントについての情報を問い合わせます。

構文

```
event_reqinfo
```

引数

なし

結果文字列

ポリシーが正常に実行される場合、ポリシーをトリガーするイベントの特性が戻されます。 次の項では、各イベント ディテクタで戻される特性を示します。

EEM_EVENT_APPLICATION について

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u"
"sub_system 0x%x type %u data1 {%s} data2 {%s} data3 {%s} data4 {%s}"
```

イベント タイプ	説明
event_id	パブリッシュされた該当イベントの ID を示す一意の番号。 同一のイベントで複数のポリシーを実行可能であり、その場合、各ポリシーは同一の event_id を保持します。
event_type	イベントのタイプ。
event_type_string	このイベント タイプのイベントの名前を表す ASCII 文字列。
event_pub_secevent_pub_msec	イベントが Embedded Event Manager にパブリッシュされたときの、秒単位およびミリ秒単位の時間。
sub_system	アプリケーション イベントをパブリッシュした EEM ポリシーに割り当てられる番号。 他のすべての番号は Cisco での使用のために予約されているため、番号は 798 に設定されます。
type	指定されたコンポーネント内のイベント サブタイプ。
data1data2data3data4	イベントがパブリッシュされるときに、アプリケーション固有のイベントに渡される、引数データ。 データは、文字テキスト、環境変数、または、この 2 つの組み合わせです。

EEM_EVENT_COUNTER について

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
"name {%s}"
```

イベント タイプ	説明
event_id	パブリッシュされた該当イベントの ID を示す一意の番号。同一のイベントで複数のポリシーを実行可能であり、その場合、各ポリシーは同一の event_id を保持します。
event_type	イベントのタイプ。
event_type_string	このイベント タイプのイベントの名前を表す ASCII 文字列。
event_pub_secevent_pub_msec	イベントが Embedded Event Manager にパブリッシュされたときの、秒単位およびミリ秒単位の時間。
name	カウンタ名。

EEM_EVENT_NONE について

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u"
```

イベント タイプ	説明
event_id	パブリッシュされた該当イベントの ID を示す一意の番号。同一のイベントで複数のポリシーを実行可能であり、その場合、各ポリシーは同一の event_id を保持します。
event_type	イベントのタイプ。
event_type_string	このイベント タイプのイベントの名前を表す ASCII 文字列。
event_pub_secevent_pub_msec	イベントが Embedded Event Manager にパブリッシュされたときの、秒単位およびミリ秒単位の時間。

EEM_EVENT_OIR について

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u"
"slot %u event %s"
```

イベント タイプ	説明
event_id	パブリッシュされた該当イベントの ID を示す一意の番号。同一のイベントで複数のポリシーを実行可能であり、その場合、各ポリシーは同一のイベント ID を保持します。
event_type	イベントのタイプ。
event_type_string	このイベント タイプのイベントの名前を表す ASCII 文字列。
event_pub_secevent_pub_msec	イベントが Embedded Event Manager にパブリッシュされたときの、秒単位およびミリ秒単位の時間。
slot	影響が及ぼされるカードのスロット番号。
event	OIR の削除イベントまたは OIR の挿入イベントを表す、removed または online の文字列を示します。

EEM_EVENT_PROCESS について（ソフトウェアモジュール方式のみ）

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u"
"sub_system 0x%x instance %u process_name {%s} path {%s} exit_status 0x%x"
"respawn_count %u last_respawn_sec %ld last_respawn_msec %ld fail_count %u"
"dump_count %u node_name {%s}"
```

イベント タイプ	説明
event_id	パブリッシュされた該当イベントの ID を示す一意の番号。同一のイベントで複数のポリシーを実行可能であり、その場合、各ポリシーは同一の event_id を保持します。
event_type	イベントのタイプ。
event_type_string	このイベント タイプのイベントの名前を表す ASCII 文字列。
event_pub_secevent_pub_msec	イベントが Embedded Event Manager にパブリッシュされたときの、秒単位およびミリ秒単位の時間。

イベント タイプ	説明
sub_system	アプリケーション固有のイベントをパブリッシュしたEEMポリシーに割り当てられる番号。他のすべての番号は Cisco での使用のために予約されているため、番号は 798 に設定されます。
instance	プロセス インスタンス ID。
process_name	プロセス名。
path	パスを含むプロセスの絶対名。
exit_status	プロセスの最後の終了ステータス。
respawn_count	プロセスが再起動された回数。
last_respawn_seclast_respawn_msec	最後の再起動が発生したカレンダー時間。
fail_count	失敗したプロセスの再起動試行の回数。プロセスが正常に再起動されると、0 にリセットされます。
イベント タイプ	**説明**
dump_count	プロセスで取られたコア ダンプの数。
node_name	プロセスが存在するノードの名前。 ノード名は、「node」という語句と、それに続く、次の形式を使用してスラッシュ文字で区切られた 2 つのフィールドで構成される、文字列です。 node<slot-number>/<cpu-number> slot-number は、ハードウェア スロット番号です。 cpu-number は、ハードウェア CPU 番号です。

EEM_EVENT_RF について

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
"event {%s}"
```

イベント タイプ	説明
event_id	パブリッシュされた該当イベントの ID を示す一意の番号。同一のイベントで複数のポリシーを実行可能であり、その場合、各ポリシーは同一の event_id を保持します。
event_type	イベントのタイプ。
event_type_string	このイベント タイプのイベントの名前を表す ASCII 文字列。
event_pub_secevent_pub_msec	イベントが Embedded Event Manager にパブリッシュされたときの、秒単位およびミリ秒単位の時間。
event	このイベントが発生する原因となる RF の進行またはステータス イベント通知。

EEM_EVENT_SYSLOG_MSG について

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u"
"msg {%s}"
```

イベント タイプ	説明
event_id	パブリッシュされた該当イベントの ID を示す一意の番号。同一のイベントで複数のポリシーを実行可能であり、その場合、各ポリシーは同一の event_id を保持します。
event_type	イベントのタイプ。
event_type_string	このイベント タイプのイベントの名前を表す ASCII 文字列。
event_pub_secevent_pub_msec	イベントが Embedded Event Manager にパブリッシュされたときの、秒単位およびミリ秒単位の時間。
msg	パターンと一致する最後の Syslog メッセージ。

EEM_EVENT_TIMER_ABSOLUTE

EEM_EVENT_TIMER_COUNTDOWN

EEM_EVENT_TIMER_WATCHDOG について

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u"
"timer_type %s timer_time_sec %ld timer_time_msec %ld"
"timer_remain_sec %ld timer_remain_msec %ld"
```

イベント タイプ	説明
event_id	パブリッシュされた該当イベントの ID を示す一意の番号。同一のイベントで複数のポリシーを実行可能であり、その場合、各ポリシーは同一の event_id を保持します。
event_type	イベントのタイプ。
event_type_string	このイベント タイプのイベントの名前を表す ASCII 文字列。
event_pub_secevent_pub_msec	イベントが Embedded Event Manager にパブリッシュされたときの、秒単位およびミリ秒単位の時間。
timer_type	タイマーのタイプ。 次のいずれかです。 • watchdog • countdown • absolute
timer_time_sectimer_time_msec	タイマーの期限が切れる時間。
timer_remain_sectimer_remain_msec	次の期限切れ前の残りの時間。

EEM_EVENT_TIMER_CRON について

```
"event_id %u event_type %u event_type_string {%s} event_pub_sec %u event_pub_msec %u"
"timer_type {%s} timer_time_sec %ld timer_time_msec %ld"
```

イベント タイプ	説明
event_id	パブリッシュされた該当イベントの ID を示す一意の番号。同一のイベントで複数のポリシーを実行可能であり、その場合、各ポリシーは同一の event_id を保持します。
event_type	イベントのタイプ。
event_type_string	このイベント タイプのイベントの名前を表す ASCII 文字列。
event_pub_secevent_pub_msec	イベントが Embedded Event Manager にパブリッシュされたときの、秒単位およびミリ秒単位の時間。
timer_type	タイマーのタイプ。
timer_time_sectimer_time_msec	タイマーの期限が切れる時間。

EEM_EVENT_TRACK について

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
"track_number {%u} track_state {%s}"
```

イベント タイプ	説明
event_id	パブリッシュされた該当イベントの ID を示す一意の番号。同一のイベントで複数のポリシーを実行可能であり、その場合、各ポリシーは同一のイベント ID を保持します。
event_type	イベントのタイプ。
event_type_string	このイベント タイプのイベントの名前を表す ASCII 文字列。
event_pub_secevent_pub_msec	イベントが Embedded Event Manager にパブリッシュされたときの、秒単位およびミリ秒単位の時間。
track_number	イベントがトリガーされる原因となるトラックされるオブジェクトの番号。

イベント タイプ	説明
track_state	イベントがトリガーされたときのトラックされるオブジェクトの状態。有効な値は up または down です。

EEM_EVENT_WDSYSMON について

```
"event_id %u event_type %u event_type_string {%s} %u event_pub_sec %u event_pub_msec %u"
"num_subs %u"
```

イベント タイプ	説明
event_id	パブリッシュされた該当イベントの ID を示す一意の番号。同一のイベントで複数のポリシーを実行可能であり、その場合、各ポリシーは同一の event_id を保持します。
event_type	イベントのタイプ。
event_type_string	このイベント タイプのイベントの名前を表す ASCII 文字列。
event_pub_secevent_pub_msec	イベントが Embedded Event Manager にパブリッシュされたときの、秒単位およびミリ秒単位の時間。
num_subs	サブイベント番号。

サブイベント情報文字列は、次のような、デッドロック サブイベント用です。

```
"{type %s num_entries %u entries {entry 1, entry 2, ...}}"
```

サブイベント タイプ	説明
type	Wdsysmon サブイベントのタイプ。
num_entries	デッドロックのプロセスおよびスレッドの番号。
entries	デッドロックのプロセスおよびスレッドの情報。

各エントリは次のとおりです。

```
"{node {%s} procname {%s} pid %u tid %u state %s b_node %s b_procname %s b_pid %u
b_tid %u}"
```

このエントリでは、プロセス A のスレッド m によって、プロセス B のスレッド n でブロックされるシナリオが記述されているとすると、次のようになります。

サブイベント タイプ	説明
node	プロセス A のスレッド m があるノードの名前。
procname	プロセス A の名前。
pid	プロセス A のプロセス ID。
tid	プロセス A のスレッド m のスレッド ID。
state	プロセス A のスレッド m のスレッドの状態。次のいずれかです。 • STATE_CONDVAR • STATE_DEAD • STATE_INTR • STATE_JOIN • STATE_MUTEX • STATE_NANOSLEEP • STATE_READY • STATE_RECEIVE • STATE_REPLY • STATE_RUNNING • STATE_SEM • STATE_SEND • STATE_SIGSUSPEND • STATE_SIGWAITINFO • STATE_STACK • STATE_STOPPED • STATE_WAITPAGE • STATE_WAITTHREAD
b_node	プロセス B のスレッドがあるノードの名前。

サブイベント タイプ	説明
b_procname	プロセス B の名前。
b_pid	プロセス B のプロセス ID。
b_tid	プロセス B のスレッド n のスレッド ID。0 は、プロセス A のスレッド m は、プロセス B のすべてのスレッド上でブロックされることを意味します。

dispatch_mgr サブイベントについて

```
"{type %s node {%s} procname {%s} pid %u value %u sec %ld msec %ld}"
```

サブイベント タイプ	説明
type	Wdsysmon サブイベントのタイプ。
node	POSIX プロセスが存在するノードの名前。
procname	このサブイベントの POSIX プロセス名。
pid	このサブイベントの POSIX プロセス ID。 （注） 前述の 3 つのフィールドは、このディスパッチ マネージャのオーナー プロセスについて説明します。
value	sec 変数と msec 変数が、0 に指定されるか、または、イベント登録 Tcl コマンド拡張で指定されない場合、ディスパッチマネージャによって処理されるイベント数は、最新のサンプルにあります。 時間ウィンドウが指定され、登録 Tcl コマンド拡張でゼロよりも大きい場合、このディスパッチマネージャによって処理されるイベントの合計数は、該当する時間ウィンドウにあります。

サブイベント タイプ	説明
secmsec	イベント登録 Tcl コマンド拡張で、sec 変数と msec 変数が 0 に指定されているか、または指定されていない場合、両方とも 0 です。 登録 Tcl コマンド拡張で時間ウィンドウが指定され、かつその時間ウィンドウがゼロよりも大きい場合、sec 変数および msec 変数は、この時間ウィンドウの最も古いサンプルと最新のサンプルとの実際の時間の差分です。

cpu_proc サブイベントについて

```
"{type %s node {%s} procname {%s} pid %u value %u sec %ld msec %ld}"
```

サブイベント タイプ	説明
type	Wdsysmon サブイベントのタイプ。
node	POSIX プロセスが存在するノードの名前。
procname	このサブイベントの POSIX プロセス名。
pid	このサブイベントの POSIX プロセス ID。 （注） 前述の 3 つのフィールドは、その CPU 使用率がモニタされているプロセスについて説明します。
value	sec 変数と msec 変数が、0 に指定されるか、または、イベント登録 Tcl コマンド拡張で指定されない場合、プロセス CPU 使用率は、最新のサンプルにあります。時間ウィンドウが指定され、登録 Tcl コマンド拡張でゼロよりも大きい場合、プロセス CPU 使用率の平均は、該当する時間ウィンドウにあります。
secmsec	イベント登録 Tcl コマンド拡張で、sec 変数と msec 変数が 0 に指定されているか、または指定されていない場合、両方とも 0 です。 登録 Tcl コマンド拡張で時間ウィンドウが指定され、かつその時間ウィンドウがゼロよりも大きい場合、sec 変数および msec 変数は、この時間ウィンドウの最も古いサンプルと最新のサンプルとの実際の時間の差分です。

cpu_tot サブイベントについて

```
"{type %s node {%s} value %u sec %ld msec %ld}"
```

サブイベント タイプ	説明
type	Wdsysmon サブイベントのタイプ。
node	CPU使用率の合計がモニタされているノードの名前。
value	sec 変数と msec 変数が、0 に指定されるか、または、イベント登録 Tcl コマンド拡張で指定されない場合、合計 CPU 使用率は、最新のサンプルにあります。時間ウィンドウが指定され、登録Tclコマンド拡張でゼロよりも大きい場合、合計 CPU 使用率の平均は、該当する時間ウィンドウにあります。
secmsec	イベント登録 Tcl コマンド拡張で、sec 変数と msec 変数が 0 に指定されているか、または指定されていない場合、両方とも 0 です。 登録 Tcl コマンド拡張で時間ウィンドウが指定され、かつその時間ウィンドウがゼロよりも大きい場合、sec 変数および msec 変数は、この時間ウィンドウの最も古いサンプルと最新のサンプルとの実際の時間の差分です。

mem_proc サブイベントについて

```
"{type %s node {%s} procname {%s} pid %u is_percent %s value %u diff %d sec %ld msec %ld}"
```

サブイベント タイプ	説明
type	Wdsysmon サブイベントのタイプ。
node	POSIX プロセスが存在するノードの名前。
procname	このサブイベントの POSIX プロセス名。
pid	このサブイベントの POSIX プロセス ID。 （注） 前述の 3 つのフィールドは、そのメモリ使用率がモニタされているプロセスについて説明します。

サブイベント タイプ	説明
is_percent	TRUE または FALSE のいずれかです。 TRUE は、値がパーセント値であることを示します。FALSE は、値が絶対値であることを示します（平均値の場合もあります）。
value	sec 変数と msec 変数が、0 に指定されるか、または、イベント登録 Tcl コマンド拡張で指定されない場合、プロセスで使用されたメモリは、最新のサンプルにあります。時間ウィンドウが指定され、登録 Tcl コマンド拡張でゼロよりも大きい場合、プロセスで使用されたメモリ使用率の平均は、該当する時間ウィンドウにあります。
サブイベント タイプ	**説明**
diff	sec 変数と msec 変数が、0 に指定されるか、または、イベント登録 Tcl コマンド拡張で指定されない場合、diff は、今まで収集された、プロセスで使用された最初のメモリ サンプルと、プロセスで使用された最新のメモリ サンプルとのパーセンテージによる差分です。時間ウィンドウが指定され、登録 Tcl コマンド拡張でゼロよりも大きい場合、diff は、指定された時間ウィンドウで、最も古いプロセスで使用されたメモリ使用率と最新のプロセスで使用されたメモリ使用率とのパーセンテージによる差分です。
secmsec	イベント登録 Tcl コマンド拡張で、sec 変数と msec 変数が 0 に指定されているか、または指定されていない場合、両方とも 0 です。 登録 Tcl コマンド拡張で時間ウィンドウが指定され、かつその時間ウィンドウがゼロよりも大きい場合、sec 変数および msec 変数は、この時間ウィンドウの最も古いサンプルと最新のサンプルとの実際の時間の差分です。

is_percent 引数が FALSE であり、イベント登録 Tcl コマンド拡張で *sec* 引数と *msec* 引数が 0 に指定されているか、または指定されていない場合は、次のようになります。

- *value* は、最新のサンプルでプロセスによって使用されたメモリです。
- *diff* は 0 です。

- *sec* と *msec* は、両方とも 0 です。

is_percent 引数が FALSE であり、イベント登録 Tcl コマンド拡張で時間ウィンドウがゼロよりも大きい値に指定されている場合は、次のようになります。

- *value* は、指定された時間ウィンドウでプロセスによって使用されたメモリ サンプル値の平均です。
- *diff* は 0 です。
- *sec* および *msec* は、両方とも、この時間ウィンドウの、最も古いタイム スタンプと最新のタイム スタンプとの実際の時間の差分です。

is_percent 引数が TRUE であり、イベント登録 Tcl コマンド拡張で時間ウィンドウがゼロよりも大きい値に指定されている場合は、次のようになります。

- value は 0 です。
- *diff* は、指定された時間ウィンドウの、最も古いプロセスで使用されたメモリ サンプルと最新のプロセスで使用されたメモリ サンプルとのパーセンテージによる差分です。
- *sec* および **msec** は、この時間ウィンドウの、最も古いプロセスで使用されたメモリ サンプルと最新のプロセスで使用されたメモリ サンプルとの実際の時間の差分です。

is_percent 引数が TRUE であり、イベント登録 Tcl コマンド拡張で *sec* 引数と *msec* 引数が 0 に指定されているか、または指定されていない場合は、次のようになります。

- *value* は 0 です。
- *diff* は、今まで収集された、プロセスで使用された最新のメモリ サンプルと、プロセスで使用された最新のメモリ サンプルとのパーセンテージによる差分です。
- *sec* および *msec* は、今まで収集された、プロセスで使用された最初のメモリ サンプルとプロセスで使用された最新のメモリ サンプルとの実際の時間の差分です。

mem_tot_avail サブイベントについて

```
"{type %s node {%s} is_percent %s used %u avail %u diff %d sec %ld msec %ld}"
```

サブイベント タイプ	説明
type	Wdsysmon サブイベントのタイプ。
node	使用可能なメモリの合計がモニタされているノードの名前。
is_percent	TRUE または FALSE のいずれかです。 TRUE は、値がパーセント値であることを示します。FALSE は、値が絶対値であることを示します（平均値の場合もあります）。

サブイベント タイプ	説明
used	sec 変数と msec 変数が、0 に指定されるか、または、イベント登録 Tcl コマンド拡張で指定されない場合、使用されたメモリの合計は、最新のサンプルにあります。時間ウィンドウが指定され、登録 Tcl コマンド拡張でゼロよりも大きい場合、使用された合計メモリ使用率の平均は、該当する時間ウィンドウにあります。
avail	sec 変数と msec 変数が、0 に指定されるか、または、イベント登録 Tcl コマンド拡張で指定されない場合、availは、最新の使用可能なメモリ サンプルの合計にあります。時間ウィンドウが指定され、登録 Tcl コマンド拡張でゼロよりも大きい場合、availは、指定された時間ウィンドウで使用可能なメモリ使用率の合計です。
diff	sec 変数と msec 変数が、0 に指定されるか、または、イベント登録 Tcl コマンド拡張で指定されない場合、diff は、今まで収集された、最初の使用可能なメモリ サンプルの合計と、最新の使用可能なメモリ サンプルの合計とのパーセンテージによる差分です。時間ウィンドウが指定され、登録 Tcl コマンド拡張でゼロよりも大きい場合、diffは、指定された時間ウィンドウで、最も古い使用可能なメモリ使用率と最新の使用可能なメモリ使用率とのパーセンテージによる差分です。
secmsec	イベント登録 Tcl コマンド拡張で、sec 変数と msec 変数が 0 に指定されているか、または指定されていない場合、両方とも 0 です。時間ウィンドウが指定され、登録 Tcl コマンド拡張でゼロよりも大きい場合、これらの変数は、この時間ウィンドウの、最も古いサンプルと最新のサンプルとの実際の時間の差分です。

is_percent 引数が FALSE であり、イベント登録 Tcl コマンド拡張で sec 引数と msec 引数が 0 に指定されているか、または指定されていない場合は、次のようになります。

- *used* は、最新のサンプルで使用されたメモリの合計です。
- *avail* は、最新のサンプルで使用可能なメモリの合計です。
- *diff* は 0 です。

- *sec* と **msec** は、両方とも 0 です。

is_percent 引数が FALSE であり、イベント登録 Tcl コマンド拡張で時間ウィンドウがゼロよりも大きい値に指定されている場合は、次のようになります。

- *used* は 0 です。
- *avail* は、指定された時間ウィンドウで使用可能なメモリ サンプル値の平均です。
- *diff* は 0 です。
- *sec* および *msec* は、両方とも、この時間ウィンドウの、最も古い使用可能なメモリ サンプルの合計のタイム スタンプと最新の使用可能なメモリ サンプルの合計のタイム スタンプとの実際の時間の差分です。

is_percent 引数が TRUE であり、イベント登録 Tcl コマンド拡張で時間ウィンドウがゼロよりも大きい値に指定されている場合は、次のようになります。

- *used* は 0 です。
- *avail* は 0 です。
- *diff* は、指定された時間ウィンドウの、最も古い使用可能なメモリ サンプルの合計と最新の可能なメモリ サンプルの合計とのパーセンテージによる差分です。
- *sec* および *msec* は、両方とも、この時間ウィンドウの、最も古い使用可能なメモリ サンプルの合計のタイム スタンプと最新の使用可能なメモリ サンプルの合計のタイム スタンプとの実際の時間の差分です。

is_percent 引数が TRUE であり、イベント登録 Tcl コマンド拡張で *sec* 引数と *msec* 引数が 0 に指定されているか、または指定されていない場合は、次のようになります。

- *used* は 0 です。
- *avail* は 0 です。
- *diff* は、今まで収集された、最初の使用可能なメモリ サンプルの合計と、最新の使用可能なメモリ サンプルの合計との間の、パーセンテージによる差です。
- *sec* および msec は、今まで収集された、使用可能な最初のメモリ サンプルの合計と使用可能な最新のメモリ サンプルの合計との間の、実際の時間の差です。

mem_tot_used サブイベントについて

```
"{type %s node {%s} is_percent %s used %u avail %u diff %d sec %ld msec %ld}"
```

サブイベント タイプ	説明
type	Wdsysmon サブイベントのタイプ。
node	使用されているメモリの合計がモニタされているノードの名前。

サブイベント タイプ	説明
is_percent	TRUE または FALSE のいずれかです。 TRUE は、値がパーセント値であることを示します。FALSE は、値が絶対値であることを示します（平均値の場合もあります）。
used	sec 変数と msec 変数が、0 に指定されるか、または、イベント登録 Tcl コマンド拡張で指定されない場合、使用されたメモリの合計は、最新のサンプルにあります。時間ウィンドウが指定され、登録 Tcl コマンド拡張でゼロよりも大きい場合、使用された合計メモリ使用率の平均は、該当する時間ウィンドウにあります。
avail	sec 変数と msec 変数が、0 に指定されるか、または、イベント登録 Tcl コマンド拡張で指定されない場合、availは、最新の使用されたメモリサンプルの合計にあります。時間ウィンドウが指定され、登録 Tcl コマンド拡張でゼロよりも大きい場合、availは、指定された時間ウィンドウで使用されたメモリ使用率の合計です。
diff	sec 変数と msec 変数が、0 に指定されるか、または、イベント登録 Tcl コマンド拡張で指定されない場合、diff は、今まで収集された、プロセスで使用可能な最初のメモリ サンプルと、プロセスで使用可能な最新のメモリ サンプルとのパーセンテージによる差分です。時間ウィンドウが指定され、登録 Tcl コマンド拡張でゼロよりも大きい場合、diff は、指定された時間ウィンドウで、最も古い使用されたメモリ使用率の合計と最新の使用されたメモリ使用率の合計とのパーセンテージによる差分です。
secmsec	イベント登録 Tcl コマンド拡張で、sec 変数と msec 変数が 0 に指定されているか、または指定されていない場合、両方とも 0 です。 登録 Tcl コマンド拡張で時間ウィンドウが指定され、かつその時間ウィンドウがゼロよりも大きい場合、sec 変数および msec 変数は、この時間ウィンドウの最も古いサンプルと最新のサンプルとの実際の時間の差分です。

is_percent 引数が FALSE であり、イベント登録 Tcl コマンド拡張で *sec* 引数と *msec* 引数が 0 に指定されているか、または指定されていない場合は、次のようになります。

- *used* は、最新のサンプルで使用されたメモリの合計です。
- *avail* は、最新のサンプルで使用可能なメモリの合計です。
- *diff* は 0 です。
- *sec* と *msec* は、両方とも 0 です。

is_percent 引数が FALSE であり、イベント登録 Tcl コマンド拡張で時間ウィンドウがゼロよりも大きい値に指定されている場合は、次のようになります。

- *used* は、指定された時間ウィンドウで使用されたメモリ サンプル値の平均です。
- *avail* は 0 です。
- *diff* は 0 です。
- *sec* および *msec* は、両方とも、この時間ウィンドウの、最も古い使用されたメモリ サンプルのタイム スタンプの合計と最新の使用されたメモリ サンプルの合計のタイム スタンプとの間の、実際の時間の差です。

is_percent 引数が TRUE であり、イベント登録 Tcl コマンド拡張で時間ウィンドウがゼロよりも大きい値に指定されている場合は、次のようになります。

- *used* は 0 です。
- *avail* は 0 です。
- *diff* は、指定された時間ウィンドウの、最も古い使用されたメモリ サンプルの合計と最新の使用されたメモリ サンプルの合計との間の、パーセンテージによる差です。
- *sec* および *msec* は、両方とも、この時間ウィンドウの、最も古い使用されたメモリ サンプルのタイム スタンプの合計と最新の使用されたメモリ サンプルのタイム スタンプの合計との間の、実際の時間の差です。

is_percent 引数が TRUE であり、イベント登録 Tcl コマンド拡張で sec 引数と msec 引数が 0 に指定されているか、または指定されていない場合は、次のようになります。

- *used* は 0 です。
- *avail* は 0 です。
- *diff* は、今まで収集された、使用された最初のメモリ サンプルの合計と、プロセスで使用された最新のメモリ サンプルの合計との間の、パーセンテージによる差です。
- *sec* および *msec* は、今まで収集された、使用された最初のメモリ サンプルの合計と使用された最新のメモリ サンプルの合計との間の、実際の時間の差です。

_cerrno を設定

Yes

event_reqinfo_multi

スクリプトのトリガーに寄与した各イベントに対して event_reqinfo データを取得するための新機能を追加します。 返されるデータは、イベント指定タグでインデックス化された結果文字列のリストです。 エラー処理は、event_reqinfo 機能と同じです。

構文

```
event_reqinfo_multi
```

引数

なし

結果文字列

ここでは、event reqinfo multi の呼び出しの結果の文字列を示します。

```
"<ev-tag> {event_id %u event_type %u event_type_string
{%s} event_pub_sec %ld event_pub_msec %ld timer_type {%s} timer_time_sec
%ld timer_time_msec %ld timer_remain_sec %ld timer_remain_msec %ld}
                 <ev-tag> {event_id %u event_type %u event_type_string
{%s} event_pub_sec %ld event_pub_msec %ld oid {%s} val {%s} delta_val
{%s} exit_event {%s}}"
                Typical usage for a multi-event consisting of both a timer event and an
SNMP event might be:
                array set arr_minfo [event_reqinfo_multi]
                if {$_cerrno != 0} {
                    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
                        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
                    error $result
                }
                array set arr_einfo $arr_minfo(<ev-tag-for-timer-event-spec>)
                global timer_type timer_time_sec
                set timer_type $arr_einfo(timer_type)
                set timer_time_sec $arr_einfo(timer_time_sec)
```

event_reqinfo_multi の出力は、ポリシーのトリガーに寄与した最も新しいイベントから最も古いイベントの順に配置されます。

Embedded Event Manager イベント パブリッシュ Tcl コマンド拡張

event_publish appl

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。

構文

```
event_publish sub_system ? type ? [arg1 ?] [arg2 ?] [arg3 ?] [arg4 ?]
```

引数

sub_system	（必須）アプリケーション固有のイベントをパブリッシュした EEM ポリシーに割り当てられる番号。 他のすべての番号は Cisco での使用のために予約されているため、番号は 798 に設定されます。
type	（必須）指定されたコンポーネント内のイベント サブタイプ。 sub_system 引数および type 引数によって、アプリケーション イベントが一意に識別されます。 1 ～ 4294967295 の範囲の整数である必要があります。
[arg1 ?]-[arg4 ?]	（任意）4 つのアプリケーション イベントのパブリッシャの文字列データ。

結果文字列

なし

_cerrno を設定

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

このエラーは、オペレーティング システムによってレポートされたエラーを意味します。エラーとともにレポートされる POSIX errno 値を使用して、オペレーティング システム エラーの原因を調べます。

使用例

この例では、同じ機能（たとえば、Tcl 文の指定されたグループによって CPU 時間の長さを測定する）を実行するため、**event_publish appl** Tcl コマンド拡張を使用して、スクリプトを *n* 回実行する方法を示します。 この例では、2 つの Tcl スクリプトが使用されます。

Script1 によって、タイプ 9999 EEM イベントがパブリッシュされ、Script2 の 1 回目の実行が行われます。 Script1 は、none イベントとして登録され、Cisco IOS XR ソフトウェア CLI **event manager run** コマンドを使用して実行されます。 Script2 は、タイプ 9999 の EEM アプリケーション イベントとして登録され、このスクリプトによって、アプリケーションによってパブリッシュされた arg1 データ（繰り返し回数）が、EEM 環境変数 test_iterations の値を超過したかどうかがチェックされます。 test_iterations の値が超えた場合、スクリプトによってメッセージが書き込まれ、終了します。これ以外の場合、スクリプトによって残りの文が実行され、別の実行が再スケジュールされます。 Script2 の CPU 使用率を測定するには、10 の倍数である test_iterations の値を使用して、Script2 によって使用される CPU 時間の平均の長さを計算します。

Tcl スクリプトを実行するには、次の Cisco IOS XR ソフトウェア コマンドを使用します。

```
configure terminal
 event manager environment test_iterations 100
 event manager policy script1.tcl
 event manager policy script2.tcl
 end
event manager run script1.tcl
```

Tcl スクリプト Script2 によって、100 回実行されます。 余分な処理なしてスクリプトを実行し、CPU 使用率の平均を導き出し、次に余分な処理を追加して、テストを繰り返す場合、以降の CPU 使用率から前の CPU 使用率を差し引き、余分な処理の平均を調べることができます。

Script1（script1.tcl）

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

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

# Query the event info.
array set arr_einfo [event_reqinfo]
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}

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

# Cause the first iteration to run.
event_publish sub_system 798 type 9999 arg1 0
if {$_cerrno != 0} {
    set result [format \
          "component=%s; subsys err=%s; posix err=%s;\n%s" \
          $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
```

Script2（script2.tcl）

```
::cisco::eem::event_register_appl sub_system 798 type 9999

# Check if all the required environment variables exist.
# If any required environment variable does not exist, print out an error msg and quit.
if {![info exists test_iterations]} {
    set result \
      "Policy cannot be run: variable test_iterations has not been set"
    error $result $errorInfo
}

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

# Query the event info.
array set arr_einfo [event_reqinfo]
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
        $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
# Data1 contains the arg1 value used to publish this event.
```

```
set iter $arr_einfo(data1)

# Use the arg1 info from the previous run to determine when to end.
if {$iter >= $test_iterations} {
    # Log a message.
    action_syslog priority info msg "EEM application_publish test end"
    if {$_cerrno != 0} {
      set result [format \
            "component=%s; subsys err=%s; posix err=%s;\n%s" \
            $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
      error $result
    }
    exit 0
}
set iter [expr $iter + 1]

# Log a message.
set msg [format "EEM application_publish test iteration %s" $iter]
action_syslog priority info msg $msg
if {$_cerrno != 0} {
    set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
      $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}

# Do whatever processing that you want to measure here.

# Cause the next iteration to run. Note that the iteration is passed to the
# next operation as arg1.
event_publish sub_system 798 type 9999 arg1 $iter
if {$_cerrno != 0} {
    set result [format \
          "component=%s; subsys err=%s; posix err=%s;\n%s" \
          $_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
    error $result
}
```

Embedded Event Manager 複数イベント サポート Tcl コマンド拡張

属性

複数イベント サポートに使用する複雑なイベントを指定します。

構文

```
attribute tag ? [occurs ?]
```

引数

tag	イベントを関連付けるために **attribute** コマンドで使用できる *event-tag* 引数を使用して、タグを指定します。
occurs	（任意）EEMイベントがトリガーされる前の発生数を指定します。 指定されない場合、EEMイベントは1回目から発生します。 範囲は1～4294967295 です。

結果文字列

なし

例：

attribute tag 1 occurs 1

correlate

1 つの複雑なイベントを構築し、ブール値のロジックを許可してイベントを関連付けます。

構文

```
correlate event ? event ?
```

引数

event	スクリプト内で複数のイベント文をサポートするために、**trigger** コマンドで使用できるイベントを指定します。 *event-tag* 引数に関連付けられているイベントが、□**rigger** コマンドによって指定されて何度も発生する場合、結果は真です。 これ以外の場合、結果は偽です。
andnot	（任意）イベント 1 が発生した場合にアクションが実行され、さらに、イベント 2 およびイベント 3 が一緒に発生した場合にはアクションが実行されないよう、指定します。
and	（任意）イベント 1 が発生した場合にアクションが実行され、さらに、イベント 2 およびイベント 3 が一緒に発生した場合にアクションが実行されるよう、指定します。
or	（任意）イベント 1 が発生した場合にアクションが実行されるか、または、イベント 2 およびイベント 3 が一緒に発生した場合にアクションが実行されるよう、指定します。

結果文字列

なし

例：

correlate event 1 or event 2 and event 3

Trigger

Embedded Event Manager（EEM）イベントの複数イベントの設定機能を指定します。 複数イベントは、1つまたは複数のイベント発生、および発生するイベントの時間に関連するイベントです。イベントは指定されたパラメータに基づいて発生します。

構文

```
trigger [occurs ?] [period ?] [period-start ?] [delay ?]
```

引数

occurs	（任意）EEMイベントが発生する前に発生した合計相関回数を指定します。 数が指定されない場合、EEMイベントは1回目から発生します。範囲は 1 ～ 4294967295 です。
period	（任意）1 つまたは複数が発生する必要がある間の、秒単位、および、任意でミリ秒単位での、時間の間隔。 これは、sssssssss[.mmm] 形式で指定します。sssssssss は、0 ～ 4294967295 の秒数を表す整数で、mmm は 0 ～ 999 のミリ秒数を表す整数である必要があります。
period-start	（任意）イベント相関ウィンドウの開始を指定します。 指定されない場合、最初の CRON 期間の発生後、イベント監視はイネーブルにされます。
delay	（任意）すべての条件が真の場合にイベントの発生後の秒数とミリ秒数（任意）を指定します（sssssssss[.mmm] 形式で指定します。sssssssss は、0 ～ 4294967295 の秒数を表す整数で、mmm は 0 ～ 999 のミリ秒数を表す整数である必要があります）。

結果文字列

なし

例：

trigger occurs 1 period-start "0 8 * * 1-5" period 720

Embedded Event Manager アクション Tcl コマンド拡張

action_process

ソフトウェア モジュール方式プロセスを起動、再起動、または停止します。 この Tcl コマンド拡張は、ソフトウェア モジュール方式イメージでのみサポートされます。

構文

```
action_process start|restart|kill [job_id ?]
[process_name ?] [instance ?]
```

引数

start	（必須）プロセスが起動されるよう指定します。
restart	（必須）プロセスが再起動されるよう指定します。
kill	（必須）プロセスが停止されるよう指定します。
job_id	（任意）システムマネージャによってプロセスに割り当てられるジョブ ID。 この引数を指定する場合、1 ～ 4294967295 の範囲の整数である必要があります。
process_name	（任意）プロセス名。 job_id を指定するか、または、process_name および instance を指定する必要があります。
instance	（任意）プロセス インスタンス ID。 この引数を指定する場合、1 ～ 4294967295 の範囲の整数である必要があります。

結果文字列

なし

_cerrno を設定

Yes

```
(_cerr_sub_err = 14)    FH_ENOSUCHACTION  (unknown action type)
```

このエラーは、要求されたアクション コマンドが未知であることを示します。

```
(_cerr_sub_num = 425, _cerr_sub_err = 1) SYSMGR_ERROR_INVALID_ARGS  (Invalid arguments
passed)
```

このエラーは、渡された引数が無効であったことを意味します。

```
(_cerr_sub_num = 425, _cerr_sub_err = 2) SYSMGR_ERROR_NO_MEMORY  (Could not allocate required
 memory)
```

このエラーは、メモリの内部 SYSMGR 要求に障害が発生したことを意味します。

```
(_cerr_sub_num = 425, _cerr_sub_err = 5) SYSMGR_ERROR_NO_MATCH  (This process is not known
 to sysmgr)
```

このエラーは、プロセス名が未知であったことを意味します。

```
(_cerr_sub_num = 425, _cerr_sub_err = 14) SYSMGR_ERROR_TOO_BIG  (outside the valid limit)
```

このエラーは、オブジェクト サイズがその最大値を超えたことを意味します。

```
(_cerr_sub_num = 425, _cerr_sub_err = 15) SYSMGR_ERROR_INVALID_OP  (Invalid operation for
this process)
```

このエラーは、その動作がプロセスに対して無効であったことを意味します。

action_program

Tcl スクリプトで、POSIX プロセス（プログラム）を実行できるようにします。任意で、該当する引数文字列、環境文字列、標準入力（stdin）パス名、標準出力（stdout）パス名、または標準エラー（stderr）パス名を使用します。 この Tcl コマンド拡張は、ソフトウェア モジュール方式イメージでのみサポートされます。

構文

```
action_program path ? [argv ?] [envp ?] [stdin ?] [stdout ?] [stderr ?]
```

引数

path	（必須）実行するプログラムのパス名。
argv	（任意）プログラムの引数文字列。
envp	（任意）プログラムの環境文字列。
stdin	（任意）stdin のパス名。

stdout	（任意）stdout のパス名。
stderr	（任意）stderr のパス名。

結果文字列

なし

_cerrno を設定

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

このエラーは、オペレーティング システムによってレポートされたエラーを意味します。エラーとともにレポートされる POSIX errno 値を使用して、オペレーティング システム エラーの原因を調べます。

```
(_cerr_sub_err = 14)    FH_ENOSUCHACTION  (unknown action type)
```

このエラーは、要求されたアクション コマンドが未知であることを示します。

```
(_cerr_sub_err = 34)    FH_EMAXLEN  (maximum length exceeded)
```

このエラーは、オブジェクト長またはオブジェクト数が、最大値を超えたことを意味します。

action_script

Tcl スクリプトで、すべての Tcl スクリプトの実行をイネーブルまたはディセーブルにします（スクリプト スケジューラをイネーブルまたはディセーブルにします）。

構文

```
action_script [status enable|disable]
```

引数

status	（任意）スクリプト実行ステータスを示すフラグ。この引数がイネーブルに設定されている場合、スクリプト実行がイネーブルにされます。この引数がディセーブルに設定されている場合、スクリプト実行がディセーブルにされます。

結果文字列

なし

_cerrno を設定

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

このエラーは、オペレーティング システムによってレポートされたエラーを意味します。エラーとともにレポートされる POSIX errno 値を使用して、オペレーティング システム エラーの原因を調べます。

```
(_cerr_sub_err = 14)    FH_ENOSUCHACTION  (unknown action type)
```

このエラーは、要求されたアクション コマンドが未知であることを示します。

```
(_cerr_sub_err = 52)    FH_ECONFIG  (configuration error)
```

このエラーは、設定エラーが発生したことを意味します。

action_setver_prior

絶対パスで示されるプロセスを以前のバージョンに戻します。

構文

```
action_setver_prior [path ?]
```

引数

path	（必須）プロセスの実行可能パス。

結果文字列

なし

_cerrno を設定

Yes

action_setnode

以降の EEM コマンドをそのノードで実行できるようにするために、指定されたノードに切り替えます。 次の EEM コマンドは action_setnode を使用してそのターゲット ノードを設定します。

- action_process
- sys_reqinfo_proc

- sys_reqinfo_proc_all
- sys_reqinfo_crash_history
- sys_reqinfo_proc_version

構文

```
action_setnode [node ?]
```

引数

node	（必須）ノードの名前。

結果文字列

なし

_cerrno を設定

Yes

action_syslog

メッセージを記録します。

構文

```
action_syslog [priority emerg|alert|crit|err|warning|notice|info|debug]
[msg ?]
```

引数

priority	（任意）action_syslog メッセージ ファシリティ レベル。この引数が指定されない場合、デフォルトのプライオリティは LOG_INFO です。
msg	（任意）記録されるメッセージ。

結果文字列

なし

_cerrno を設定

Yes

```
(_cerr_sub_err = 14)    FH_ENOSUCHACTION  (unknown action type)
```

このエラーは、要求されたアクション コマンドが未知であることを示します。

action_track_read

Embedded Event Manager（EEM）スクリプトがトリガーされるときにトラックされるオブジェクトの状態を読み取ります。

構文

```
action_track_read ?
```

引数

?（文字列を表す）	（必須）トラッキング対象オブジェクトの名前。

結果文字列

```
name {%s}
state {%s}
```

_cerrno を設定

Yes

```
FH_ENOTRACK
```

このエラーは、トラックされるオブジェクト名が見つからなかったことを意味します。

Embedded Event Manager ユーティリティ Tcl コマンド拡張

appl_read

Embedded Event Manager（EEM）アプリケーションの揮発性データを読み取ります。 この Tcl コマンド拡張では、EEM アプリケーションの揮発性データの読み取りがサポートされます。 EEM アプリケーションの揮発性データは、API をパブリッシュする EEM アプリケーションが使用される Cisco IOS XR ソフトウェア プロセスによってパブリッシュすることができます。 EEM アプリケーションの揮発性データは、EEM ポリシーによってパブリッシュできません。

（注） 現在、アプリケーションの揮発性データをパブリッシュする Cisco IOS XR ソフトウェア プロセスはありません。

構文

```
appl_read name ? length ?
```

引数

name	（必須）アプリケーションによってパブリッシュされる文字列データの名前。
length	（必須）読み取る文字列データの長さ。 1 ～ 4294967295 の範囲の整数である必要があります。

結果文字列

```
data %s
```

data は、読み取られる、アプリケーションによってパブリッシュされた文字列データです。

_cerrno を設定

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

このエラーは、オペレーティングシステムによってレポートされたエラーを意味します。エラーとともにレポートされる POSIX errno 値を使用して、オペレーティング システム エラーの原因を調べます。

```
(_cerr_sub_err =  7)    FH_ENOSUCHKEY  (could not find key)
```

このエラーは、アプリケーション イベント ディテクタ情報キーまたはその他の ID が見つからなかったことを意味します。

```
(_cerr_sub_err =  9)    FH_EMEMORY  (insufficient memory for request)
```

このエラーは、メモリの内部 EEM 要求に障害が発生したことを意味します。

appl_reqinfo

Embedded Event Manager（EEM）から、前に保存された情報が取得されます。 この Tcl コマンド拡張によって、一意のキーで前に保存された EEM からの情報の取得がサポートされます。これは、情報を取得するために指定する必要があります。 情報の取得によって、その情報が EEM から削除されることに、注意してください。 再度取得できるようにするには、再保存する必要があります。

構文

```
appl_reqinfo key ?
```

引数

key	（必須）データの文字列キー。

結果文字列

```
data %s
```
data は、取得されるアプリケーション文字列データです。

_cerrno を設定

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

このエラーは、オペレーティングシステムによってレポートされたエラーを意味します。エラーとともにレポートされる POSIX errno 値を使用して、オペレーティング システム エラーの原因を調べます。

```
(_cerr_sub_err =  7)    FH_ENOSUCHKEY  (could not find key)
```

このエラーは、アプリケーション イベント ディテクタ情報キーまたはその他の ID が見つからなかったことを意味します。

appl_setinfo

EEM に情報を保存します。 この Tcl コマンド拡張によって、同じポリシーまたは別のポリシーによって、後で取得できる EEM への情報の保存がサポートされます。 一意のキーを指定する必要があります。 このキーを使用すると、情報を後で取得することができます。

構文

```
appl_setinfo key ? data ?
```

引数

key	（必須）データの文字列キー。
data	（必須）保存するアプリケーション文字列データ。

結果文字列

なし

_cerrno を設定

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

このエラーは、オペレーティング システムによってレポートされたエラーを意味します。エラーとともにレポートされる POSIX errno 値を使用して、オペレーティング システム エラーの原因を調べます。

```
(_cerr_sub_err =  8)    FH_EDUPLICATEKEY  (duplicate appl info key)
```

このエラーは、アプリケーション イベント ディテクタ情報キーまたはその他の ID が重複していたことを意味します。

```
(_cerr_sub_err =  9)    FH_EMEMORY  (insufficient memory for request)
```

このエラーは、メモリの内部 EEM 要求に障害が発生したことを意味します。

```
(_cerr_sub_err = 34)    FH_EMAXLEN  (maximum length exceeded)
```

このエラーは、オブジェクト長またはオブジェクト数が、最大値を超えたことを意味します。

```
(_cerr_sub_err = 43)    FH_EBADLENGTH  (bad API length)
```

このエラーは、API メッセージ長が無効であったことを意味します。

counter_modify

カウンタの値を変更します。

構文

```
counter_modify event_id ? val ? op nop|set|inc|dec
```

引数

event_id	（必須）**register_counter** Tcl コマンド拡張によって返されるカウンタ イベント ID。 0 ～ 4294967295 の範囲の整数である必要があります。

val	（必須） • op が設定されている場合、この引数は、設定されるカウンタ値を表します。 • op が inc の場合、この引数は、カウンタを増やすために使用される値です。 • op が dec の場合、この引数は、カウンタを減らすために使用される値です。
op	（必須） • nop：現在のカウンタの値を取得します。 • set：カウンタの値を指定値に設定します。 • inc：カウンタの値を指定値分増やします。 • dec：カウンタの値を指定値分減らします。

結果文字列

```
val_remain %d
```

val_remain は、カウンタの現在の値です。

_cerrno を設定

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

このエラーは、オペレーティングシステムによってレポートされたエラーを意味します。エラーとともにレポートされる POSIX errno 値を使用して、オペレーティング システム エラーの原因を調べます。

```
(_cerr_sub_err = 11)    FH_ENOSUCHESID  (unknown event specification ID)
```

このエラーは、イベントが登録されたときか、またはイベント ディテクタの内部イベント構造が破損したときに、イベント指定 ID を照会できなかったことを意味します。

```
(_cerr_sub_err = 22)    FH_ENULLPTR  (event detector internal error - ptr is null)
```

このエラーは、内部EEMイベントディテクタポインタに値が含まれている必要があったときに、ヌルであったことを意味します。

```
(_cerr_sub_err = 30)    FH_ECTBADOPER  (bad counter threshold operator)
```

このエラーは、カウンタイベントディテクタの設定演算子または変更演算子が、無効であったことを意味します。

fts_get_stamp

最後にソフトウェアがブートされて以来の経過時間を返します。 この Tcl コマンド拡張を使用すると、配列 nsec nnnn に、ブート以降のナノ秒数が返されます。nnnn は、ナノ秒数です。

構文

```
fts_get_stamp
```

引数

なし

結果文字列

```
nsec %d
```
nsec は、ブート以降のナノ秒数です。

_cerrno を設定

No

register_counter

カウンタを登録し、カウンタ イベント ID を返します。 この Tcl コマンド拡張は、カウンタのパブリッシャによって使用され、イベント ID を使用してカウンタを操作する前に、この登録が実行されます。

構文

```
register_counter name ?
```

引数

name	（必須）操作されるカウンタの名前。

結果文字列

```
event_id %d
event_spec_id %d
```
event_id は、指定されたカウンタのカウンタ イベント ID です。**unregister_counter** Tcl コマンド拡張または **counter_modify** Tcl コマンド拡張によって、カウンタの操作に使用されます。 event_spec_id 引数は、指定されたカウンタのイベント指定 ID です。

_cerrno を設定

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

このエラーは、オペレーティング システムによってレポートされたエラーを意味します。エラーとともにレポートされる POSIX errno 値を使用して、オペレーティング システム エラーの原因を調べます。

```
(_cerr_sub_err =  4)    FH_EINITONCE  (Init() is not yet done, or done twice.)
```

このエラーは、EEM イベント ディテクタがその初期化を完了する前に、特定のイベントを登録する要求が行われたことを意味します。

```
(_cerr_sub_err =  6)    FH_EBADEVENTTYPE  (unknown EEM event type)
```

このエラーは、内部イベント指定で指定されたイベント タイプが無効であったことを意味します。

```
(_cerr_sub_err =  9)    FH_EMEMORY  (insufficient memory for request)
```

このエラーは、メモリの内部 EEM 要求に障害が発生したことを意味します。

```
(_cerr_sub_err = 10)    FH_ECORRUPT  (internal EEM API context is corrupt)
```

このエラーは、内部 EEM API コンテキスト構造が破損したことを意味します。

```
(_cerr_sub_err = 11)    FH_ENOSUCHESID  (unknown event specification ID)
```

このエラーは、イベントが登録されたときか、またはイベント ディテクタの内部イベント構造が破損したときに、イベント指定 ID を照会できなかったことを意味します。

```
(_cerr_sub_err = 12)    FH_ENOSUCHEID  (unknown event ID)
```

このエラーは、イベントが登録されたときか、またはイベント ディテクタの内部イベント構造が破損したときに、イベント ID を照会できなかったことを意味します。

```
(_cerr_sub_err = 16)    FH_EBADFMPPTR  (bad ptr to fh_p data structure)
```

このエラーは、各 EEM API コールで使用されるコンテキスト ポインタが不正確であったことを意味します。

```
(_cerr_sub_err = 17)    FH_EBADADDRESS  (bad API control block address)
```

このエラーは、EEM API に渡された制御ブロック アドレスが不正確であったことを意味します。

```
(_cerr_sub_err = 22)    FH_ENULLPTR  (event detector internal error - ptr is null)
```

このエラーは、内部 EEM イベント ディテクタ ポインタに値が含まれている必要があったときに、ヌルであったことを意味します。

```
(_cerr_sub_err = 25)    FH_ESUBSEXCEED  (number of subscribers exceeded)
```

このエラーは、タイマーまたはカウンタのサブスクライバの数が、最大値を超えたことを意味します。

```
(_cerr_sub_err = 26)    FH_ESUBSIDXINV  (invalid subscriber index)
```

これは、サブスクライバの索引が無効であったことを意味します。

```
(_cerr_sub_err = 54)    FH_EFDUNAVAIL  (connection to event detector unavailable)
```

このエラーは、イベント ディテクタが使用できなかったことを意味します。

```
(_cerr_sub_err = 56)    FH_EFDCONNERR  (event detector connection error)
```

このエラーは、この要求を処理するEEMイベントディテクタは使用できないことを意味します。

register_timer

タイマーを登録し、タイマー イベント ID を返します。 この Tcl コマンド拡張は、カウンタのパブリッシャによって使用され、パブリッシャまたはサブスクライバとしての登録に、**event_register_timer** コマンド拡張が使用されなかった場合に、イベントIDを使用してタイマーを操作する前に、この登録が実行されます。

構文

```
register_timer watchdog|countdown|absolute|cron name ?
```

引数

name	（必須）操作されるタイマーの名前。

結果文字列

```
event_id %u
```

event_id は、指定されたタイマーのタイマー イベント ID です（**timer_arm** コマンド拡張または **timer_cancel** コマンド拡張によってタイマーの操作に使用できます）。

_cerrno を設定

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

このエラーは、オペレーティングシステムによってレポートされたエラーを意味します。エラーとともにレポートされる POSIX errno 値を使用して、オペレーティング システム エラーの原因を調べます。

```
(_cerr_sub_err =  4)    FH_EINITONCE  (Init() is not yet done, or done twice.)
```

このエラーは、EEMイベントディテクタがその初期化を完了する前に、特定のイベントを登録する要求が行われたことを意味します。

```
(_cerr_sub_err =  6)    FH_EBADEVENTTYPE  (unknown EEM event type)
```

このエラーは、内部イベント指定で指定されたイベント タイプが無効であったことを意味します。

```
(_cerr_sub_err =  9)    FH_EMEMORY  (insufficient memory for request)
```

このエラーは、メモリの内部 EEM 要求に障害が発生したことを意味します。

```
(_cerr_sub_err = 10)    FH_ECORRUPT  (internal EEM API context is corrupt)
```

このエラーは、内部 EEM API コンテキスト構造が破損したことを意味します。

```
(_cerr_sub_err = 11)    FH_ENOSUCHESID  (unknown event specification ID)
```

このエラーは、イベントが登録されたときか、またはイベント ディテクタの内部イベント構造が破損したときに、イベント指定 ID を照会できなかったことを意味します。

```
(_cerr_sub_err = 16)    FH_EBADFMPPTR  (bad ptr to fh_p data structure)
```

このエラーは、各 EEM API コールで使用されるコンテキスト ポインタが不正確であったことを意味します。

```
(_cerr_sub_err = 17)    FH_EBADADDRESS  (bad API control block address)
```

このエラーは、EEM API に渡された制御ブロック アドレスが不正確であったことを意味します。

```
(_cerr_sub_err = 22)    FH_ENULLPTR  (event detector internal error - ptr is null)
```

このエラーは、内部EEMイベントディテクタ ポインタに値が含まれている必要があったときに、ヌルであったことを意味します。

```
(_cerr_sub_err = 25)    FH_ESUBSEXCEED  (number of subscribers exceeded)
```

このエラーは、タイマーまたはカウンタのサブスクライバの数が、最大値を超えたことを意味します。

```
(_cerr_sub_err = 26)    FH_ESUBSIDXINV  (invalid subscriber index)
```

これは、サブスクライバの索引が無効であったことを意味します。

```
(_cerr_sub_err = 54)    FH_EFDUNAVAIL  (connection to event detector unavailable)
```

このエラーは、イベント ディテクタが使用できなかったことを意味します。

```
(_cerr_sub_err = 56)    FH_EFDCONNERR  (event detector connection error)
```

このエラーは、この要求を処理するEEMイベントディテクタは使用できないことを意味します。

timer_arm

タイマーを搭載します。 タイプは、CRON、ウォッチドッグ、カウントダウン、または絶対の場合があります。

構文

```
timer_arm event_id ? cron_entry ?|time ?
```

引数

event_id	（必須）**register_timer** Tcl コマンド拡張によって返されるタイマー イベント ID。 0 ～ 4294967295 の範囲の整数である必要があります。
cron_entry	（必須）タイマー タイプが CRON の場合に存在する必要があります。他のタイプのタイマーの場合には、存在させることはできません。CRON タイマー指定によって、CRON テーブル エントリの形式が使用されます。
time	（必須）タイマー タイプが CRON ではない場合に存在する必要があります。 タイマー タイプが CRON の場合には、存在できません。ウォッチドッグ タイマーおよびカウントダウン タイマーでは、タイマーの期限が切れるまでの秒数およびミリ秒数です。絶対タイマーでは、期限切れ時刻のカレンダー時間です（SSSSSSSSSS[.MMM] 形式で指定します。SSSSSSSSSS は、0 ～ 4294967295 の秒数を表す整数で、MMM は 0 ～ 999 のミリ秒数を表す整数である必要があります）。 期限の絶対日付は、1970 年 1 月 1 日以降の秒およびミリ秒の単位での数です。 指定された日付がすでに過ぎた場合、タイマーの期限はただちに切れます。

結果文字列

```
sec_remain %ld msec_remain %ld
```

sec_remain および msec_remain は、タイマーの次の期限切れまでの残り時間です。

（注） タイマー タイプが CRON の場合、sec_remain 引数および msec_remain 引数には 0 が返されます。

_cerrno を設定

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

このエラーは、オペレーティング システムによってレポートされたエラーを意味します。エラーとともにレポートされる POSIX errno 値を使用して、オペレーティング システム エラーの原因を調べます。

```
(_cerr_sub_err =  6)    FH_EBADEVENTTYPE  (unknown EEM event type)
```

このエラーは、内部イベント指定で指定されたイベント タイプが無効であったことを意味します。

```
(_cerr_sub_err =  9)    FH_EMEMORY  (insufficient memory for request)
```

このエラーは、メモリの内部 EEM 要求に障害が発生したことを意味します。

```
(_cerr_sub_err = 11)    FH_ENOSUCHESID  (unknown event specification ID)
```

このエラーは、イベントが登録されたときか、またはイベント ディテクタの内部イベント構造が破損したときに、イベント指定 ID を照会できなかったことを意味します。

```
(_cerr_sub_err = 12)    FH_ENOSUCHEID  (unknown event ID)
```

このエラーは、イベントが登録されたときか、またはイベント ディテクタの内部イベント構造が破損したときに、イベント ID を照会できなかったことを意味します。

```
(_cerr_sub_err = 22)    FH_ENULLPTR  (event detector internal error - ptr is null)
```

このエラーは、内部EEMイベント ディテクタ ポインタに値が含まれている必要があったときに、ヌルであったことを意味します。

```
(_cerr_sub_err = 27)    FH_ETMDELAYZR  (zero delay time)
```

このエラーは、タイマーの搭載に指定された時間がゼロであったことを意味します。

```
(_cerr_sub_err = 42)    FH_ENOTREGISTERED  (request for event spec that is unregistered)
```

このエラーは、イベント検出が登録できなかったことを意味します。

```
(_cerr_sub_err = 54)    FH_EFDUNAVAIL  (connection to event detector unavailable)
```

このエラーは、イベント ディテクタが使用できなかったことを意味します。

```
(_cerr_sub_err = 56)    FH_EFDCONNERR  (event detector connection error)
```

このエラーは、この要求を処理するEEMイベントディテクタは使用できないことを意味します。

timer_cancel

タイマーを取り消します。

構文

```
timer_cancel event_id ?
```

引数

event_id	（必須）**register_timer** Tcl コマンド拡張によって返されるタイマー イベント ID。 0 ～ 4294967295 の範囲の整数である必要があります。

結果文字列

```
sec_remain %ld msec_remain %ld
```

sec_remain および msec_remain は、タイマーの次の期限切れまでの残り時間です。

（注） タイマー タイプが CRON の場合、sec_remain および msec_remain には 0 が返されます。

_cerrno を設定

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

このエラーは、オペレーティングシステムによってレポートされたエラーを意味します。エラーとともにレポートされる POSIX errno 値を使用して、オペレーティング システム エラーの原因を調べます。

```
(_cerr_sub_err =  6)    FH_EBADEVENTTYPE  (unknown EEM event type)
```

このエラーは、内部イベント指定で指定されたイベント タイプが無効であったことを意味します。

```
(_cerr_sub_err =  7)    FH_ENOSUCHKEY  (could not find key)
```

このエラーは、アプリケーション イベント ディテクタ情報キーまたはその他の ID が見つからなかったことを意味します。

```
(_cerr_sub_err = 11)    FH_ENOSUCHESID  (unknown event specification ID)
```

このエラーは、イベントが登録されたときか、またはイベント ディテクタの内部イベント構造が破損したときに、イベント指定 ID を照会できなかったことを意味します。

```
(_cerr_sub_err = 12)    FH_ENOSUCHEID  (unknown event ID)
```

このエラーは、イベントが登録されたときか、またはイベント ディテクタの内部イベント構造が破損したときに、イベント ID を照会できなかったことを意味します。

```
(_cerr_sub_err = 22)    FH_ENULLPTR  (event detector internal error - ptr is null)
```

このエラーは、内部EEMイベントディテクタポインタに値が含まれている必要があったときに、ヌルであったことを意味します。

```
(_cerr_sub_err = 54)    FH_EFDUNAVAIL  (connection to event detector unavailable)
```

このエラーは、イベント ディテクタが使用できなかったことを意味します。

```
(_cerr_sub_err = 56)    FH_EFDCONNERR  (event detector connection error)
```

このエラーは、この要求を処理するEEMイベントディテクタは使用できないことを意味します。

unregister_counter

カウンタの登録を解除します。 この Tcl コマンド拡張は、カウンタのパブリッシャによって使用され、**register_counter** コマンド拡張で前に登録されたカウンタの登録を解除します。

構文

```
unregister_counter event_id ? event_spec_id ?
```

引数

event_id	（必須）**register_counter** コマンド拡張によって返されるカウンタ イベント ID。 0 ～ 4294967295 の範囲の整数である必要があります。
event_spec_id	（必須）**register_counter** コマンド拡張によって返された、指定されたカウンタのカウンタ イベント指定 ID。 0 ～ 4294967295 の範囲の整数である必要があります。

結果文字列

なし

_cerrno を設定

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

このエラーは、オペレーティングシステムによってレポートされたエラーを意味します。エラーとともにレポートされる POSIX errno 値を使用して、オペレーティング システム エラーの原因を調べます。

```
(_cerr_sub_err =  9)    FH_EMEMORY  (insufficient memory for request)
```

このエラーは、メモリの内部 EEM 要求に障害が発生したことを意味します。

```
(_cerr_sub_err = 11)    FH_ENOSUCHESID  (unknown event specification ID)
```

このエラーは、イベントが登録されたときか、またはイベント ディテクタの内部イベント構造が破損したときに、イベント指定 ID を照会できなかったことを意味します。

```
(_cerr_sub_err = 22)    FH_ENULLPTR  (event detector internal error - ptr is null)
```

このエラーは、内部EEMイベントディテクタ ポインタに値が含まれている必要があったときに、ヌルであったことを意味します。

```
(_cerr_sub_err = 26)    FH_ESUBSIDXINV  (invalid subscriber index)
```

これは、サブスクライバの索引が無効であったことを意味します。

```
(_cerr_sub_err = 54)    FH_EFDUNAVAIL  (connection to event detector unavailable)
```

このエラーは、イベント ディテクタが使用できなかったことを意味します。

```
(_cerr_sub_err = 56)    FH_EFDCONNERR  (event detector connection error)
```

このエラーは、この要求を処理するEEMイベントディテクタは使用できないことを意味します。

Embedded Event Manager システム情報 Tcl コマンド拡張

（注） すべての EEM システム情報コマンド **sys_reqinfo _xxx** には、**yes** に設定された「_cerrno を設定」セクションがあります。

sys_reqinfo_cpu_all

指定された期間で、指定された順序で、上位プロセスのCPU使用率（POSIXプロセスとIOSプロセスの両方）を問い合わせます。 この Tcl コマンド拡張は、ソフトウェア モジュール方式イメージでのみサポートされます。

構文

```
sys_reqinfo_cpu_all order cpu_used [sec ?] [msec ?] [num ?]
```

引数

order	（必須）プロセスの CPU 使用率のソートに使用される順序。
cpu_used	（必須）指定されたウィンドウの、CPU使用率の平均が、降順でソートされるよう、指定します。
secmsec	（任意）CPU使用率の平均が計算される、秒単位およびミリ秒単位での時間。 0 から 4294967295 の範囲の整数である必要があります。 指定されない場合か、または、sec と msec の両方が 0 と指定される場合、最新の CPU サンプルが使用されます。
num	（任意）表示される、ソートされたプロセスのリストの上位からのエントリの数。 1 ～ 4294967295 の範囲の整数である必要があります。 デフォルト値は 5 です。

結果文字列

```
rec_list {{process CPU info string 0},{process CPU info string 1}, ...}
```

各プロセスの CPU 情報文字列は、次のとおりです。

```
pid %u name {%s} cpu_used %u
```

rec_list	プロセス CPU 情報リストの開始をマークします。
pid	プロセス ID。
name	プロセス名。

cpu_used	sec と msec が、ゼロよりも大きい数で指定される場合、平均パーセンテージは、指定された時間のプロセスCPU使用率から計算されるよう、指定します。 sec と msec が、両方ともゼロか、または指定されない場合。平均パーセンテージは、最新のサンプルのプロセス CPU 使用率から計算されます。

_cerrno を設定

Yes

sys_reqinfo_crash_history

クラッシュしたすべてのプロセスのプロセス情報を問い合わせます。 この Tcl コマンド拡張は、ソフトウェア モジュール方式イメージでのみサポートされます。

構文

```
sys_reqinfo_crash_history
```

引数

なし

結果文字列

```
rec_list {{crash info string 0},{crash info string 1}, ...}

Where each crash info string is:

job_id %u name {%s} respawn_count %u fail_count %u dump_count %u
inst_id %d exit_status 0x%x exit_type %d proc_state {%s} component_id 0x%x
crash_time_sec %ld crash_time_msec %ld
```

job_id	システムマネージャによってプロセスに割り当てられるジョブ ID。 1 ～ 4294967295 の整数。
name	プロセス名。
respawn_count	プロセスの再起動の合計回数。
fail_count	プロセスの再起動試行の回数。プロセスが正常に再起動されると、このカウントはゼロにリセットされます。
dump_count	実行されたコア ダンプの回数。

inst_id	プロセス インスタンス ID。
exit_status	プロセスの最後の終了ステータス。
exit_type	最後の終了タイプ。
proc_state	Sysmgr プロセスの状態。 error、forced_stop、hold、init、ready_to_run、run、run_rnode、stop、waitEOltimer、wait_rnode、wait_spawntimer、wait_tpl の 1 つです。
component_id	プロセスが属するコンポーネントのコンポーネントIDに割り当てられているバージョンマネージャ。
crash_time_sec crash_time_msec	1970 年 1 月 1 日以降の秒およびミリ秒の単位で、プロセスがクラッシュした最後の時刻を表します。

_cerrno を設定

Yes

sys_reqinfo_mem_all

指定された期間で、指定された順序で、上位プロセスのメモリの使用状況（POSIX と IOS の両方）を問い合わせます。 この Tcl コマンド拡張は、ソフトウェア モジュール方式イメージでのみサポートされます。

構文

```
sys_reqinfo_mem_all order allocates|increase|used [sec ?] [msec ?] [num ?]
```

引数

order	（必須）プロセスのメモリの使用状況のソートに使用される順序。
allocates	（必須）指定された時間ウィンドウの期間に、メモリの使用状況が、プロセス割り当ての数によって降順でソートされるよう、指定します。

increase	（必須）指定された時間ウィンドウの期間に、メモリの使用状況が、プロセスで増えたメモリのパーセンテージによって降順でソートされるよう、指定します。
used	（必須）メモリが、プロセスによって使用される現在のメモリによってソートされるよう、指定します。
secmsec	（任意）プロセスでのメモリの使用状況が計算される、秒単位およびミリ秒単位での時間。 0 から 4294967295 の範囲の整数である必要があります。 sec と msec の両方が指定され、ゼロではない場合、割り当て数は、該当する時間で収集された最も古いサンプルと最新のサンプルでの、割り当て数の差です。パーセンテージは、該当する時間で収集された最も古いサンプルと最新のサンプルとの、パーセンテージの差分として計算されます。指定されない場合か、または、sec と msec の両方が 0 と指定される場合、収集された最初のサンプルが、最も古いサンプルとして使用されます。つまり、時間は、起動から現在時までの時間で設定されます。
num	（任意）表示される、ソートされたプロセスのリストの上位からのエントリの数。 1 ～ 4294967295 の範囲の整数である必要があります。 デフォルト値は 5 です。

結果文字列

```
rec_list {{process mem info string 0},{process mem info string 1}, ...}
```

各プロセスのメモリ情報文字列は、次のとおりです。

```
pid %u name {%s} delta_allocs %d initial_alloc %u current_alloc %u percent_increase %d
```

rec_list	プロセスのメモリの使用状況情報リストの開始をマークします。
pid	プロセス ID。
name	プロセス名。

delta_allocs	該当する期間で収集された、最も古いサンプルと最新のサンプルでの、割り当て数の差として、差を指定します。
initial_alloc	時間の開始時にプロセスによって使用される、キロバイト単位での、メモリの容量を指定します。
current_alloc	プロセスによって使用される、キロバイト単位での、メモリの容量を指定します。
percent_increase	該当する時間で収集された最も古いサンプルと最新のサンプルとの、使用メモリのパーセンテージの差分を指定します。パーセンテージの差は、current_alloc から initial_alloc の 100 を差し引いた数として、および、initial_alloc で割った数として、表すことができます。

_cerrno を設定

Yes

sys_reqinfo_proc

1 つの POSIX プロセスに関する情報を問い合わせます。 この Tcl コマンド拡張は、ソフトウェアモジュール方式イメージでのみサポートされます。

構文

```
sys_reqinfo_proc job_id ?
```

引数

job_id	（必須）システムマネージャによってプロセスに割り当てられるジョブ ID。 1 ～ 4294967295 の範囲の整数である必要があります。

結果文字列

```
job_id %u component_id 0x%x name {%s} helper_name {%s} helper_path {%s} path {%s}
node_name {%s} is_respawn %u is_mandatory %u is_hold %u dump_option %d
max_dump_count %u respawn_count %u fail_count %u dump_count %u
last_respawn_sec %ld last_respawn_msec %ld inst_id %u proc_state %s
level %d exit_status 0x%x exit_type %d
```

job_id	システムマネージャによってプロセスに割り当てられるジョブ ID。 1 ～ 4294967295 の整数。
component_id	プロセスが属するコンポーネントのコンポーネントIDに割り当てられているバージョンマネージャ。
name	プロセス名。
helper_name	ヘルパー プロセスの名前。
helper_path	ヘルパー プロセスの実行可能パス。
path	プロセスの実行可能パス。
node_name	プロセスが属するノードのノード名に割り当てられているシステム マネージャ。
is_respawn	プロセスが再生成できることを指定するフラグ。
is_mandatory	プロセスが実行され続ける必要があることを指定するフラグ。
is_hold	API によって呼び出されるまでプロセスが再生成されることを指定するフラグ。
dump_option	コア ダンプのオプション。
max_dump_count	許可されるコア ダンプの最大数。
respawn_count	プロセスの再起動の合計回数。
fail_count	プロセスの再起動試行の回数。プロセスが正常に再起動されると、このカウントはゼロにリセットされます。
dump_count	実行されたコア ダンプの回数。
last_respawn_seclast_respawn_msec	1970 年 1 月 1 日以降の POSIX タイマー ユニットでの秒およびミリ秒の単位で、プロセスが開始された最後の時刻を表します。
inst_id	プロセス インスタンス ID。

proc_state	Sysmgr プロセスの状態。 error、forced_stop、hold、init、ready_to_run、run、run_rnode、stop、waitEOltimer、wait_rnode、wait_spawntimer、wait_tpl の 1 つです。
level	プロセス実行レベル。
exit_status	プロセスの最後の終了ステータス。
exit_type	最後の終了タイプ。

_cerrno を設定

Yes

sys_reqinfo_proc_all

すべての POSIX プロセスの情報を問い合わせます。 この Tcl コマンド拡張は、ソフトウェア モジュール方式イメージでのみサポートされます。

構文

```
sys_reqinfo_proc_all
```

引数

なし

結果文字列

```
rec_list {{process info string 0}, {process info string 1},...}
```
各プロセスの情報文字列は、**sysreq_info_proc** Tcl コマンド拡張の結果文字列と同じです。

_cerrno を設定

Yes

sys_reqinfo_proc_version

指定したプロセスのバージョンを問い合わせます。

構文

```
sys_reqinfo_proc_version [job_id ?]
```

引数

job_id	（必須）システムマネージャによってプロセスに割り当てられるジョブ ID。 1 ～ 2147483647 の範囲の整数であることが必要です。

結果文字列

```
version_id %02d.%02d.%04d
```

version_id は、プロセスのバージョン番号が割り当てられたバージョン マネージャです。

_cerrno を設定

Yes

sys_reqinfo_routername

ルータ名を問い合わせます。

構文

```
sys_reqinfo_routername
```

引数

なし

結果文字列

```
routername %s
```

routername は、ルータの名前です。

_cerrno を設定

Yes

sys_reqinfo_syslog_freq

すべての Syslog イベントの頻度情報を問い合わせます。

構文

```
sys_reqinfo_syslog_freq
```

引数

なし

結果文字列

```
rec_list {{event frequency string 0}, {log freq str 1}, ...}
```

各イベントの頻度の文字列は、次のとおりです。

```
time_sec %ld time_msec %ld match_count %u raise_count %u occurs %u
period_sec %ld period_msec %ld pattern {%s}
```

time_sectime_msec	1970 年 1 月 1 日以降の POSIX タイマー ユニットでの秒およびミリ秒の単位で、最後のイベントが発生した時刻を表します。
match_count	イベントの登録以降、この Syslog イベント指定によって指定されたパターンが、Syslog メッセージによって照会される回数。
raise_count	この Syslog イベントが発生した回数。
occurs	イベントを発生させるために必要な発生回数。指定されない場合、イベントは 1 回目から発生します。
period_secperiod_msec	イベントを発生させるには、発生回数が POSIX タイマーユニットのこの数以内である必要があります。この引数が指定されない場合、時間のチェックは適用されません。
pattern	Syslog メッセージのパターン マッチの実行に使用される正規表現。

_cerrno を設定

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

このエラーは、オペレーティング システムによってレポートされたエラーを意味します。エラーとともにレポートされる POSIX errno 値を使用して、オペレーティング システム エラーの原因を調べます。

```
(_cerr_sub_err =  9)    FH_EMEMORY  (insufficient memory for request)
```

このエラーは、メモリの内部 EEM 要求に障害が発生したことを意味します。

```
(_cerr_sub_err = 22)    FH_ENULLPTR  (event detector internal error - ptr is null)
```

このエラーは、内部EEMイベントディテクタポインタに値が含まれている必要があったときに、ヌルであったことを意味します。

```
(_cerr_sub_err = 45)    FH_ESEQNUM  (sequence or workset number out of sync)
```

このエラーは、イベントディテクタシーケンスまたは作業セット番号が無効であったことを意味します。

```
(_cerr_sub_err = 46)    FH_EREGEMPTY  (registration list is empty)
```

このエラーは、イベントディテクタ登録リストが空であったことを意味します。

```
(_cerr_sub_err = 54)    FH_EFDUNAVAIL  (connection to event detector unavailable)
```

このエラーは、イベントディテクタが使用できなかったことを意味します。

sys_reqinfo_syslog_history

指定された Syslog メッセージの履歴を問い合わせます。

構文

```
sys_reqinfo_syslog_history
```

引数

なし

結果文字列

```
rec_list {{log hist string 0}, {log hist str 1}, ...}
```

各記録の履歴の文字列は、次のとおりです。

```
time_sec %ld time_msec %ld msg {%s}
```

time_sec time_msec	1970 年 1 月 1 日以降の秒およびミリ秒の単位で、メッセージが記録された時刻を表します。
msg	Syslog メッセージ。

_cerrno を設定

Yes

```
(_cerr_sub_err =  2)    FH_ESYSERR  (generic/unknown error from OS/system)
```

このエラーは、オペレーティングシステムによってレポートされたエラーを意味します。エラーとともにレポートされる POSIX errno 値を使用して、オペレーティング システム エラーの原因を調べます。

```
(_cerr_sub_err = 22)    FH_ENULLPTR  (event detector internal error - ptr is null)
```

このエラーは、内部EEMイベントディテクタポインタに値が含まれている必要があったときに、ヌルであったことを意味します。

```
(_cerr_sub_err = 44)    FH_EHISTEMPTY  (history list is empty)
```

このエラーは、履歴のリストが空であったことを意味します。

```
(_cerr_sub_err = 45)    FH_ESEQNUM  (sequence or workset number out of sync)
```

このエラーは、イベントディテクタシーケンスまたは作業セット番号が無効であったことを意味します。

```
(_cerr_sub_err = 54)    FH_EFDUNAVAIL  (connection to event detector unavailable)
```

このエラーは、イベント ディテクタが使用できなかったことを意味します。

sys_reqinfo_stat

名前で指定された統計エントリの値と、必要に応じて第 1 修飾子および第 2 修飾子を問い合わせます。

構文

```
sys_reqinfo_stat [name ?][mod1 ?][mod2 ?]
```

引数

name	（必須）統計データ要素名。
mod_1	（任意）統計データ要素修飾子 1。
mod_2	（任意）統計データ要素修飾子 2。

結果文字列

```
name %s value %s
```

name	統計データ要素名。
value	統計データ要素の値文字列。

_cerrno を設定

Yes

sys_reqinfo_snmp

簡易ネットワーク管理プロトコル（SNMP）オブジェクト ID によって指定されたエンティティの値を問い合わせます。

構文

```
sys_reqinfo_snmp oid ? get_type exact|next
```

引数

oid	（必須）ドット付き表記での SNMP OID（たとえば、1.3.6.1.2.1.2.1.0）。
get_type	（必須）指定された OID に適用する必要がある SNMP 取得操作のタイプ。 get_type が「exact」の場合、指定された OID の値が取得されます。get_type が「next」の場合、指定された OID の辞書順での後続値が取得されます。

結果文字列

```
oid {%s} value {%s}
```

oid	SNMP OID。
value	割り当てられた SNMP データ エレメントの値文字列。

sys_reqinfo_snmp_trap

このコマンドは、トラップを送信するために使用されます。

構文

```
sys_reqinfo_snmp_trap enterprise_oid ent-oid generic_trapnum gen-trapnum specific_trapnum
spe-trapnum
 trap_oid oid trap_var varname
```

- トラップのエンタープライズ OID を指定するには ***enterprise_oid*** の引数を使用します。
- トラップの汎用トラップ数を指定するには ***generic_trapnum*** の引数を使用します。
- トラップの特定のトラップ番号を指定するには ***specific_trapnum*** の引数を使用します。

- 送信するトラップの OID を指定するには ***trap_oid*** の引数を使用します。
- 送信する OID の変数を指定するには ***trap_var*** の引数を使用します。

例

sys_reqinfo_snmp_trap **enterprise_oid** 1.3.6.1.4.1.9.9.41.2 **generic_trapnum** 6 **specific_trapnum** 1 **trap_oid** 1.3.6.1.4.1.9.9.41.2.0.1 **trap_var** var1

sys_reqinfo_snmp_trapvar

このコマンドは、トラップ変数に指定された OID と値の配列を設定するために使用されます。IOS と同様に、トラップ変数には 10 の複数の OID と値のリストを含めることができます。

構文

```
sys_reqinfo_snmp_trapvar var varname oid oid int|uint|counter|gauge|octet|string|ipv4  value
```

- トラップ変数の名前を指定するには、***var*** 引数を使用します。
- トラップの OID を指定するには、***oid*** 引数を使用します。

例

sys_reqinfo_snmp_trapvar **var** var1 **oid** 1.3.6.1.4.1.9.9.41.1.2.3.1.3 **int** 4

SMTP ライブラリのコマンド拡張

すべてのシンプル メール転送プロトコル（SMTP）ライブラリ コマンドは、::cisco::lib名前空間に属します。

このライブラリを使用するには、ユーザは、電子メールテンプレートファイルを用意する必要があります。電子メール サービスと電子メールテキストを、**event manager environment** Cisco IOS XR ソフトウェア コマンドライン インターフェイス（CLI）コンフィギュレーション コマンドを使用して設定できるよう、テンプレート ファイルに Tcl グローバル変数を含めることができます。電子メール テンプレート ファイルでグローバル変数を置き換え、設定された電子メール サーバを使用して、設定された To アドレス、CC アドレス、From アドレス、および Subject 行プロパティで必要な電子メール コンテキストを送信するには、このライブラリにあるコマンドを使用します。

電子メール テンプレート

電子メール テンプレート ファイルの形式は、次のとおりです。

```
Mailservername:<space><the list of candidate SMTP server addresses>
From:<space><the e-mail address of sender>
To:<space><the list of e-mail addresses of recipients>
Cc:<space><the list of e-mail addresses that the e-mail will be copied to>
Subject:<subject line>
<a blank line>
<body>
```

（注） テンプレートには、通常、設定される Tcl グローバル変数が含まれています。

次に、サンプル電子メール テンプレート ファイルを示します。

```
Mailservername: $_email_server
From: $_email_from
To: $_email_to
Cc: $_email_cc
Subject: From router $routername: Process terminated

process name: $process_name
subsystem: $sub_system
exit status: $exit_status
respawn count: $respawn_count
```

エクスポートされる Tcl コマンド拡張

smtp_send_email

電子メールテンプレートファイルのテキストが、すべてのグローバル変数ですでに置き換えられている場合、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。 電子メール テンプレートによって、候補メール サーバのアドレス、To アドレス、CC アドレス、From アドレス、件名の行、および電子メールの本文が指定されます。

（注） ライブラリが、リストにあるサーバの 1 つに接続できるまで、サーバへの接続が、1 つ 1 つ試行されるよう、候補電子メール サーバのリストを用意できます。

構文

```
smtp_send_email text
```

引数

text	（必須）すべてのグローバル変数ですでに置き換えられた、電子メール テンプレート ファイルのテキスト。

結果文字列

なし

_cerrno を設定

- 1 行目の形式が間違っている：Mailservername：サーバ名のリスト。
- 2 行目の形式が間違っている：From：送信元アドレス。

- 3 行目の形式が間違っている：To：送信先アドレスのリスト。
- 4 行目の形式が間違っている：CC：コピー送信先アドレスのリスト。
- メール サーバへの接続エラー：リモート サーバによって $sock が閉じられている（$sock はメール サーバに開かれているソケットの名前）。
- メール サーバへの接続エラー：$sock 応答コードが service ready greeting ではなく $k である（$sock はメール サーバに開かれているソケットの名前、$k は $sock の応答コード）。
- メール サーバへの接続エラー：すべてのメール サーバ候補に接続できない。
- メール サーバからの接続解除エラー：リモート サーバによって $sock が閉じられている（$sock はメール サーバに開かれているソケットの名前）。

サンプル スクリプト

電子メール テンプレートですべての必要なグローバル変数が定義された後には、次のようになります。

```
if [catch {smtp_subst [file join $tcl_library email_template_sm]} result] {
    puts stderr $result
    exit 1
}
if [catch {smtp_send_email $result} result] {
    puts stderr $result
    exit 1
}
```

smtp_subst

電子メール テンプレート ファイル e-mail_template の場合、ファイルにある各グローバル変数を、そのユーザ定義値によって置き換えます。 置換後に、ファイルのテキストを返します。

構文

```
smtp_subst e-mail_template
```

引数

e-mail_template	（必須）グローバル変数が、ユーザ定義値によって置き換えられる必要がある、電子メール テンプレート ファイルの名前。 ファイル名の例は /disk0://example.template で、スロット 0 の ATA フラッシュ ディスクの上位レベル ディレクトリにある example.template という名前のファイルを表します。

結果文字列

すべてのグローバル変数で置き換えられた、電子メール テンプレート ファイルのテキスト。

_cerrno を設定

- 電子メール テンプレート ファイルを開けられない。
- 電子メール テンプレート ファイルを閉じられない。

CLI ライブラリのコマンド拡張

すべてのコマンドライン インターフェイス（CLI）ライブラリ コマンド拡張は、::cisco::eem 名前空間に属します。

このライブラリによって、ユーザに対し、CLI コマンドを実行し、Tcl でコマンドの出力を取得する機能が用意されます。 コマンドが exec によって実行され、コマンドの出力が読み戻されるようにするため、ユーザは、このライブラリでコマンドを使用して、exec を生成し、それに対して仮想端末チャネルをオープンし、コマンドを記述してチャネルに対して実行できます。

CLI コマンドには、対話式コマンドと非対話式コマンドの、2 つのタイプがあります。

対話式コマンドでは、コマンドの入力後、ルータによって、異なるユーザ オプションが質問される「Q&A」フェーズがあり、ユーザは、各質問に対する答えを入力する必要があります。 すべての質問が適切に答えられた後、ユーザのオプションに従って、完了するまでコマンドが実行されます。

非対話式コマンドでは、コマンドが一度入力されると、コマンドが完了まで実行されます。 EEM スクリプトで異なるタイプのコマンドを実行するには、異なる CLI ライブラリ コマンド シーケンスを使用する必要があり、これは、CLI ライブラリを使用した非対話式コマンドの実行，（227 ページ） および CLI ライブラリを使用した対話式コマンドの実行，（228 ページ） で説明します。

エクスポートされる Tcl コマンド拡張

cli_close

exec プロセスをクローズし、コマンドライン インターフェイス（CLI）に接続された、VTY および指定されたチャネル ハンドラをリリースします。

構文

```
cli_close fd tty_id
```

引数

fd	（必須）CLI チャネル ハンドラ。

tty_id	（必須）**cli_open** コマンド拡張から返された TTY ID。

結果文字列

なし

_cerrno を設定

チャネルをクローズできない。

cli_exec

指定されたチャネル ハンドラにコマンドを記述し、コマンドを実行します。次に、チャネルからコマンドの出力を読み取り、出力を返します。

構文

```
cli_exec fd cmd
```

引数

fd	（必須）コマンドライン インターフェイス（CLI）チャネル ハンドラ。
cmd	（必須）実行する CLI コマンド。

結果文字列

実行された CLI コマンドの出力。

_cerrno を設定

チャネルを読み取れない。

cli_get_ttyname

該当する TTY ID の実際と疑似の tty の名前を返します。

構文

```
cli_get_ttyname tty_id
```

引数

tty_id	（必須）**cli_open** コマンド拡張から返された TTY ID。

結果文字列

```
pty %s tty %s
```

_cerrno を設定

なし

cli_open

vty を割り当て、EXEC コマンドライン インターフェイス（CLI）セッションを作成し、vty をチャネル ハンドラに接続します。 チャネル ハンドラを含む配列を返します。

（注） **cli_open** を呼び出すたびに Cisco IOS XR ソフトウェア EXEC セッションが開始され、Cisco IOS XR ソフトウェア vty が割り当てられます。 vty は、cli_close ルーチンが呼び出されるまで、使用中のままです。 vty は、**line vty vty-pool** CLI コンフィギュレーション コマンドを使用して設定された vty のプールから割り当てられます。 使用可能な vty が 2 つ以下の場合、cli_open ルーチンは失敗し、残りの vty は Telnet で使用できるよう保存されることに注意してください。

構文

```
cli_open
```

引数

なし

結果文字列

```
"tty_id {%s} pty {%d} tty {%d} fd {%d}"
```

イベント タイプ	説明
tty_id	TTY ID。
pty	PTY デバイス名。
tty	TTY デバイス名。

イベント タイプ	説明
fd	CLI チャネル ハンドラ。

_cerrno を設定

- EXEC の pty を取得できない。
- EXEC CLI セッションを作成できない。
- 最初のプロンプトを読み取れない。

cli_read

読み取られている内容でルータ プロンプトのパターンが発生するまで、指定されたコマンドライン インターフェイス（CLI）のチャネル ハンドラからコマンド出力を読み取ります。 一致するまで、読み取られたすべての内容を返します。

構文

```
cli_read fd
```

引数

fd	（必須）CLI チャネル ハンドラ。

結果文字列

読み取られたすべての内容。

_cerrno を設定

ルータ名を取得できない。

（注） この Tcl コマンド拡張によって、ルータ プロンプトを待つ状態がブロックされ、読み取られた内容が表示されます。

cli_read_drain

指定されたコマンドライン インターフェイス（CLI）のチャネル ハンドラのコマンド出力を読み取り、排出します。 読み取られたすべての内容を返します。

構文

```
cli_read_drain fd
```

引数

fd	（必須）CLI チャネル ハンドラ。

結果文字列

読み取られたすべての内容。

_cerrno を設定

なし

cli_read_line

指定されたコマンドライン インターフェイス（CLI）のチャネル ハンドラから、コマンド出力の 1 行を読み取ります。 読み取られた回線を返します。

構文

```
cli_read_line fd
```

引数

fd	（必須）CLI チャネル ハンドラ。

結果文字列

読み取られた回線。

_cerrno を設定

なし

（注） この Tcl コマンド拡張によって、行の末尾を待つ状態がブロックされ、読み取られた内容が表示されます。

cli_read_pattern

読み取られている内容でパターンが発生するまで、指定されたコマンドライン インターフェイス（CLI）のチャネル ハンドラからコマンド出力を読み取ります。 一致するまで、読み取られたすべての内容を返します。

（注） パターン マッチ ロジックで、Cisco IOS XR ソフトウェア コマンドから配信されるコマンド出力データを探すことによって、照会が試行されます。 照会は、出力バッファの最新の 256 文字で常に行われます。ただし、使用可能な文字がより少ない場合は、より少ない文字で照会が行われます。 正常な一致に 256 よりも多い文字が必要な場合、パターン マッチは実行されません。

構文

```
cli_read_pattern fd ptn
```

引数

fd	（必須）CLI チャネル ハンドラ。
ptn	（必須）チャネルからコマンド出力を読み取るときに、パターンが照会されます。

結果文字列

読み取られたすべての内容。

_cerrno を設定

なし

（注） この Tcl コマンド拡張によって、指定されたパターンを待つ状態がブロックされ、読み取られた内容が表示されます。

cli_write

指定された CLI チャネル ハンドラに対して実行されるコマンドを書き込みます。 CLI チャネル ハンドラによって、コマンドが実行されます。

構文

```
cli_write fd cmd
```

引数

fd	（必須）CLI チャネル ハンドラ。
cmd	（必須）実行する CLI コマンド。

結果文字列

なし

_cerrno を設定

なし

使用例

たとえば、次のように、コンフィギュレーション CLI コマンドを使用して、イーサネット インターフェイス 1/0 をアップにします。

```
if [catch {cli_open} result] {
puts stderr $result
exit 1
} else {
array set cli1 $result
}
if [catch {cli_exec $cli1(fd) "config t"} result] {
puts stderr $result
exit 1
}
if [catch {cli_exec $cli1(fd) "interface Ethernet1/0"} result] {
puts stderr $result
exit 1
}
if [catch {cli_exec $cli1(fd) "no shut"} result] {
puts stderr $result
exit 1
}
if [catch {cli_exec $cli1(fd) "end"} result] {
puts stderr $result
exit 1
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} } result] {
puts stderr $result
exit 1
```

CLI ライブラリを使用した非対話式コマンドの実行

非対話式コマンドを実行するには、**cli_exec** コマンド拡張を使用して、コマンドを発行し、次に、出力とルータ プロンプトを待ちます。 たとえば、コンフィギュレーション CLI コマンドを使用して、イーサネット インターフェイス 1/0 をアップにする例を示します。

```
if [catch {cli_open} result] {
error $result $errorInfo
```

```
} else {
set fd $result
}
if [catch {cli_exec $fd "config t"} result] {
error $result $errorInfo
}
if [catch {cli_exec $fd "interface Ethernet1/0"} result] {
error $result $errorInfo
}
if [catch {cli_exec $fd "no shut"} result] {
error $result $errorInfo
}
if [catch {cli_exec $fd "end"} result] {
error $result $errorInfo
}
if [catch {cli_close $fd} result] {
error $result $errorInfo
}
```

CLI ライブラリを使用した対話式コマンドの実行

対話式コマンドを実行するには、次の 3 つのフェーズが必要です。

- フェーズ 1：**cli_write** コマンド拡張を使用して、コマンドを発行します。
- フェーズ2：Q&A フェーズ。 **cli_read_pattern** コマンド拡張を使用して、質問を読み取り（質問テキストの照合に指定される通常パターン）、**cli_write** コマンド拡張を使用して、代わりに回答を書き戻します。
- フェーズ 3：非対話式フェーズ。 すべての質問が回答され、完了までコマンドが実行されます。 **cli_read** コマンド拡張を使用して、コマンドの出力とルータ プロンプトを待ちます。

たとえば、CLI コマンドを使用して、ブートフラッシュをまとめます。Tcl 変数 cmd_output に、このコマンドの出力を保存します。

```
if [catch {cli_open} result] {
error $result $errorInfo
} else {
array set cli1 $result
}

# Phase 1: issue the command
if [catch {cli_write $cli1(fd) "squeeze bootflash:"} result] {
error $result $errorInfo
}

# Phase 2: Q&A phase
# wait for prompted question:
# All deleted files will be removed. Continue? [confirm]
if [catch {cli_read_pattern $cli1(fd) "All deleted"} result] {
error $result $errorInfo
}
# write a newline character
if [catch {cli_write $cli1(fd) "\n"} result] {
error $result $errorInfo
}
# wait for prompted question:
# Squeeze operation may take a while. Continue? [confirm]
if [catch {cli_read_pattern $cli1(fd) "Squeeze operation"} result] {
error $result $errorInfo
}
# write a newline character
if [catch {cli_write $cli1(fd) "\n"} result] {
error $result $errorInfo
}
```

```
# Phase 3: noninteractive phase
# wait for command to complete and the router prompt
if [catch {cli_read $cli1(fd) } result] {
error $result $errorInfo
} else {
set cmd_output $result
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
error $result $errorInfo
}
```

次に、CLI **reload** コマンドを使用して、ルータがリロードされる例を示します。EEM **action_reload** コマンドによって、より効率的な方法で同じ結果が達成されますが、この例は、対話式コマンド実行での CLI ライブラリでの融通性を描くために示します。

```
# 1. execute the reload command
if [catch {cli_open} result] {
    error $result $errorInfo
} else {
    array set cli1 $result
}
if [catch {cli_write $cli1(fd) "reload"} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
}
if [catch {cli_read_pattern $cli1(fd) ".*(System configuration has been modified. Save\\\?
 \\\[yes/no\\\]: )"} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
}
if [catch {cli_write $cli1(fd) "no"} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
}
if [catch {cli_read_pattern $cli1(fd) ".*(Proceed with reload\\\? \\\[confirm\\\])"} result]
 {
    error $result $errorInfo
} else {
    set cmd_output $result
}
if [catch {cli_write $cli1(fd) "y"} result] {
    error $result $errorInfo
} else {
    set cmd_output $result
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
    error $result $errorInfo
}
```

Tcl コンテキスト ライブラリ コマンド拡張

すべての Tcl コンテキスト ライブラリ コマンド拡張は、::cisco::eem 名前空間に属します。

エクスポートされるコマンド

context_retrieve

該当するコンテキスト名、使用されている可能性があるスカラ変数名、配列型変数名、および配列の索引によって指定される Tcl 変数を取得します。取得される情報は、自動的に削除されます。

（注） 保存される情報が一度取得されると、自動的に削除されます。 その情報が別のポリシーで必要な場合、（**context_retrieve** コマンド拡張を使用して）それを取得するポリシーも、（**context_save** コマンド拡張を使用して）再度保存する必要があります。

構文

```
context_retrieve ctxt [var] [index_if_array]
```

引数

ctxt	（必須）コンテキスト名。
var	（任意）スカラ変数名または配列型変数名。この引数が指定されない場合、ヌル文字列を定義します。
index_if_array	（任意〕配列インデックス。

（注） *var* 引数がスカラ変数の場合、*index_if_array* 引数は無視されます。

var が未指定の場合、コンテキストに保存されている変数テーブル全体を取得します。

var が指定され、*index_if_array* が指定されない場合、または、*index_if_array* が指定されるが *var* がスカラ変数の場合、*var* の値を取得します。

var が指定され、*index_if_array* が指定され、*var* が配列変数の場合、指定された配列エレメントの値を取得します。

結果文字列

保存が実行されたときの状態に、Tcl グローバル変数をリセットします。

_cerrno を設定

- appl_reqinfo エラーが原因で、_cerrno、_cerr_sub_num、_cerr_sub_err、_cerr_posix_err、_cerr_str を表示する文字列。
- 変数がコンテキストにない。

使用例

次に、**context_save** コマンド拡張機能および **context_retrieve** コマンド拡張機能を使用して、データを保存し、取得する例を示します。 例は、保存と取得のペアで示されます。

例 1：保存

var が未指定か、またはパターンが指定される場合、複数の変数をコンテキストに保存します。

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

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

set testvara 123
set testvarb 345
set testvarc 789
if {[catch {context_save TESTCTX "testvar*"} errmsg]} {
      action_syslog msg "context_save failed: $errmsg"
} else {
      action_syslog msg "context_save succeeded"
}
```

例 1：取得

var が未指定の場合、複数の変数をコンテキストから取得します。

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

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

if {[catch {foreach {var value} [context_retrieve TESTCTX] {set $var $value}} errmsg]} {
      action_syslog msg "context_retrieve failed: $errmsg"
} else {
      action_syslog msg "context_retrieve succeeded"
}
if {[info exists testvara]} {
      action_syslog msg "testvara exists and is $testvara"
} else {
      action_syslog msg "testvara does not exist"
}
if {[info exists testvarb]} {
      action_syslog msg "testvarb exists and is $testvarb"
} else {
      action_syslog msg "testvarb does not exist"
}
if {[info exists testvarc]} {
      action_syslog msg "testvarc exists and is $testvarc"
} else {
      action_syslog msg "testvarc does not exist"
}
```

例 2：保存

var が指定される場合、var の値を保存します。

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

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

set testvar 123
if {[catch {context_save TESTCTX testvar} errmsg]} {
      action_syslog msg "context_save failed: $errmsg"
} else {
      action_syslog msg "context_save succeeded"
}
```

例 2：取得

var が指定され、index_if_array が指定されない場合、または、index_if_array が指定されるが var がスカラ変数の場合、var の値を取得します。

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

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

if {[catch {set testvar [context_retrieve TESTCTX testvar]} errmsg]} {
      action_syslog msg "context_retrieve failed: $errmsg"
} else {
      action_syslog msg "context_retrieve succeeded"
}
if {[info exists testvar]} {
      action_syslog msg "testvar exists and is $testvar"
} else {
      action_syslog msg "testvar does not exist"
}
```

例 3：保存

var が指定される場合、それが配列の場合でも、var の値を保存します。

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

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

array set testvar "testvar1 ok testvar2 not_ok"
if {[catch {context_save TESTCTX testvar} errmsg]} {
      action_syslog msg "context_save failed: $errmsg"
} else {
      action_syslog msg "context_save succeeded"
}
```

例 3：取得

var が指定され、index_if_array が指定されず、var が配列変数の場合、配列全体を取得します。

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

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

if {[catch {array set testvar [context_retrieve TESTCTX testvar]} errmsg]} {
      action_syslog msg "context_retrieve failed: $errmsg"
} else {
      action_syslog msg "context_retrieve succeeded"
}
if {[info exists testvar]} {
      action_syslog msg "testvar exists and is [array get testvar]"
} else {
      action_syslog msg "testvar does not exist"
}
```

例 4：保存

var が指定される場合、それが配列の場合でも、var の値を保存します。

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

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

array set testvar "testvar1 ok testvar2 not_ok"
if {[catch {context_save TESTCTX testvar} errmsg]} {
      action_syslog msg "context_save failed: $errmsg"
} else {
      action_syslog msg "context_save succeeded"
}
```

例 4：取得

var が指定され、index_if_array が指定され、var が配列変数の場合、指定された配列エレメントの値を取得します。

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

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

if {[catch {set testvar [context_retrieve TESTCTX testvar testvar1]} errmsg]} {
      action_syslog msg "context_retrieve failed: $errmsg"
} else {
      action_syslog msg "context_retrieve succeeded"
}
if {[info exists testvar]} {
      action_syslog msg "testvar exists and is $testvar"
} else {
      action_syslog msg "testvar doesn't exist"
}
```

context_save

現在およびグローバルな名前空間で、指定されたパターンが、識別情報として指定されたコンテキスト名と一致する、Tcl 変数を保存します。 この Tcl コマンド拡張を使用すると、ポリシー外の情報が保存されます。 保存された情報は、**context_retrieve** コマンド拡張を使用して、異なるポリシーによって取得できます。

（注） 保存される情報が一度取得されると、自動的に削除されます。 その情報が別のポリシーで必要な場合、（**context_retrieve** コマンド拡張を使用して）それを取得するポリシーも、（**context_save** コマンド拡張を使用して）再度保存する必要があります。

構文

```
context_save ctxt [pattern]
```

引数

ctxt	（必須）コンテキスト名。
pattern	（任意）**string match** Tcl コマンドによって使用される、glob-style パターン。 この引数が指定されない場合、パターンのデフォルトは、ワイルドカード * です。 glob パターンで使用されている、3 つの構成があります。 • * = すべての文字 • ? = 1 文字 • [abc] = 文字のセットの 1 つと照合

結果文字列

なし

_cerrno を設定

appl_setinfo エラーが原因で、_cerrno、_cerr_sub_num、_cerr_sub_err、_cerr_posix_err、_cerr_str を表示する文字列。

使用例

context_save コマンド拡張機能および **context_retrieve** コマンド拡張機能を使用して、データの保存や取得を行う方法の例については、使用例, （230 ページ）を参照してください。

第 4 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、IPアプリケーションやサービスの IP サービス レベルの分析、生産性の向上、運用コストの削減、およびネットワーク停止頻度の低減が可能になります。

IP SLA を使用することで、サービス プロバイダーは、サービス レベル契約を測定および提供できます。 IP SLA は、ネットワーク アセスメントの実行、Quality of Service（QoS）の検証、新規サービスの展開の簡易化、およびネットワークのトラブルシューティングにおける管理者の支援が可能です。

（注） この章で使用する IP SLA コマンドの詳しい説明については、『*Cisco ASR 9000 Series Aggregation Services Router System Management Command Reference*』の「*IP Service Level Agreement Commands on Cisco ASR 9000* シリーズ ルータ」モジュールを参照してください。

IP サービス レベル契約を実現するための機能履歴

リリース	変更内容
リリース 3.7.2	この機能が導入されました。

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。
- 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。
- 頻度を 60 秒未満に設定することで、パケットのスケーラビリティが指数的に増加し、スケジュールされた動作の開始時刻が同じ場合に、設定可能な IP SLA IPSLA 動作のパフォーマンスに悪影響を及ぼすおそれがあります。
- IP SLA は HA に対応していません。
- 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 ASR 9000 Series シリーズ アグリゲーション サービス ルータはハードウェアのタイムスタンプ処理をサポートしていません。
- 同時に動作をスケジューリングするために現在有効なスケール番号は次のとおりです。
  - UDP エコー動作の最大数は、デフォルトの頻度では 400 動作です。
  - 到達可能性を測定する場合の UDP エコー動作の最大数は、デフォルトの頻度では 2000 動作です。
  - UDP ジッター動作の最大数は、デフォルトの頻度では 400 動作です。
  - 到達可能性を測定する場合の UDP ジッター動作の最大数は、デフォルトの頻度では 1000 動作です。
  - ICMP エコー動作の最大数は、デフォルトの頻度では 400 動作です。
  - 到達可能性を測定する場合の ICMP エコー動作の最大数は、デフォルトの頻度では 2000 動作です。
  - ICMP エコーパス動作の最大数は、デフォルトの頻度では 400 動作です。
  - 到達可能性を測定する場合の ICMP エコーパス動作の最大数は、デフォルトの頻度では 2000 動作です。

- デフォルトの頻度で出力パケット破棄がある場合の ICMP ジッター動作の最大数は、は 75 動作です。
- デフォルトの頻度で出力パケット破棄がある場合の MPLS LSP ping 動作の最大数は、は 100 動作です。
- デフォルトの頻度で出力パケット破棄がある場合の MPLS LSP トレース動作の最大数は、100 動作です。

IP サービス レベル契約の実装に関する情報

IP SLA を実装するには、次の概念について理解する必要があります。

IP サービス レベル契約テクノロジーについて

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）
- Packet loss
- ネットワーク リソースのアベイラビリティ

IP SLA は、以前サービス保証エージェント（SAA）と呼ばれていたテクノロジーが元になっています。 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。

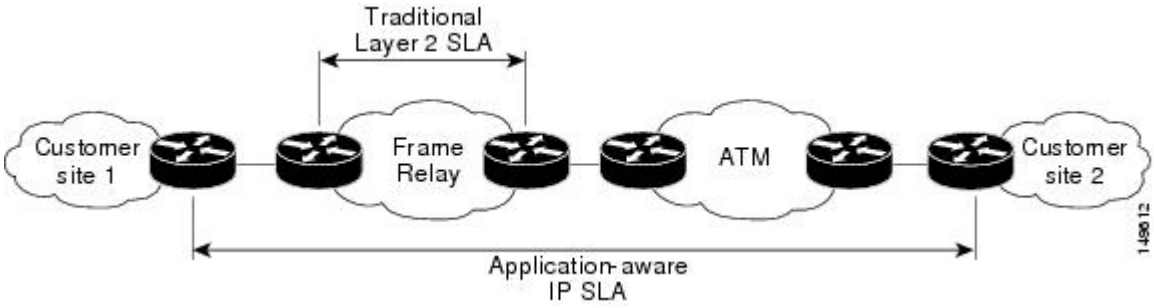
特定の IP SLA 動作に応じて、遅延の統計情報、パケット損失、ジッター、パケット シーケンス、接続性、およびパスがルータによってモニタされて保存され、コマンドライン インターフェイス（CLI）、Extensive Markup Language（XML）、および SNMP MIB を通じて提供されます。 IP SLA は Cisco RTTMON MIB を使用して外部のネットワーク管理システム（NMS）アプリケーションおよびシスコのデバイスで動作している IP SLA 動作と通信します。 IP SLA によって参照されるオブジェクト変数の詳しい説明については、Cisco MIB Locator から入手できる CISCO-RTTMON-MIB.my ファイルの文章を参照してください。

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 3：従来のサービス レベル契約と IP SLA の範囲, （238 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されているかを示します。

図 *3*：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 *20*：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。

改良の種類	説明
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。
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 *21*：*IP SLA* の利点の一覧

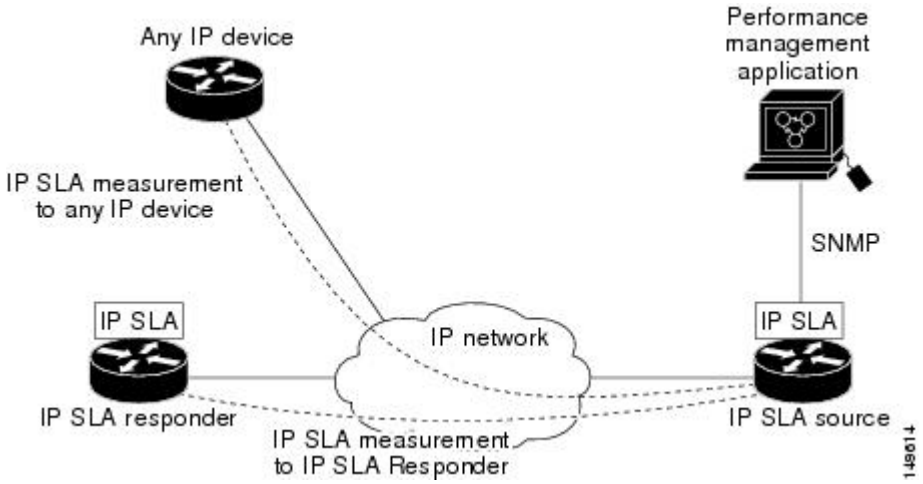
利点	説明
IP SLA のモニタリング	サービス レベル契約モニタリング、評価、および検証の提供
ネットワーク パフォーマンス モニタリング。	ネットワーク内のジッター、遅延、パケット損失が測定できる。 また、IP SLA は、予防的な通知に加えて、連続的で、信頼性が高く、予測可能な測定を提供します。
IP サービス ネットワーク稼働状態評価	既存の QoS が新しい IP サービスに対して十分であることの検証

利点	説明
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 4：IP SLA の動作, （240 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IPSLA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 *4*：*IP SLA* の動作



動作は、ターゲット デバイスで動作するために IP SLA レスポンダ コンポーネントを必要としているかどうかに応じて、2 つのクラスに分かれています。 前者は、シスコ デバイスのみで使用されるのに対し、後者はIP接続が可能な任意のデバイスで使用されます。インターネット制御メッセージ プロトコル（ICMP）に基づく動作は、第 2 のクラスの例であり、UDP ベースの動作は最初のクラスの例です。

レスポンダ ベースの動作では、IP SLA レスポンダが宛先デバイスでイネーブルになっており、IP SLA パケットの遅延などの情報を提供します。 レスポンダベースの動作では、前述の ICMP 動作に対する精度が向上し、単方向測定の機能を提供します。 IP SLA 送信元デバイスへの応答に、レスポンダは処理遅延に関する情報を含めます。 IP SLA 送信元デバイスはその最終的なパフォーマンス計算で遅延を除去します。 レスポンダの使用は、UDP エコー動作では任意ですが、UDP ジッ

ター動作では必須です。 IP SLA レスポンダが使用されない場合、ターゲット デバイスは UDP エコー動作をサポートする必要があります。

ICMP 動作で、IP SLA デバイスはいくつかの ICMP パケットを宛先に送信します。 宛先デバイスは、任意の IP デバイスであり、エコーに応答します。 送信元 IP SLA デバイスは送信および受信したタイムスタンプを使用して応答時間を計算します。 ICMP エコー動作は、従来の拡張された ping ユーティリティに似ており、送信元デバイスと宛先デバイスの間の応答時間のみを測定します。 ICMP パスエコーおよびパスジッター動作は、traceroute メカニズムを使用してパス全体を認識します。 以降の ICMP パケットは各パス ノードに送信され、測定が相互に関連付けられて、ホップバイホップのラウンドトリップ遅延とジッター情報が提供されます。

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 CLI、XML、または NMS システムで SNMP を使用し、動作の結果を表示および解釈します。

IP サービス レベル契約の動作タイプ

IP SLA は、さまざまなタイプの動作を設定して、応答時間、ジッター、スループット、およびパケット損失を測定します。 また、各動作は複数のアプリケーションにマッピングされます。

次の表に、さまざまな動作タイプの一覧を示します。

表 *22*：*IP SLA* のための動作タイプ

動作	説明
UDP エコー	ラウンドトリップ遅延を測定し、UDP トラフィックの応答時間を正確に測定するために役立ちます。
UDP ジッター	ラウンドトリップ遅延、単方向遅延、単方向ジッター、双方向ジッター、単方向パケット損失を測定します。
ICMP エコー	パス全体のラウンドトリップ遅延を測定します。

動作	説明
ICMP パスエコー	ルータとネットワーク上の任意の IP デバイスの間のホップバイホップの応答時間を計算します。 パスは traceroute アルゴリズムを使用して検出され、送信元ルータとパス内の各中間ホップの間の応答時間が測定されます。 送信元デバイスと宛先デバイスの間に複数の等コストルートがある場合、ICMP パスエコー動作は、設定可能なルーズ ソース ルーティング（LSR）オプションを使用していずれかのパスを選択します。
ICMP パスジッター	IP ネットワーク内のホップバイホップ ジッター、パケット損失、および遅延測定統計情報を測定します。
MPLS LSP ping	ラベル スイッチド パス（LSP）の接続をテストし、MPLS ネットワーク内の LSP のラウンドトリップ遅延を測定します。 次の Forwarding Equivalence Class（FEC）がサポートされています。 • IPv4 ラベル配布プロトコル（LDP） • トラフィック エンジニアリング（TE）トンネル • 疑似回線 エコー要求は、FEC に属する他のパケットと同じデータ パスに沿って送信されます。 エコー要求パケットがパスの終端に達すると、出力のラベル スイッチング ルータ（LSR）のコントロール プレーンに送信されます。 LSR は、それが本当に FEC の出力であることを確認し、MPLS パスを確認する FEC に関する情報が格納されているエコー応答パケットを送信します。 デフォルトの VRF テーブルのみがサポートされています。

動作	説明
MPLS LSP トレース	LSP パスのホップバイホップのルートをトレースし、MPLS ネットワーク内の IPv4 LDP プレフィックスと TE トンネル FEC のホップバイホップのラウンドトリップ遅延を測定します。 エコー要求パケットは、各中継 LSR のコントロールプレーンにデータを送信し、そこでこのパスの中継 LSR であるかどうかが確認されます。 また、各中継 LSR は、テスト対象の LSP にバインドされているラベルに関する情報を返します。 デフォルトの VRF テーブルのみがサポートされています。

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。 通常の ICMP ベースの測定では得られない、追加の統計情報も提供されます。 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 4：IP SLA の動作, （240 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。

IP SLA レスポンダは、UDP ジッター動作で使用する必要がありますが、UDP エコー動作では任意です。 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

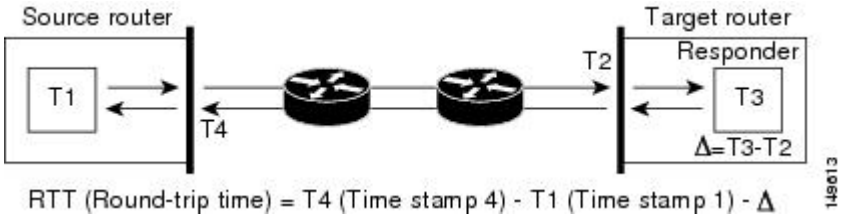
T3 は、応答パケットが IP SLA レスポンダ ノードで送信された時刻であり、T1 は送信元ノードで要求が送信された時刻です。 他の優先順位が高いプロセスにより、ルータは着信パケットを処理するために数十ミリ秒を要します。 テストパケットへの応答が処理を待つ間キューに格納される

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。 ネットワークアクティビティが活発なとき、ICMP ping テストによる応答時間は長く、不正確になることがよくあります。それに対して、IP SLA ベースのレスポンダは正確な時間を示します。

図 5：IP SLA レスポンダのタイムスタンプ処理, （244 ページ） に、レスポンダの動作を示します。 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 *5*：*IP SLA* レスポンダのタイムスタンプ処理



IP SLA の VRF サポート

サービス プロバイダーは、コア ネットワークとカスタマー ネットワークの両方の観点からネットワークのパフォーマンスを監視および測定する必要があります。 そのためには、デフォルトの VPN ルーティングおよび転送（VRF）テーブルに加えて、IP SLA 動作でデフォルト以外の VRF も使用する必要があります。 表 22：IP SLA のための動作タイプ, （241 ページ） では、動作がデフォルト以外の VRF テーブルの使用をサポートしているかどうかなど、さまざまな IP SLA 動作について説明しています。

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。

IP SLA：予防的しきい値モニタリング

ここでは、しきい値と反応トリガーを使用したIP SLAのプロアクティブな監視機能について説明します。 IP SLA では、IP アプリケーションの監視、分析、IP サービス レベルの確認を行って、生産性の向上、運用コストの削減、ネットワークの輻輳や停止の発生の低減を行うことができます。 IP SLA は、アクティブ トラフィック監視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 を使用したプロアクティブなしきい値監視を設定する必要があるタスクを実行するには、次の概念について理解する必要があります。

IP SLA 反応コンフィギュレーション

IP SLA は、特定の測定されたネットワーク条件に反応するように設定できます。 たとえば、IP SLA が接続上で測定したジッターが大きすぎる場合、IP SLA はネットワーク管理アプリケーションに通知を生成したり、より多くのデータを収集するために別のIP SLA動作をトリガーしたりできます。

IP SLA 反応を設定するには、**ipsla reaction operation** コマンドを使用します。

IP SLA しきい値モニタリングおよび通知

IP SLA では、ジッター平均、双方向ラウンドトリップ時間、接続性などのパフォーマンス パラメータのしきい値モニタリングがサポートされています。 パケット損失とジッターでは、いずれかの方向（たとえば、送信元から宛先と、宛先から送信元）での違反またはラウンドトリップ値について、通知を生成できます。

MPLS LSP モニタリング

IP サービス レベル契約（SLA）ラベル スイッチド パス（LSP）のモニタ機能は、レイヤ 3 マルチプロトコル ラベル スイッチング（MPLS）バーチャル プライベート ネットワーク（VPN）を予防的に監視するための機能を備えています。 この機能は、ネットワークのアベイラビリティを確認したり、MPLS VPN内のプロバイダーエッジ（PE）ルータ間のネットワーク接続をテストするために便利です。 MPLS LSP モニタを設定すると、ネットワーク トポロジに基づいて、自動的にIP SLA LSP ping または LSP traceroute 処理を生成または削除できます。

MPLS SLA モニタ機能では、IP SLA 動作の複数動作スケジューリングを実行することも可能であり、SNMP トラップ通知とSyslogメッセージを使用した予防的しきい値違反モニタリングもサポートされています。

MPLS LSP モニタ機能を使用するには、次の概念を理解しておく必要があります。

MPLS LSP モニタリングのしくみ

MPLS LSP モニタ機能では、レイヤ 3 MPLS VPN を予防的にモニタできます。 MPLS LSP モニタの動作方法の一般的なプロセスは次のとおりです。

1. ユーザは、MPLS LSP モニタ インスタンスを設定します。

   MPLS LSP モニタ インスタンスを設定することは、標準的な IP SLA 動作の設定に似ています。たとえば、MPLS LSP モニタ インスタンスのすべての動作パラメータは、動作の ID 番号を指定した後で設定します。ただし、標準的な IP SLA 動作と異なり、これらの設定されたパラメータは、MPLS LSP モニタ インスタンスによって作成される個々の IP SLA LSP ping および LSP traceroute 動作の基本設定として使用されます。

   最初の MPLS LSP モニタ インスタンスが設定され、開始がスケジュールされると、BGP ネクストホップ ネイバー探索がイネーブルになります。BGP ネクストホップ ネイバー探索，（247 ページ）を参照してください。

2. ユーザが MPLS LSP モニタ インスタンスについて予防的しきい値違反モニタリングを設定します。

3. ユーザが、MPLS LSP モニタ インスタンスの複数動作スケジューリング パラメータを設定します。

4. 選択した設定オプションに応じて、MPLS LSP モニタ インスタンスは、該当する各 BGP ネクストホップ ネイバーに対する、個々の IP SLA LSP ping または LSP traceroute 動作を自動的に作成します。

   すべての MPLS LSP モニタ動作について、BGP ネクストホップ ネイバーあたり 1 つの IP SLA LSP ping 動作または LSP traceroute 動作のみが設定されます。 しかし、特定の PE ルータで、同時に複数の MPLS LSP モニタ インスタンスを実行できます。 （詳細については、この項の最後にある注記を参照してください）。

5. 各 IP SLA LSP ping または LSP traceroute 動作は、送信元 PE ルータと検出された宛先 PE ルータの間のネットワーク接続を測定します。

（注） 複数の MPLS LSP モニタ インスタンスを特定の PE ルータで同時に実行できます。 たとえば、ある MPLS LSP モニタ インスタンスを、VPN1 という VRF に属する BGP ネクストホップ ネイバーを探索するように設定できます。 同じ PE ルータで、別の MPLS LSP モニタ インスタンスを、VPN2 という VRF に属するネイバーを探索するように設定できます。 この場合、BGP ネクストホップ ネイバーが VPN1 と VPN2 の両方に属していた場合、PE ルータはこのネイバーに対して 2 つの IP SLA 動作（1 つは VPN1 用、もう 1 つは VPN2 用）を作成します。

IP SLA 動作の MPLS LSP モニタ データベースへの追加と削除

MPLS LSP モニタ インスタンスは、特定の VPN に対して追加または削除された BGP ネクストホップ ネイバーについて定期的な通知を受けます。 この情報は、MPLS LSP モニタが保持するキューに格納されます。 キュー内の情報とユーザ指定の期間に基づき、新たに検出された PE ルー

タに対して新しい IP SLA 動作が自動的に作成され、有効でなくなった PE ルータに対する既存の IP SLA 動作は自動的に削除されます。

BGP ネクストホップ ネイバー探索

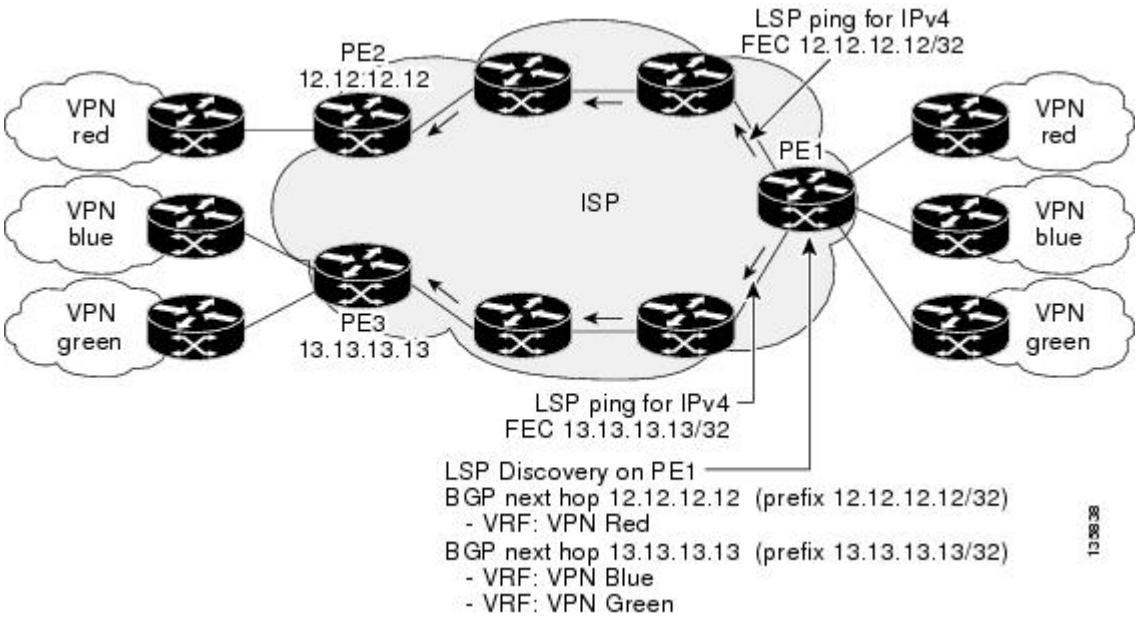
BGP ネクストホップ ネイバー探索は、送信元プロバイダー エッジ（PE）ルータに関連付けられているすべての VRF によって使用中の BGP ネクストホップ ネイバーを見つけるために使用されます。 ほとんどの場合、これらのネイバーは PE ルータです。

BGP ネクストホップ ネイバー探索がイネーブルな場合、ローカル VRF とグローバル ルーティング テーブルの情報に基づいて、送信元 PE に関連付けられているすべての VRF によって使用中の BGP ネクストホップ ネイバーのデータベースが生成されます。 ルーティング アップデートが受信されると、新しい BGP ネクスト ホップ ネイバーがただちにデータベースに追加されます。 ただし、有効でなくなった BGP ネクストホップ ネイバーは、ユーザ定義に従って、定期的にデータベースから削除されます。

図 6：単純な VPN の BGP ネクストホップ ネイバー探索, （248 ページ） に、インターネット サービス プロバイダー（ISP）の単純な VPN シナリオでの BGP ネクストホップ ネイバー探索の動作を示します。 この例で、ルータ PE1 に関連付けられた 3 つの VPN があります（赤、青、緑）。 ルータ PE1 から見ると、これらの VPN には、BGP ネクスト ホップ ネイバー PE2（ルータ ID：12.12.12.12）および PE3（ルータ ID：13.13.13.13）を経由してリモートで到達可能です。 BGP ネクストホップ ネイバー探索プロセスがルータ PE1 でイネーブルになっている場合、ローカル VRF とグローバル ルーティング テーブルに基づいてデータベースが生成されます。 この例のデータベースには、2 つの BGP ネクスト ホップ ルータ エントリとして PE2 12.12.12.12 および PE3 13.13.13.13 が格納されます。 ルーティング エントリは、どのネクスト ホップ ルータがどの特定の VRF 内に属しているか区別するために、ネクスト ホップ ルータ単位で維持されます。 各ネクストホップ ルータ エントリに対し、グローバル ルーティング テーブル中の BGP ネクストホップ

ルータの IPv4 Forward Equivalence Class（FEC）が、MPLS LSP ping 動作で使用するために提供されます。

図 *6*：単純な *VPN* の *BGP* ネクストホップ ネイバー探索



IP SLA LSP ping 動作と LSP traceroute 動作

この機能により、IP SLA LSP ping 動作と IP SLA LSP traceroute 動作に対するサポートが追加されます。 これらの動作は、ネットワークの接続性の問題をトラブルシューティングし、MPLS VPN のネットワークのアベイラビリティを判定するために役立ちます。 MPLS LSP モニタリングを使用する場合、送信元 PE ルータと検出された宛先 PE ルータの間のネットワーク接続を測定するために、IP SLA LSP ping 動作と LSP traceroute 動作が自動的に作成されます。 個々の IP SLA LSP ping 動作と LSP traceroute 動作を手動で設定することもできます。 これらの動作の手動の設定は、接続性の問題をトラブルシューティングするために役立ちます。

MPLS LSP モニタリングを使用して IP SLA LSP ping または LSP traceroute 動作を設定する方法の詳細については、MPLS LSP モニタリング ping インスタンスの設定，（330 ページ）およびMPLS LSP モニタリング トレース インスタンスの設定，（335 ページ）を参照してください。

IP SLA LSP ping 動作と IP SLA LSP traceroute 動作は、それぞれ MPLS LSP ping 機能と MPLS LSP traceroute 機能で使用されるのと同じインフラストラクチャに基づいて、LSP をテストするためのエコー応答パケットとエコー要求パケットを送受信します。

MPLS LSP モニタリングの予防的しきい値モニタリング

MPLS LSP モニタの予防的しきい値モニタリング サポート機能では、ユーザ定義の応答条件（接続損失やタイムアウトなど）が満たされたときに、SNMP トラップ通知と Syslog メッセージをト

リガーできます。 MPLS LSP モニタ インスタンスのしきい値モニタリング動作の設定方法は、標準的な IP SLA 動作の設定方法と同様です。

LSP ヘルス モニタの複数動作スケジューリング

MPLS LSP モニタの複数動作スケジューリング サポート機能では、（各 MPLS LSP モニタ インスタンスに対して）自動的に作成された IP SLA 動作を、指定された期間（スケジュール期間）にわたって均等に分散される間隔で開始し、指定された頻度で再開するように簡単にスケジューリングできます。 複数動作スケジューリングは、多数の PE ネイバーが存在し、その結果として多数の IP SLA 動作が同時に稼働している送信元 PE ルータ上で MPLS LSP モニタリングがイネーブルにされる場合に特に有用です。

（注） （新たに検出された BGP ネクスト ホップ ネイバーに対して）新たに作成された IP SLA 動作は、現在稼働している動作と同じスケジュール期間に追加されます。 同時に開始する動作が多くなりすぎないように、複数動作スケジューリング機能は、それらの動作を、スケジュール期間にわたって均一に分散されるランダムな間隔で開始するようにスケジューリングします。

LSP パス ディスカバリ

LSP パス ディスカバリ（LPD）は、MPLS LSP モニタ（MPLSLM）の拡張で、MPLSLM インスタンスの一部である操作を許可し、パス ディスカバリ プロセスを開始してその結果が処理されます。 この機能には、MPLS OAM インフラストラクチャにより LSPV サーバを通じて提供されるツリー トレース機能が必要です。

2 つの PE ルータ間にコストが等しい複数のパスが存在する場合（これを等コスト マルチパス（ECMP）と呼びます）、これら PE ルータの間にあるルータは、転送するトラフィックの特性（たとえばパケット中の宛先アドレス）に基づいて、ロードバランシングを行います。 このようなネットワーク トポロジでは、PE ルータ間の使用可能なパスのうち 1 つ（またはいくつか）をモニタするだけでは、トラフィックが正しく転送される保証が得られません。

LPD は、**path discover** コマンドを使用して設定します。

（注） LPD 機能を使用した場合、LSPV サーバが一度に多数のパス ディスカバリ要求を受信すると、CPU の負荷が高まる可能性があります。

IP サービス レベル契約の実装方法

ここでは、次の手順について説明します。

UDP ジッター動作を使用した IP サービス レベルの設定

IP SLA UDP ジッター モニタリング動作は、VoIP、Video over IP、リアルタイム会議などのリアルタイム トラフィックに対するネットワークの適切さを診断するように設計されています。

ジッターはパケット間の遅延がばらつくことを指します。 複数のパケットが送信元から宛先に連続的に送信された（たとえば 10 ms 間隔で送信された場合）、ネットワークが理想的に振る舞えば、宛先でも 10 ms 間隔でパケットを受信します。 しかし、ネットワーク内に遅延（キューイング、代替ルートを介した受信など）が存在する場合、パケット間の到着遅延は、10 ms より大きい場合も、10 ms より小さい場合もあります。 この例を使用すると、正のジッター値は、パケットが 10 ms を超える間隔で到着することを示します。 パケットの到着が 12 ミリ秒の場合のジッター値は +2 ミリ秒（正の値）です。8 ミリ秒で到着する場合は、2 ミリ秒（負の値）です。 Voice over IP（VoIP）など遅延に影響されやすいネットワークでは、正のジッター値は望ましくありません。0 のジッター値が理想的です。

しかし、IP SLA UDP ジッター動作の機能は、ジッターのモニタリングだけではありません。 IP SLA が生成するパケットは、パケットの送信シーケンスと受信シーケンス情報を伝送し、送信元ターゲットと動作ターゲットとの間でタイム スタンプの送受信を行います。 UDP ジッター動作は、次の機能を測定できます。

- 方向別ジッター（送信元から宛先へ、宛先から送信元へ）
- 方向別パケット損失
- 方向別遅延（一方向遅延）
- ラウンドトリップ遅延（平均 RTT）

データの送信と受信でパスが異なることがあるので（非対称）、方向別データを使用してネットワークの輻輳などの問題が発生している場所を簡単に特定できます。

UDP ジッター動作は、合成（シミュレーション）UDP トラフィックを生成して機能します。 デフォルトでは、ペイロード サイズが 32 バイト（S）のパケット フレーム 10 個（N）を 20 ミリ秒（T）ごとに生成し、60 秒（F）ごとに動作を繰り返します。 に示すように、これらのパラメータは、提供している IP サービスまたはこれから提供する IP サービスの最適なシミュレーションを行うようにそれぞれユーザ設定可能です。

ここでは、次の手順について説明します。

宛先デバイスでの IP SLA レスポンダのイネーブル化

IP SLA レスポンダは、動作のターゲットであるターゲット デバイスでイネーブルにする必要があります。

ipsla responder コマンドを設定することにより、IP SLA レスポンダは、UDP ポート 1967 をオープンし、（プローブではなく）制御パケットを待ちます。 ポートは、UDP ポート 1967 を通じ、IP SLA 制御パケットを使用して動的にオープンまたはクローズできます。 また、永続的なポートも設定できます。

永続的なポートは、設定が削除されるまでオープンされます。 ポートは設定によってオープンされるため、エージェントは、制御要求パケットを使用せずに、IP SLA プローブ パケットを永続的なポートに直接送信できます。

永続的なポートを使用しない場合、**ipsla responder** コマンドのみを使用して設定する必要があります。

動的なポートを使用するには、次の例に示すように **ipsla responder** コマンドを使用します。

```
configure
ipsla responder
```

動的なポートは、エージェント側で動作を開始したときに、IP SLA 制御プロトコルを通じて、レスポンダ側でオープンされます。

例は、レスポンダ側の永続的なポートとして設定されています。 UDP エコーと UDP ジッターは、動的なポートまたは永続的なポートを使用できます。 UDP ジッターで永続的なポートを使用する場合、一部の統計情報は収集されません。 たとえば、RTT は、UDP ジッターで永続的なポートを使用した場合でも RTT が収集されます。

手順の概要

1. **configure**
2. **ipsla responder**
3. **type udp ipv4 address** *ip-address* **port** *port*
4. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla responder**
	
	例：
	
	`RP/0/RSP0/CPU0:router(config)# ipsla responder`
	`RP/0/RSP0/CPU0:router(config-ipsla-resp)#`
		UDP エコーまたはジッター動作に対する IP SLA レスポンダをイネーブルにします。

	コマンドまたはアクション	目的
ステップ 3	**type udp ipv4 address** *ip-address* **port** *port* 例： `RP/0/RSP0/CPU0:router(config-ipsla-resp)# type udp ipv4 address 12.25.26.10 port 10001`	IP SLA レスポンダ上で永続的アドレスとポートをイネーブルにします。

ステップ 4	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

次の作業

IP SLA レスポンダをイネーブルにした後、送信元デバイスでの UDP ジッター動作の設定およびスケジューリング，（252 ページ）を参照してください。

送信元デバイスでの UDP ジッター動作の設定およびスケジューリング

IP SLA 動作は、合成（シミュレーション）ネットワーク トラフィックを生成して機能します。 1 つの IP SLA 動作（たとえば IP SLA 動作 10）は、動作の存続期間の間、指定された頻度で繰り返されます。

1 つの UDP ジッター動作は、指定された頻度 F の、送信元ルータからターゲット ルータへの、T ミリ秒間隔で送信される、サイズ S の N 個の UDP パケットからなります。 デフォルトでは、ペイロード サイズが 32 バイト（S）のパケット 10 個（N）を 20 ミリ秒（T）ごとに生成し、60 秒（F）ごとに動作を繰り返します。 これらの各パラメータは、表 23：UDP ジッター動作パラメータ，（253 ページ） に示すように、ユーザが設定可能です。

表 *23*：*UDP* ジッター動作パラメータ

UDP ジッター動作パラメータ	デフォルト	設定方法
パケット数（n）	10 パケット	• *operation-number* 引数を指定した **ipsla operation** コマンド • **type udp jitter** コマンド • *count* 引数を指定した **packet count** コマンド
パケットあたりのペイロード サイズ（S）	32 バイト	• *operation-number* 引数を指定した **ipsla operation** コマンド • **type udp jitter** コマンド • *size* 引数を指定した **datasize request** コマンド
パケット間隔（ミリ秒単位）（T）	20 ms	• *operation-number* 引数を指定した **ipsla operation** コマンド • **type udp jitter** コマンド • *interval* 引数を指定した **packet interval** コマンド
動作を繰り返すまでの経過時間（秒単位）（F）	60 秒	• *operation-number* 引数を指定した **ipsla operation** コマンド • **type udp jitter** コマンド • *seconds* 引数を指定した **frequency** コマンド

送信元デバイスで UDP ジッター動作を設定する前提条件

UDP ジッター動作の使用には、IP SLA レスポンダをターゲットのシスコ デバイスでイネーブルにする必要があります。 IP SLA レスポンダをイネーブルにするには、宛先デバイスでの IP SLA レスポンダのイネーブル化, （250 ページ）のタスクを実行します。

送信元デバイスでの基本的な UDP ジッター動作の設定およびスケジューリング

UDP ジッター動作を設定およびスケジューリングできます。

手順の概要

1. **configure**
2. **ipsla operation** *operation-number*
3. **type udp jitter**
4. **destination address** *ipv4address*
5. **destination port** *port*
6. **packet count** *count*
7. **packet interval** *interval*
8. **frequency** *seconds*
9. **exit**
10. **ipsla schedule operation** *op-num*
11. **life** { **forever** | *seconds*}
12. **ageout** *seconds*
13. **recurring**
14. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
15. 次のいずれかのコマンドを使用します。
    - **end**
    - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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。

	コマンドまたはアクション	目的
ステップ 2	**ipsla operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla operation 432`	動作番号を指定します。 範囲は 1 ～ 2048 です。
ステップ 3	**type udp jitter** 例： `RP/0/RSP0/CPU0:router(config-ipsla-op)# type udp jitter`	動作を UDP ジッター動作として設定し、動作の特性を設定します。
ステップ 4	**destination address** *ipv4address* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-jitter)# destination address 12.25.26.10`	UDP ジッター動作の宛先の IP アドレスを指定します。
ステップ 5	**destination port** *port* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-jitter)# destination port 11111`	宛先ポート番号を指定します。範囲は 1 ～ 65535 です。
ステップ 6	**packet count** *count* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-jitter)# packet count 30`	（任意）プローブ中に送信されるパケット数を指定します。 UDP ジッター動作の場合、範囲は 1 ～ 60000 です。 ICMP パスジッター動作の場合、範囲は 1 ～ 100 です。 送信されるデフォルトのパケット数は 10 です。
ステップ 7	**packet interval** *interval* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-jitter)# packet interval 30`	（任意）パケット間隔を指定します。 パケット間のデフォルト間隔は 20 ミリ秒です。
ステップ 8	**frequency** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-jitter)# frequency 300`	（任意）指定した IP SLA 動作がネットワークに送信されるレートを設定します。 • （任意）*seconds* 引数を使用して、IP SLA 動作間隔の秒数を指定します。 有効な値の範囲は 1 ～ 12604800 秒です。 デフォルトは 60 秒です。

	コマンドまたはアクション	目的

ステップ 9	**exit**
	
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-udp-jitter)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/CPU0:router(config)#`
		IP SLA コンフィギュレーション モードおよび動作モードを終了し、CLIをグローバル コンフィギュレーション モードに戻します。

ステップ 10	**ipsla schedule operation** *op-num*
	
	例：
	`RP/0/RSP0/CPU0:router(config)# ipsla schedule operation 432`
	`RP/0/RSP0/CPU0:router(config-ipsla-sched)#`
		動作の開始時間をスケジューリングします。 基本スケジュールを設定できます。

ステップ 11	**life** { **forever** | *seconds*} 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# life 30`	**forever** キーワードを指定すると、動作を無期限で実行するようにスケジューリングされます。 *seconds* 引数を指定すると、動作のライフタイムが秒単位でスケジューリングされます。 デフォルトの動作のライフタイムは 3600 秒（1 時間）です。
ステップ 12	**ageout** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# ageout 3600`	（任意）情報をアクティブに収集していない場合、動作をメモリに常駐させておく時間を秒数で指定します。 デフォルト値の 0 秒は、動作がタイムアウトしないことを意味します。
ステップ 13	**recurring** 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# recurring`	（任意）動作が毎日指定された時刻に自動的に開始され、指定された期間実行されるように指定します。
ステップ 14	**start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*] 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00`	動作が開始される時刻を指定します。 ここでは、次のキーワードについて説明します。 • （任意）**pending** キーワードを使用して、動作を保留（未開始）状態にしておくように設定します。 デフォルトは inactive です。 **start-time** コマンドが指定されていない場合、開始時刻が設定されるか、即時スタートを実行するトリガーが発生するまで、情報は収集されません。 • （任意）**now** キーワードを使用して、動作を即時スタートする必要があることを示します。

	コマンドまたはアクション	目的
		• （任意）**after** キーワードおよび関連する引数を使用して時刻を指定します。ここで指定した時刻以降に、動作が情報の収集を開始します。

ステップ 15	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

追加特性を指定した UDP ジッター動作の設定およびスケジューリング

UDP ジッター動作を設定およびスケジューリングできます。

手順の概要

1. **configure**
2. **ipsla operation** *operation-number*
3. **type udp jitter**
4. **vrf** *vrf-name*
5. **destination address** *ipv4address*
6. **destination port** *port*
7. **frequency** *seconds*
8. **statistics** [**hourly** | **interval** *seconds*]
9. **buckets** *hours*
10. **distribution count** *slot*
11. **distribution interval** *interval*
12. **datasize request** *size*
13. **timeout** *milliseconds*
14. **tos** *number*
15. **exit**
16. **ipsla schedule operation** *op-num*
17. **life** {**forever** | *seconds*}
18. **ageout** *seconds*
19. **recurring**
20. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
21. 次のいずれかのコマンドを使用します。
    - **end**
    - **commit**
22. **show ipsla statistics** [*operation-number*]
23. **show ipsla statistics aggregated** [*operation-number*]

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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。

	コマンドまたはアクション	目的
ステップ 2	**ipsla operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla operation 432`	動作番号を指定します。 範囲は 1 ～ 2048 です。
ステップ 3	**type udp jitter** 例： `RP/0/RSP0/CPU0:router(config-ipsla-op)# type udp jitter`	動作を UDP ジッター動作として設定し、動作の特性を設定します。
ステップ 4	**vrf** *vrf-name* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-jitter)# vrf VPN-A`	（任意）UDP ジッター動作の中で、デフォルト以外のルーティングテーブルを使用して VPN のモニタリングをイネーブルにします。 最大 32 文字の英数字です。
ステップ 5	**destination address** *ipv4address* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-jitter)# destination address 12.25.26.10`	正しい動作タイプの宛先の IP アドレスを指定します。
ステップ 6	**destination port** *port* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-jitter)# destination port 11111`	宛先ポート番号を指定します。範囲は 1 ～ 65535 です。
ステップ 7	**frequency** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-jitter)# frequency 300`	（任意）指定した IP SLA 動作がネットワークに送信されるレートを設定します。 • （任意）*seconds* 引数を使用して、IP SLA 動作間隔の秒数を指定します。 有効な値の範囲は 1 ～ 12604800 秒です。 デフォルトは 60 秒です。

ステップ 8	**statistics** [**hourly** | **interval** *seconds*]
	
	例：
	
	`RP/0/RSP0/CPU0:router(config-ipsla-udp-jitter)# statistics hourly`
	`RP/0/RSP0/CPU0:router(config-ipsla-op-stats)#`
		（任意）UDP ジッター動作に対して統計情報収集パラメータを指定します。

	コマンドまたはアクション	目的
ステップ 9	**buckets** *hours* 例： `RP/0/RSP0/CPU0:router(config-ipsla-op-stats)# buckets 10`	（任意）IP SLA 動作用に統計情報を保持する時間を設定します。 このコマンドは、必ず **hourly** キーワードを指定した **statistics** コマンドとともに使用する必要があります。 範囲は 0 ～ 25 時間です。 デフォルト値は 2 時間です。
ステップ 10	**distribution count** *slot* 例： `RP/0/RSP0/CPU0:router(config-ipsla-op-stats)# distribution count 15`	（任意）IP SLA 動作のライフタイム中にホップごとに保持される統計情報の配布数を設定します。 指定できる範囲は 1 ～ 20 です。 デフォルト値は 1 配布です。
ステップ 11	**distribution interval** *interval* 例： `RP/0/RSP0/CPU0:router(config-ipsla-op-stats)# distribution interval 20`	（任意）統計情報の配布ごとのインターバルを設定します。 指定できる範囲は 1 ～ 100 ms です。 デフォルトの値は 20 ms です。
ステップ 12	**datasize request** *size* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-jitter)# datasize request 512`	（任意）動作の要求パケットのペイロードのデータサイズを設定します。 UDP ジッターの場合は、範囲は 16 ～ 1500 バイトです。
ステップ 13	**timeout** *milliseconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-jitter)# timeout 10000`	指定された IP SLA 動作がその要求パケットからの応答を待機する時間を設定します。 • （任意）*milliseconds* 引数を使用して、動作が応答を待機するミリ秒数を指定します。
ステップ 14	**tos** *number* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-jitter)# tos 255`	タイプ オブ サービス番号を指定します。

ステップ 15	**exit**
	
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-udp-jitter)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/CPU0:router(config)#`
		IP SLA コンフィギュレーション モードおよび動作モードを終了し、CLI をグローバル コンフィギュレーション モードに戻します。

	コマンドまたはアクション	目的

ステップ 16	**ipsla schedule operation** *op-num*
	
	例：
	
	`RP/0/RSP0/CPU0:router(config)# ipsla schedule operation 432`
	`RP/0/RSP0/CPU0:router(config-ipsla-sched)#`
		動作の開始時間をスケジューリングします。 基本スケジュールを設定できます。

ステップ 17	**life** {**forever** | *seconds*} 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# life 30`	**forever** キーワードを指定すると、動作を無期限で実行するようにスケジューリングされます。 *seconds* 引数を指定すると、動作のライフタイムが秒単位でスケジューリングされます。 デフォルトの動作のライフタイムは 3600 秒（1 時間）です。
ステップ 18	**ageout** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# ageout 3600`	（任意）情報をアクティブに収集していない場合、動作をメモリに常駐させておく時間を秒数で指定します。 デフォルト値の 0 秒は、動作がタイムアウトしないことを意味します。
ステップ 19	**recurring** 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# recurring`	（任意）動作が毎日指定された時刻に自動的に開始され、指定された期間実行されるように指定します。
ステップ 20	**start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*] 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00`	（任意）動作が開始される時刻を指定します。 ここでは、次のキーワードについて説明します。 • （任意）**pending** キーワードを使用して、動作を保留（未開始）状態にしておくように設定します。 デフォルトは inactive です。 **start-time** コマンドが指定されていない場合、開始時刻が設定されるか、即時スタートを実行するトリガーが発生するまで、情報は収集されません。 • （任意）**now** キーワードを使用して、動作を即時スタートする必要があることを示します。 • （任意）**after** キーワードおよび関連する引数を使用して時刻を指定します。ここで指定した時刻以降に、動作が情報の収集を開始します。
ステップ 21	次のいずれかのコマンドを使用します。	設定変更を保存します。

	コマンドまたはアクション	目的

	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

ステップ 22	**show ipsla statistics** [*operation-number*] 例： `RP/0/RSP0/CPU0:router# show ipsla statistics 432`	現在の統計情報を表示します。
ステップ 23	**show ipsla statistics aggregated** [*operation-number*] 例： `RP/0/RSP0/CPU0:router# show ipsla statistics aggregated 432`	ネットワークのパフォーマンスに関する 1 時間ごとの統計情報（集計データ）を返します。 UDP ジッター動作は、次の 1 時間ごとの統計情報を提供します。 • ジッター統計情報：テレフォニーおよびマルチメディア会議要件を解釈します。 • パケット損失およびパケット シーケンシング統計情報：テレフォニー、マルチメディア会議、ストリーミング メディア、およびその他の低遅延データ要件を解釈します。

	コマンドまたはアクション	目的
		• 単方向遅延統計情報：テレフォニー、マルチメディア会議、およびストリーミング メディア要件を解釈します。

UDP エコー動作のための IP SLA の設定

ネットワーク上の UDP パフォーマンスを測定するには、IP SLA UDP エコー動作を使用します。UDPエコー動作は、ラウンドトリップ遅延時間を測定し、シスコデバイスとシスコ以外のデバイスの間の接続をテストします。 UDPエコー動作の結果は、ビジネスクリティカルアプリケーションでの問題をトラブルシューティングするために役立ちます。

（注） UDP エコー動作では、IP SLA レスポンダが動作するシスコ デバイスか、UDP エコー サービスが動作する非シスコ デバイスが必要です。

基本的な UDP エコー動作を設定するのか、オプションのパラメータを使用した UDP エコー動作を設定するのかに応じて、次のいずれかのタスクを実行します。

送信元デバイスでの UDP エコー動作の設定のための前提条件

IP SLA Responder を使用する場合は、宛先デバイスでの IP SLA レスポンダのイネーブル化, （250 ページ）セクションを完了しておきます。

送信元デバイスでの UDP エコー動作の設定およびスケジューリング

オプション パラメータを指定せずに UDP エコー動作をイネーブルにできます。

手順の概要

1. **configure**
2. **ipsla operation** *operation-number*
3. **type udp echo**
4. **destination address** *ipv4address*
5. **destination port** *port*
6. **frequency** *seconds*
7. **exit**
8. **ipsla schedule operation** *op-num*
9. **life** [**forever** | *seconds*]
10. **ageout** *seconds*
11. **recurring**
12. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
13. 次のいずれかのコマンドを使用します。
    - **end**
    - **commit**
14. **show ipsla statistics** [*operation-number*]
15. **show ipsla statistics aggregated** [*operation-number*]

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla operation 432`	動作番号を指定します。 範囲は 1 ～ 2048 です。
ステップ 3	**type udp echo** 例： `RP/0/RSP0/CPU0:router(config-ipsla-op)# type udp echo`	動作を UDP エコー動作として設定し、動作の特性を設定します。

	コマンドまたはアクション	目的
ステップ 4	**destination address** *ipv4address* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-echo)# destination address 12.25.26.10`	正しい動作タイプの宛先の IP アドレスを指定します。IP SLA レスポンダ側の永続的ポートを設定するか、UDP エコー サーバを使用できます。
ステップ 5	**destination port** *port* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-echo)# destination port 11111`	宛先ポート番号を指定します。範囲は 1 ～ 65535 です。
ステップ 6	**frequency** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-echo)# frequency 300`	（任意）指定した IP SLA 動作がネットワークに送信されるレートを設定します。 • （任意）*seconds* 引数を使用して、IP SLA 動作間隔の秒数を指定します。 有効な値の範囲は 1 ～ 12604800 秒です。 デフォルトは 60 秒です。

ステップ 7	**exit**
	
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-udp-echo)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/CPU0:router(config)#`
		IP SLA 動作コンフィギュレーション モードおよび IP SLA コンフィギュレーションモードを終了します。グローバル コンフィギュレーション モードに戻ります。

ステップ 8	**ipsla schedule operation** *op-num*
	
	例：
	`RP/0/RSP0/CPU0:router(config)# ipsla schedule operation 432`
	`RP/0/RSP0/CPU0:router(config-ipsla-sched)#`
		動作の開始時間をスケジューリングします。 基本スケジュールを設定できます。

ステップ 9	**life** [**forever** | *seconds*] 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# life 1`	**forever** キーワードを指定すると、動作を無期限で実行するようにスケジューリングされます。 *seconds* 引数を指定すると、動作のライフタイムが秒単位でスケジューリングされます。 デフォルトの動作のライフタイムは 3600 秒（1 時間）です。
ステップ 10	**ageout** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# ageout 3600`	（任意）情報をアクティブに収集していない場合、動作をメモリに常駐させておく時間を秒数で指定します。デフォルト値の 0 秒は、動作がタイムアウトしないことを意味します。

	コマンドまたはアクション	目的
ステップ 11	**recurring** 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# recurring`	（任意）動作が毎日指定された時刻に自動的に開始され、指定された期間実行されるように指定します。
ステップ 12	**start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*] 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00`	（任意）動作が開始される時刻を指定します。 ここでは、次のキーワードについて説明します。 • （任意）**pending** キーワードを使用して、動作を保留（未開始）状態にしておくように設定します。768 ビットは、デフォルト値です。 **start-time** コマンドが指定されていない場合、開始時刻が設定されるか、即時スタートを実行するトリガーが発生するまで、情報は収集されません。 • （任意）**now** キーワードを使用して、動作を即時スタートする必要があることを示します。 • （任意）**after** キーワードおよび関連する引数を使用して時刻を指定します。ここで指定した時刻以降に、動作が情報の収集を開始します。

ステップ 13	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		
		◦ yes と入力すると、実行コンフィギュレーションファイルに変更が保存され、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。

	コマンドまたはアクション	目的
		• 設定変更を実行コンフィギュレーションファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。
ステップ 14	**show ipsla statistics** [*operation-number*] 例： `RP/0/RSP0/CPU0:router# show ipsla statistics 432`	現在の統計情報を表示します。
ステップ 15	**show ipsla statistics aggregated** [*operation-number*] 例： `RP/0/RSP0/CPU0:router# show ipsla statistics aggregated 1`	1 時間ごとの統計エラーと、すべての IP SLA 動作または指定した動作の 1 時間ごとの統計情報を表示します。

任意のパラメータを指定した、送信元デバイスでの UDP エコー動作の設定およびスケジューリング

送信元デバイスで UDP エコー動作をイネーブルにして、省略可能な IP SLA パラメータを設定できます。 送信元デバイスは、測定統計情報が保存される場所です。

手順の概要

1. **configure**
2. **ipsla operation** *operation-number*
3. **type udp echo**
4. **vrf** *vrf-name*
5. **destination address** *ipv4address*
6. **destination port** *port*
7. **frequency** *seconds*
8. **datasize request** *size*
9. **tos** *number*
10. **timeout** *milliseconds*
11. **tag** *text*
12. **exit**
13. **ipsla schedule operation** *op-num*
14. **life** {**forever** | *seconds*}
15. **ageout** *seconds*
16. **recurring**
17. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
18. 次のいずれかのコマンドを使用します。
    - **end**
    - **commit**
19. **show ipsla statistics enhanced aggregated** [*operation-number*] **interval** *seconds*
20. **show ipsla statistics** [*operation-number*]

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla operation 432`	動作番号を指定します。 範囲は 1 ～ 2048 です。

	コマンドまたはアクション	目的
ステップ 3	**type udp echo** 例： `RP/0/RSP0/CPU0:router(config-ipsla-op)# type udp echo`	動作をUDPエコー動作として設定し、動作の特性を設定します。
ステップ 4	**vrf** *vrf-name* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-echo)# vrf VPN-A`	（任意）UDP エコー動作の中で、デフォルト以外のルーティング テーブルを使用して VPN のモニタリングをイネーブルにします。 最大32文字の英数字です。
ステップ 5	**destination address** *ipv4address* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-echo)# destination address 12.25.26.10`	正しい動作タイプの宛先のIPアドレスを指定します。
ステップ 6	**destination port** *port* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-echo)# destination port 11111`	宛先ポート番号を指定します。範囲は1～65535です。
ステップ 7	**frequency** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-echo)# frequency 300`	（任意）指定したIP SLA動作がネットワークに送信されるレートを設定します。 • （任意）*seconds* 引数を使用して、IP SLA 動作間隔の秒数を指定します。 有効な値の範囲は 1 ～ 12604800 秒です。 デフォルトは 60 秒です。
ステップ 8	**datasize request** *size* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-echo)# datasize request 512`	（任意）IP SLA 動作の要求パケットのペイロードにおけるプロトコル データ サイズを設定します。 • プロトコル データ サイズ（バイト単位）を指定するには、*size* 引数を使用します。 範囲は 0 ～プロトコルの最大サイズです。 デフォルト値は1バイトです。
ステップ 9	**tos** *number* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-echo)# tos 255`	IP SLA 動作の IP ヘッダーに、タイプ オブ サービス（ToS）バイトを定義します。 （注） ToS バイトは DiffServ コード ポイント（DSCP）値に変換されますが、DSCP 値を直接入力することはできません。 DSCP値を使用するには、それに 4 を掛けて、結果を *number* 引数の値として入力します。

	コマンドまたはアクション	目的
ステップ 10	**timeout** *milliseconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-echo)# timeout 10000`	指定された IP SLA 動作がその要求パケットからの応答を待機する時間を設定します。 • *milliseconds* 引数を使用して、動作が応答を待機するミリ秒数を指定します。
ステップ 11	**tag** *text* 例： `RP/0/RSP0/CPU0:router(config-ipsla-udp-echo)# type udp echo tag ipsla`	（任意）IP SLA 動作のユーザ指定 ID を作成します。

ステップ 12	**exit**
	
	例：
	
	`RP/0/RSP0/CPU0:router(config-ipsla-udp-echo)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/CPU0:router(config)#`
		IP SLA 動作コンフィギュレーション モードおよび IPSLA コンフィギュレーション モードを終了します。グローバル コンフィギュレーション モードに戻ります。

ステップ 13	**ipsla schedule operation** *op-num*
	
	例：
	
	`RP/0/RSP0/CPU0:router(config)# ipsla schedule operation 432`
	`RP/0/RSP0/CPU0:router(config-ipsla-sched)#`
		動作の開始時間をスケジューリングします。 基本的なスケジュールを設定するか、グループ スケジューリングを使用して複数の動作をスケジューリングできます。

ステップ 14	**life** {**forever** | *seconds*} 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# life 30`	**forever** キーワードを指定すると、動作を無期限で実行するようにスケジューリングされます。 *seconds* 引数を指定すると、動作のライフタイムが秒単位でスケジューリングされます。 デフォルトの動作のライフタイムは 3600 秒（1 時間）です。
ステップ 15	**ageout** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# ageout 3600`	（任意）情報をアクティブに収集していない場合、動作をメモリに常駐させておく時間を秒数で指定します。デフォルト値の 0 秒は、動作がタイムアウトしないことを意味します。
ステップ 16	**recurring** 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# recurring`	（任意）動作が毎日指定された時刻に自動的に開始され、指定された期間実行されるように指定します。

	コマンドまたはアクション	目的
ステップ 17	**start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*] 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00`	動作が開始される時刻を指定します。 ここでは、次のキーワードについて説明します。 • （任意）**pending** キーワードを使用して、動作を保留（未開始）状態にしておくように設定します。 デフォルト値は inactive です。 **start-time** コマンドが指定されていない場合、開始時刻が設定されるか、即時スタートを実行するトリガーが発生するまで、情報は収集されません。 • （任意）**now** キーワードを使用して、動作を即時スタートする必要があることを示します。 • （任意）**after** キーワードおよび関連する引数を使用して時刻を指定します。ここで指定した時刻以降に、動作が情報の収集を開始します。

ステップ 18	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

	コマンドまたはアクション	目的
ステップ 19	**show ipsla statistics enhanced aggregated** [*operation-number*] **interval** *seconds* 例： `RP/0/RSP0/CPU0:router# show ipsla statistics enhanced aggregated 432`	拡張された履歴統計情報を表示します。 サンプル出力を表示するには、拡張された履歴統計情報を設定する必要があります。
ステップ 20	**show ipsla statistics** [*operation-number*] 例： `RP/0/RSP0/CPU0:router# show ipsla statistics 432`	現在の統計情報を表示します。

ICMP エコー動作の設定

デバイス上の IP 接続をモニタするには、IP SLA ICMP エコー動作を使用します。 ICMP エコー動作は、シスコ ルータと IP を使用するデバイスの間のエンドツーエンド応答時間を測定します。 ICMPエコーは、ネットワークの接続上の問題をトラブルシューティングするために使用します。

（注） ICMP エコー動作では、IP SLA レスポンダをイネーブルにする必要はありません。

基本的な ICMP エコー動作を設定およびスケジューリングするのか、省略可能なパラメータを使用した ICMP エコー動作を設定およびスケジューリングするのかに応じて、次のいずれかの手順を実行します。

送信元デバイスでの基本の ICMP エコー動作の設定およびスケジューリング

オプション パラメータを指定せずに ICMP エコー動作をイネーブルにしてスケジューリングできます。

手順の概要

1. **configure**
2. **ipsla operation** *operation-number*
3. **type icmp echo**
4. **destination address** *ipv4address*
5. **frequency** *seconds*
6. **exit**
7. **ipsla schedule operation** *op-num*
8. **life** {**forever** | *seconds*}
9. **ageout** *seconds*
10. **recurring**
11. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
12. 次のいずれかのコマンドを使用します。
    - **end**
    - **commit**
13. **show ipsla statistics** [*operation-number*]

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla operation 432`	動作番号を指定します。 範囲は 1 ～ 2048 です。
ステップ 3	**type icmp echo** 例： `RP/0/RSP0/CPU0:router(config-ipsla-op)# type icmp echo`	ICMP エコー動作タイプを定義します。
ステップ 4	**destination address** *ipv4address* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-echo)# destination address 12.25.26.10`	正しい動作タイプの宛先の IP アドレスを指定します。

	コマンドまたはアクション	目的
ステップ 5	**frequency** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-echo) frequency 300`	（任意）指定した IP SLA 動作がネットワークに送信されるレートを設定します。 • （任意）*seconds* 引数を使用して、IP SLA 動作間隔の秒数を指定します。 有効な値の範囲は 1 ～ 12604800 秒です。 デフォルトは 60 秒です。

ステップ 6	**exit**
	
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-icmp-echo)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/CPU0:router(config)#`
		IP SLA 動作コンフィギュレーション モードおよび IP SLA コンフィギュレーション モードを終了します。 グローバル コンフィギュレーション モードに戻ります。

ステップ 7	**ipsla schedule operation** *op-num*
	
	例：
	`RP/0/RSP0/CPU0:router(config)# ipsla schedule operation 432`
	`RP/0/RSP0/CPU0:router(config-ipsla-sched)#`
		動作の開始時間をスケジューリングします。 基本スケジュールを設定できます。

ステップ 8	**life** {**forever** | *seconds*} 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# life 30`	**forever** キーワードを指定すると、動作を無期限で実行するようにスケジューリングされます。 *seconds* 引数を指定すると、動作のライフタイムが秒単位でスケジューリングされます。デフォルトの動作のライフタイムは 3600 秒（1 時間）です。
ステップ 9	**ageout** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# ageout 3600`	（任意）情報をアクティブに収集していない場合、動作をメモリに常駐させておく時間を秒数で指定します。デフォルト値の 0 秒は、動作がタイムアウトしないことを意味します。
ステップ 10	**recurring** 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# recurring`	（任意）動作が毎日指定された時刻に自動的に開始され、指定された期間実行されるように指定します。
ステップ 11	**start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*] 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00`	動作が開始される時刻を指定します。 ここでは、次のキーワードについて説明します。 • （任意）**pending** キーワードを使用して、動作を保留（未開始）状態にしておくように設定します。 デフォルト値は inactive です。 **start-time** コマンドが指定されていない場合、開始時刻が設定

	コマンドまたはアクション	目的
		されるか、即時スタートを実行するトリガーが発生するまで、情報は収集されません。 • （任意）**now** キーワードを使用して、動作を即時スタートする必要があることを示します。 • （任意）**after** キーワードおよび関連する引数を使用して時刻を指定します。ここで指定した時刻以降に、動作が情報の収集を開始します。

ステップ 12	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーションファイルに変更が保存され、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

ステップ 13	**show ipsla statistics** [*operation-number*] 例： `RP/0/RSP0/CPU0:router# show ipsla statistics 432`	現在の統計情報を表示します。

送信元デバイスでの省略可能なパラメータを使用した ICMP エコー動作の設定およびスケジューリング

送信元デバイスで ICMP エコー動作をイネーブルにして、省略可能な IP SLA パラメータを設定できます。

手順の概要

1. **configure**
2. **ipsla operation** *operation-number*
3. **type icmp echo**
4. **vrf** *vrf-name*
5. **destination address** *ipv4address*
6. **frequency** *seconds*
7. **datasize request** *size*
8. **tos** *number*
9. **timeout** *milliseconds*
10. **tag** *text*
11. exit
12. **ipsla schedule operation** *op-num*
13. **life** {**forever** | *seconds*}
14. **ageout** *seconds*
15. **recurring**
16. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
17. 次のいずれかのコマンドを使用します。
    - **end**
    - **commit**
18. **show ipsla statistics** [*operation-number*]

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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。

	コマンドまたはアクション	目的
ステップ 2	**ipsla operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla operation 432`	動作番号を指定します。 範囲は 1 ～ 2048 です。
ステップ 3	**type icmp echo** 例： `RP/0/RSP0/CPU0:router(config-ipsla-op)# type icmp echo`	ICMP エコー動作タイプを定義します。
ステップ 4	**vrf** *vrf-name* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-echo)# vrf VPN-A`	（任意）ICMP エコー動作の中で、デフォルト以外のルーティング テーブルを使用して VPN のモニタリングをイネーブルにします。 最大32 文字の英数字です。
ステップ 5	**destination address** *ipv4address* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-echo)# destination address 12.25.26.10`	正しい動作タイプの宛先の IP アドレスを指定します。
ステップ 6	**frequency** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-echo)# frequency 300`	（任意）指定した IP SLA 動作がネットワークに送信されるレートを設定します。 • （任意）*seconds* 引数を使用して、IP SLA 動作間隔の秒数を指定します。 有効な値の範囲は 1 ～ 12604800 秒です。 デフォルトは 60 秒です。
ステップ 7	**datasize request** *size* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-echo)# datasize request 512`	（任意）指定した IP SLA 動作の要求パケットのペイロードにおけるプロトコル データ サイズを設定します。 • プロトコル データ サイズ（バイト単位）を指定するには、*bytes* 引数を使用します。 範囲は 0 ～ 16384 です。 ICMP エコー動作のデフォルトは 36 バイトです。
ステップ 8	**tos** *number* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-echo)# tos 1`	IP SLA 動作の IP ヘッダーに、タイプ オブ サービス（ToS）バイトを定義します。

	コマンドまたはアクション	目的
		（注） ToS バイトは DiffServ コード ポイント（DSCP）値に変換できますが、DSCP 値を直接入力することはできません。 DSCP 値を使用するには、それに 4 を掛けて、結果を *number* 引数の値として入力します。
ステップ 9	**timeout** *milliseconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-echo)# timeout 10000`	IP SLA 動作がその要求パケットからの応答を待機する時間を設定します。 • *milliseconds* 引数を使用して、動作が応答を待機するミリ秒数を指定します。
ステップ 10	**tag** *text* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-echo)# tag ipsla`	（任意）IP SLA 動作のユーザ指定 ID を作成します。

ステップ 11	exit
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-icmp-echo)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/CPU0:router(config)#`
		IP SLA 動作コンフィギュレーション モードおよび IP SLA コンフィギュレーション モードを終了します。グローバル コンフィギュレーション モードに戻ります。

ステップ 12	**ipsla schedule operation** *op-num*
	例：
	`RP/0/RSP0/CPU0:router(config)# ipsla schedule operation 432`
	`RP/0/RSP0/CPU0:router(config-ipsla-sched)#`
		動作の開始時間をスケジューリングします。 基本スケジュールを設定できます。

ステップ 13	**life** {**forever** | *seconds*} 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# life 30`	**forever** キーワードを指定すると、動作を無期限で実行するようにスケジューリングされます。 *seconds* 引数を指定すると、動作のライフタイムが秒単位でスケジューリングされます。デフォルトの動作のライフタイムは 3600 秒（1 時間）です。
ステップ 14	**ageout** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# ageout 3600`	（任意）情報をアクティブに収集していない場合、動作をメモリに常駐させておく時間を秒数で指定します。デフォルト値の 0 秒は、動作がタイムアウトしないことを意味します。

	コマンドまたはアクション	目的
ステップ 15	**recurring** 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# recurring`	（任意）動作が毎日指定された時刻に自動的に開始され、指定された期間実行されるように指定します。
ステップ 16	**start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*] 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00`	動作が開始される時刻を指定します。 ここでは、次のキーワードについて説明します。 • （任意）**pending** キーワードを使用して、動作を保留（未開始）状態にしておくように設定します。 デフォルト値は inactive です。 **start-time** コマンドが指定されていない場合、開始時刻が設定されるか、即時スタートを実行するトリガーが発生するまで、情報は収集されません。 • （任意）**now** キーワードを使用して、動作を即時スタートする必要があることを示します。 • （任意）**after** キーワードおよび関連する引数を使用して時刻を指定します。ここで指定した時刻以降に、動作が情報の収集を開始します。

ステップ 17	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーションファイルに変更が保存され、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッショ

	コマンドまたはアクション	目的
		ンを継続するには、**commit** コマンドを使用します。
ステップ 18	**show ipsla statistics** [*operation-number*] 例： `RP/0/RSP0/CPU0:router# show ipsla statistics 432`	現在の統計情報を表示します。

ICMP パスエコー動作の設定

IP SLA ICMP パスエコー動作は、IP SLA 動作が宛先に到達するためにたどるパスに沿った各ホップの統計情報を記録します。 ICMP パスエコー動作では、traceroute 機能を使用してパスを検出することにより、Cisco ルータとネットワーク上の IP デバイスの間のホップバイホップ応答時間が判断されます。

送信元 IP SLA デバイスは、traceroute を使用して宛先 IP デバイスへのパスを検出します。 その後、ping を使用して、送信元 IP SLA デバイスと、宛先 IP デバイスへのパス中の以降の各ホップの間の応答時間が測定されます。

（注） ICMP パスエコー動作では、IP SLA レスポンダをイネーブルにする必要はありません。

基本的な ICMP パスエコー動作を設定およびスケジューリングするのか、省略可能なパラメータを使用した ICMP パスエコー動作を設定およびスケジューリングするのかに応じて、次のいずれかの手順を実行します。

送信元デバイスでの基本の ICMP パスエコー動作の設定およびスケジューリング

オプション パラメータを指定せずに ICMP パスエコー動作をイネーブルにしてスケジューリングできます。

手順の概要

1. **configure**
2. **ipsla operation** *operation-number*
3. **type icmp path-echo**
4. **destination address** *ipv4address*
5. **frequency** *seconds*
6. **exit**
7. **ipsla schedule operation** *op-num*
8. **life** {**forever** | *seconds*}
9. **ageout** *seconds*
10. **recurring**
11. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
12. 次のいずれかのコマンドを使用します。
    - **end**
    - **commit**
13. **show ipsla statistics** [*operation-number*]

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla operation 432`	動作番号を指定します。 範囲は 1 ～ 2048 です。

ステップ 3	**type icmp path-echo**
	
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# type icmp path-echo`
	`RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-echo)#`
		ICMP パスエコー動作タイプを定義します。

	コマンドまたはアクション	目的
ステップ 4	**destination address** *ipv4address* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-echo)# destination address 12.25.26.10`	正しい動作タイプの宛先の IP アドレスを指定します。
ステップ 5	**frequency** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-echo)# frequency 300`	（任意）指定した IP SLA 動作がネットワークに送信されるレートを設定します。 • （任意）*seconds* 引数を使用して、IP SLA 動作間隔の秒数を指定します。 有効な値の範囲は 1 ～ 12604800 秒です。 デフォルトは 60 秒です。

ステップ 6	**exit**
	
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-echo)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/CPU0:router(config)#`
		IP SLA 動作コンフィギュレーション モードおよび IP SLA コンフィギュレーション モードを終了します。 グローバル コンフィギュレーション モードに戻ります。

ステップ 7	**ipsla schedule operation** *op-num*
	
	例：
	`RP/0/RSP0/CPU0:router(config)# ipsla schedule operation 432`
	`RP/0/RSP0/CPU0:router(config-ipsla-sched)#`
		動作の開始時間をスケジューリングします。 基本スケジュールを設定できます。

ステップ 8	**life** {**forever** | *seconds*} 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# life 30`	**forever** キーワードを指定すると、動作を無期限で実行するようにスケジューリングされます。 *seconds* 引数を指定すると、動作のライフタイムが秒単位でスケジューリングされます。 デフォルトの動作のライフタイムは 3600 秒（1 時間）です。
ステップ 9	**ageout** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# ageout 3600`	（任意）情報をアクティブに収集していない場合、動作をメモリに常駐させておく時間を秒数で指定します。 デフォルト値の 0 秒は、動作がタイムアウトしないことを意味します。
ステップ 10	**recurring** 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# recurring`	（任意）動作が毎日指定された時刻に自動的に開始され、指定された期間実行されるように指定します。

	コマンドまたはアクション	目的
ステップ 11	**start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*] 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00`	動作が開始される時刻を指定します。 ここでは、次のキーワードについて説明します。 • （任意）**pending** キーワードを使用して、動作を保留（未開始）状態にしておくように設定します。 デフォルト値は inactive です。 **start-time** コマンドが指定されていない場合、開始時刻が設定されるか、即時スタートを実行するトリガーが発生するまで、情報は収集されません。 • （任意）**now** キーワードを使用して、動作を即時スタートする必要があることを示します。 • （任意）**after** キーワードおよび関連する引数を使用して時刻を指定します。ここで指定した時刻以降に、動作が情報の収集を開始します。

ステップ 12	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーションファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。

	コマンドまたはアクション	目的
		• 設定変更を実行コンフィギュレーションファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。
ステップ 13	**show ipsla statistics** [*operation-number*] 例： `RP/0/RSP0/CPU0:router# show ipsla statistics 432`	現在の統計情報を表示します。

送信元デバイスでの省略可能なパラメータを使用した ICMP パスエコー動作の設定およびスケジューリング

送信元デバイスで ICMP パスエコー動作をイネーブルにして、省略可能な IP SLA パラメータを設定できます。

手順の概要

1. **configure**
2. **ipsla operation** *operation-number*
3. **type icmp path-echo**
4. **vrf** *vrf-name*
5. **lsr-path** *ip-address*
6. **destination address** *ipv4address*
7. **frequency** *seconds*
8. **datasize request** *size*
9. **tos** *number*
10. **timeout** *milliseconds*
11. **tag** *text*
12. **lsr-path** *ipaddress1* {*ipaddress2* {... {*ipaddress8*}}}
13. **exit**
14. **ipsla schedule operation** *op-num*
15. **life** {**forever** | *seconds*}
16. **ageout** *seconds*
17. **recurring**
18. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
19. 次のいずれかのコマンドを使用します。
    - **end**
    - **commit**
20. **show ipsla statistics** [*operation-number*]

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla operation 432`	動作番号を指定します。 範囲は 1 ～ 2048 です。

	コマンドまたはアクション	目的

ステップ 3	**type icmp path-echo**
	
	例：
	
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# type icmp path-echo`
	`RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-echo)#`
		ICMP パスエコー動作タイプを定義します。

ステップ 4	**vrf** *vrf-name* 例： `RP/0/RSP0/CPU0:router(config-ipsla-imcp-path-echo)# vrf VPN-A`	（任意）ICMP パスエコー動作の中で、デフォルト以外のルーティング テーブルを使用して VPN のモニタリングをイネーブルにします。 最大 32 文字の英数字です。
ステップ 5	**lsr-path** *ip-address* 例： `RP/0/RSP0/CPU0:router(config-ipsla-imcp-path-echo)# lsr-path 20.25.22.1`	ルーズ ソース ルーティング パスを使用することを指定します。
ステップ 6	**destination address** *ipv4address* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-echo)# destination address 12.25.26.10`	正しい動作タイプの宛先の IP アドレスを指定します。
ステップ 7	**frequency** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-echo)# frequency 300`	（任意）指定した IP SLA 動作がネットワークに送信されるレートを設定します。 • （任意）*seconds* 引数を使用して、IP SLA 動作間隔の秒数を指定します。 有効な値の範囲は 1 ～ 12604800 秒です。 デフォルトは 60 秒です。
ステップ 8	**datasize request** *size* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-echo)# datasize request 512`	（任意）指定した IP SLA 動作の要求パケットのペイロードにおけるプロトコル データ サイズを設定します。 • プロトコル データ サイズ（バイト単位）を指定するには、*bytes* 引数を使用します。 範囲は 0 ～ 16384 です。 デフォルト値は 36 バイトです。

	コマンドまたはアクション	目的
ステップ 9	**tos** *number* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-echo)# tos 5`	IP SLA 動作の IP ヘッダーに、タイプ オブ サービス（ToS）バイトを定義します。 （注） ToS バイトは DiffServ コード ポイント（DSCP）値に変換できますが、DSCP 値を直接入力することはできません。DSCP 値を使用するには、それに 4 を掛けて、結果を *number* 引数として入力します。
ステップ 10	**timeout** *milliseconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-echo)# timeout 10000`	IP SLA 動作がその要求パケットからの応答を待機する時間を設定します。 • *milliseconds* 引数を使用して、動作が応答を待機するミリ秒数を指定します。
ステップ 11	**tag** *text* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-echo)# tag ipsla`	（任意）IP SLA 動作のユーザ指定 ID を作成します。
ステップ 12	**lsr-path** *ipaddress1* {*ipaddress2* {... {*ipaddress8*}}} 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-echo)# lsr-path 20.25.22.1`	ICMP エコー応答時間を測定するパスを指定します。 • （任意）宛先へのパス中の中間ノードの *ip address* 引数を使用します。

ステップ 13	**exit**
	
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-echo)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/CPU0:router(config)#`
		IP SLA 動作コンフィギュレーション モードおよび IP SLA コンフィギュレーション モードを終了します。グローバル コンフィギュレーション モードに戻ります。

ステップ 14	**ipsla schedule operation** *op-num*
	
	例：
	`RP/0/RSP0/CPU0:router(config)# ipsla schedule operation 432`
	`RP/0/RSP0/CPU0:router(config-ipsla-sched)#`
		動作の開始時間をスケジューリングします。 基本スケジュールを設定できます。

ステップ 15	**life** {**forever** | *seconds*} 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# life 1`	**forever** キーワードを指定すると、動作を無期限で実行するようにスケジューリングされます。*seconds* 引数を指定すると、動作のライフタイムが秒単位でスケジューリングされます。デフォル

	コマンドまたはアクション	目的
		トの動作のライフタイムは 3600 秒（1 時間）です。
ステップ 16	**ageout** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# ageout 3600`	（任意）情報をアクティブに収集していない場合、動作をメモリに常駐させておく時間を秒数で指定します。デフォルト値の 0 秒は、動作がタイムアウトしないことを意味します。
ステップ 17	**recurring** 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# recurring`	（任意）動作が毎日指定された時刻に自動的に開始され、指定された期間実行されるように指定します。
ステップ 18	**start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*] 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00`	動作が開始される時刻を指定します。 ここでは、次のキーワードについて説明します。 • （任意）**pending** キーワードを使用して、動作を保留（未開始）状態にしておくように設定します。 デフォルト値は inactive です。 **start-time** コマンドが指定されていない場合、開始時刻が設定されるか、即時スタートを実行するトリガーが発生するまで、情報は収集されません。 • （任意）**now** キーワードを使用して、動作を即時スタートする必要があることを示します。 • （任意）**after** キーワードおよび関連する引数を使用して時刻を指定します。ここで指定した時刻以降に、動作が情報の収集を開始します。

ステップ 19	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		
		◦ yes と入力すると、実行コンフィギュレーションファイルに変更が保存され、コンフィギュレーション セッションが

	コマンドまたはアクション	目的
		終了して、ルータが EXEC モードに戻ります。 ◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。 ◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。 • 設定変更を実行コンフィギュレーションファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。
ステップ 20	**show ipsla statistics** [*operation-number*] 例： `RP/0/RSP0/CPU0:router# show ipsla statistics 432`	現在の統計情報を表示します。

ICMP パスジッター動作の設定

IP SLA ICMP パスジッター動作は、IP ネットワーク内のホップバイホップ ジッター、パケット損失、および遅延測定統計情報を提供します。 パスジッター動作は、一方向データの総計と往復データの総計を提供する標準的な UDP ジッター動作とは異なる機能を果たします。

ICMP パスジッター動作は、標準的な UDP ジッター動作を補完するものとして使用できます。 たとえば、UDP ジッター動作から得られた結果が予期しない遅延や高いジッター値を示すことがあります。この場合に ICMP パスジッター動作を使用すると、ネットワーク パスのトラブルシューティングを行い、伝送パス沿いの特定のセグメントでトラフィックが渋滞していないかどうかを確認できます。

ICMP パス ジッター動作は、まず traceroute ユーティリティを使用して送信元から宛先までのホップバイホップ IP ルートを検出し、次に ICMP エコーを使用して、パス沿いの各ホップの応答時間、パケット損失、およびジッターの概算値を測定します。 ICMP パスジッター動作を使用して得られたジッター値は、ターゲット ノードでの遅延が考慮されていないため、近似値です。

ICMP パスジッター動作は、送信元デバイスから指定した宛先デバイスまでの IP パスをトレースし、次にそのトレース パス沿いの各ホップに N 個のエコー プローブを T ミリ秒間隔で送信します。 動作全体は、F 秒ごとに 1 回の頻度で繰り返されます。 次の表に示すように、属性はユーザ設定可能です。

表 *24* : *ICMP* パスジッター動作のパラメータ

ICMP パスジッター動作のパラメータ	デフォルト	設定方法
エコー プローブの数（N）	10 個のエコー	• *operation-number* 引数を指定した **ipsla operation** コマンド • *count* 引数を指定した **packet count** コマンド
エコー プローブ間隔（ミリ秒単位）（T）	20 ms	• *operation-number* 引数を指定した **ipsla operation** コマンド • *interval* 引数を指定した **packet interval** コマンド
動作の繰り返し頻度（F）	60 秒に 1 回	• *operation-number* 引数を指定した **ipsla operation** コマンド • *seconds* 引数を指定した **frequency** コマンド

基本的な ICMP パスジッター動作を設定およびスケジューリングするのか、追加のパラメータを使用した ICMP ジッター動作を設定およびスケジューリングするのかに応じて、次のいずれかの手順を実行します。

基本的な ICMP パスジッター動作の設定およびスケジューリング

動作の一般的なデフォルト特性を使用して ICMP パスジッター動作を設定およびスケジューリングできます。

手順の概要

1. **configure**
2. **ipsla operation** *operation-number*
3. **type icmp path-jitter**
4. **destination address** *ipv4address*
5. **packet count** *count*
6. **packet interval** *interval*
7. **frequency** *seconds*
8. exit
9. **ipsla schedule operation** *op-num*
10. **life** {**forever** | *seconds*}
11. **ageout** *seconds*
12. **recurring**
13. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
14. 次のいずれかのコマンドを使用します。
    - **end**
    - **commit**
15. **show ipsla statistics** [*operation-number*]

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla operation 432`	動作番号を指定します。範囲は 1 ～ 2048 です。
ステップ 3	**type icmp path-jitter** 例： `RP/0/RSP0/CPU0:router(config-ipsla-op)# type icmp path-jitter`	ICMP パスジッター動作タイプを定義します。

	コマンドまたはアクション	目的
ステップ 4	**destination address** *ipv4address* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-jitter)# destination address 12.25.26.10`	正しい動作タイプの宛先の IP アドレスを指定します。
ステップ 5	**packet count** *count* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-jitter)# packet count 30`	（任意）プローブ中に送信されるパケット数を指定します。 UDP ジッター動作の場合、範囲は 1 ～ 60000 です。 ICMP パスジッター動作の場合、範囲は 1 ～ 100 です。 送信されるデフォルトのパケット数は 10 です。
ステップ 6	**packet interval** *interval* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-jitter)# packet interval 30`	（任意）パケット間隔を指定します。 パケット間のデフォルト間隔は 20 ミリ秒です。
ステップ 7	**frequency** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-jitter)# frequency 300`	（任意）指定した IP SLA 動作がネットワークに送信されるレートを設定します。 • （任意）*seconds* 引数を使用して、IP SLA 動作間隔の秒数を指定します。 有効な値の範囲は 1 ～ 12604800 秒です。 デフォルトは 60 秒です。

ステップ 8	exit
	
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-jitter)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/CPU0:router(config)#`
		IP SLA 動作コンフィギュレーション モードおよび IP SLA コンフィギュレーション モードを終了します。 グローバル コンフィギュレーション モードに戻ります。

ステップ 9	**ipsla schedule operation** *op-num*
	
	例：
	`RP/0/RSP0/CPU0:router(config)# ipsla schedule operation 432`
	`RP/0/RSP0/CPU0:router(config-ipsla-sched)#`
		動作の開始時間をスケジューリングします。 基本スケジュールを設定できます。

ステップ 10	**life** {**forever** | *seconds*} 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# life 30`	**forever** キーワードを指定すると、動作を無期限で実行するようにスケジューリングされます。 *seconds* 引数を指定すると、動作のライフタイムが秒単位でスケジューリングされます。 デフォ

	コマンドまたはアクション	目的
		ルトの動作のライフタイムは 3600 秒（1 時間）です。
ステップ 11	**ageout** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# ageout 3600`	（任意）情報をアクティブに収集していない場合、動作をメモリに常駐させておく時間を秒数で指定します。 デフォルト値の 0 秒は、動作がタイムアウトしないことを意味します。
ステップ 12	**recurring** 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# recurring`	（任意）動作が毎日指定された時刻に自動的に開始され、指定された期間実行されるように指定します。
ステップ 13	**start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*] 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00`	（任意）動作が開始される時刻を指定します。ここでは、次のキーワードについて説明します。 • （任意）**pending** キーワードを使用して、動作を保留（未開始）状態にしておくように設定します。 デフォルト値は inactive です。 **start-time** コマンドが指定されていない場合、開始時刻が設定されるか、即時スタートを実行するトリガーが発生するまで、情報は収集されません。 • （任意）**now** キーワードを使用して、動作を即時スタートする必要があることを示します。 • （任意）**after** キーワードおよび関連する引数を使用して時刻を指定します。ここで指定した時刻以降に、動作が情報の収集を開始します。

ステップ 14	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーションセッショ

	コマンドまたはアクション	目的
		ンが終了して、ルータがEXECモードに戻ります。 ◦no と入力すると、コンフィギュレーションセッションが終了して、ルータがEXECモードに戻ります。変更はコミットされません。 ◦cancel と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。 • 設定変更を実行コンフィギュレーションファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。
ステップ **15**	**show ipsla statistics** [*operation-number*] 例： `RP/0/RSP0/CPU0:router# show ipsla statistics 432`	現在の統計情報を表示します。

追加パラメータを指定した ICMP パスジッター動作の設定およびスケジューリング

送信元デバイスで ICMP パスエコー動作をイネーブルにして、省略可能な IP SLA パラメータを設定できます。

手順の概要

1. **configure**
2. **ipsla operation** *operation-number*
3. **type icmp path-jitter**
4. **vrf** *vrf-name*
5. **lsr-path** *ip-address*
6. **destination address** *ipv4address*
7. **packet count** *count*
8. **packet interval** *interval*
9. **frequency** *seconds*
10. **datasize request** *size*
11. **tos** *number*
12. **timeout** *milliseconds*
13. **tag** *text*
14. **exit**
15. **ipsla schedule operation** *op-num*
16. **life** {**forever** | *seconds*}
17. **ageout** *seconds*
18. **recurring**
19. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
20. 次のいずれかのコマンドを使用します。
    - **end**
    - **commit**
21. **show ipsla statistics** [*operation-number*]

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla operation 432`	動作番号を指定します。範囲は 1 ～ 2048 です。

	コマンドまたはアクション	目的
ステップ 3	**type icmp path-jitter** 例： `RP/0/RSP0/CPU0:router(config-ipsla-op)# type icmp path-jitter`	ICMP パスジッター動作タイプを定義します。
ステップ 4	**vrf** *vrf-name* 例： `RP/0/RSP0/CPU0:router(config-ipsla-imcp-path-jitter)# vrf VPN-A`	（任意）ICMP パスジッター動作の中で、デフォルト以外のルーティング テーブルを使用して VPN のモニタリングをイネーブルにします。 最大 32 文字の英数字です。
ステップ 5	**lsr-path** *ip-address* 例： `RP/0/RSP0/CPU0:router(config-ipsla-imcp-path-jitter)# lsr-path 20.25.22.1`	ルーズ ソース ルーティング パスを使用することを指定します。
ステップ 6	**destination address** *ipv4address* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-jitter)# destination address 12.25.26.10`	正しい動作タイプの宛先の IP アドレスを指定します。
ステップ 7	**packet count** *count* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-jitter)# packet count 30`	（任意）プローブ中に送信されるパケット数を指定します。 UDP ジッター動作の場合、範囲は 1 ～ 60000 です。 ICMP パスジッター動作の場合、範囲は 1 ～ 100 です。 送信されるデフォルトのパケット数は 10 です。
ステップ 8	**packet interval** *interval* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-jitter)# packet interval 30`	（任意）パケット間隔を指定します。 パケット間のデフォルト間隔は 20 ミリ秒です。
ステップ 9	**frequency** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-jitter)# frequency 300`	（任意）指定した IP SLA 動作がネットワークに送信されるレートを設定します。 • （任意）*seconds* 引数を使用して、IP SLA 動作間隔の秒数を指定します。 有効な値の範囲は 1 ～ 12604800 秒です。 デフォルトは 60 秒です。

	コマンドまたはアクション	目的
ステップ 10	**datasize request** *size* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-jitter)# datasize request 512`	（任意）指定した IP SLA 動作の要求パケットのペイロードにおけるプロトコルデータ サイズを設定します。 • プロトコル データ サイズ（バイト単位）を指定するには、*size* 引数を使用します。ジッターのデフォルトは 36 バイトです。有効な範囲は 0 ～ 16384 バイトです。
ステップ 11	**tos** *number* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-jitter)# tos 1`	IP SLA 動作の IP ヘッダーに、タイプ オブ サービス（ToS）バイトを定義します。 （注） ToS バイトは DiffServ コード ポイント（DSCP）値に変換できますが、DSCP 値を直接入力することはできません。DSCP 値を使用するには、それに 4 を掛けて、結果を *number* 引数として入力します。
ステップ 12	**timeout** *milliseconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-jitter)# timeout 10000`	IP SLA 動作がその要求パケットからの応答を待機する時間を設定します。 • *milliseconds* 引数を使用して、動作が応答を待機するミリ秒数を指定します。
ステップ 13	**tag** *text* 例： `RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-jitter)# tag ipsla`	（任意）IP SLA 動作のユーザ指定 ID を作成します。

ステップ 14	**exit**
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-icmp-path-jitter)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/CPU0:router(config)#`
		IP SLA 動作コンフィギュレーション モードおよび IP SLA コンフィギュレーション モードを終了します。グローバル コンフィギュレーション モードに戻ります。

ステップ 15	**ipsla schedule operation** *op-num*
	例：
	`RP/0/RSP0/CPU0:router(config)# ipsla schedule operation 432`
	`RP/0/RSP0/CPU0:router(config-ipsla-sched)#`
		動作の開始時間をスケジューリングします。基本スケジュールを設定できます。

	コマンドまたはアクション	目的
ステップ 16	**life** {**forever** | *seconds*} 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# life 30`	**forever** キーワードを指定すると、動作を無期限で実行するようにスケジューリングされます。*seconds* 引数を指定すると、動作のライフタイムが秒単位でスケジューリングされます。デフォルトの動作のライフタイムは 3600 秒（1 時間）です。
ステップ 17	**ageout** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# ageout 3600`	（任意）情報をアクティブに収集していない場合、動作をメモリに常駐させておく時間を秒数で指定します。デフォルト値の 0 秒は、動作がタイムアウトしないことを意味します。
ステップ 18	**recurring** 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# recurring`	（任意）動作が毎日指定された時刻に自動的に開始され、指定された期間実行されるように指定します。
ステップ 19	**start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*] 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00`	動作が開始される時刻を指定します。 ここでは、次のキーワードについて説明します。 • （任意）**pending** キーワードを使用して、動作を保留（未開始）状態にしておくように設定します。 デフォルト値は inactive です。 **start-time** コマンドが指定されていない場合、開始時刻が設定されるか、即時スタートを実行するトリガーが発生するまで、情報は収集されません。 • （任意）**now** キーワードを使用して、動作を即時スタートする必要があることを示します。 • （任意）**after** キーワードおよび関連する引数を使用して時刻を指定します。ここで指定した時刻以降に、動作が情報の収集を開始します。
ステップ 20	次のいずれかのコマンドを使用します。 • **end** • **commit**	設定変更を保存します。 • **end** コマンドを実行すると、変更をコミットするように要求されます。 `Uncommitted changes found, commit them`

	コマンドまたはアクション	目的

	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		`before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーションセッションが終了して、ルータがEXECモードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータがEXECモードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

ステップ 21	**show ipsla statistics** [*operation-number*] 例： `RP/0/RSP0/CPU0:router# show ipsla statistics 432`	現在の統計情報を表示します。

IP SLA MPLS LSP ping 動作およびトレース動作の設定

MPLS LSP ping 動作とトレース動作を使用すると、サービス プロバイダーは、ラベル スイッチド パス（LSP）をモニタし、MPLSフォワーディングの問題をすばやく切り分けることができます。送信元ルータとターゲット ルータの間のネットワーク接続の問題をトラブルシューティングするには、これらのIP SLA動作を使用します。 LSP をテストするため、MPLS LSP ping 動作とトレース動作は、エコー要求パケットを送信しエコー応答パケットを受信します。

MPLS LSP ping 動作またはトレース動作を設定およびスケジューリングするには、次のいずれかのタスクを実行します。

MPLS LSP ping 動作の設定およびスケジューリング

MPLS LSP ping 動作は、LSP の終端にエコー要求（ユーザ データグラム プロトコル（UDP）パケット）を送信し、診断データが格納されたエコー応答を受信することで、MPLS ネットワーク内の LSP パスに沿ったルータの接続性をテストします。

MPLS エコー要求パケットは、検証対象の LSP に関連付けられた適切なラベル スタックを使用してターゲット ルータに送信されます。 ラベル スタックを使用すると、パケットは LSP 自体を介して転送されます。

MPLS エコー要求パケットの宛先 IP アドレスは、ラベル スタックの選択に使用されるアドレスとは異なります。 宛先 IP アドレスは、127.x.y.z/8 アドレスとして定義されます。 127.x.y.z/8 アドレスを使用すると、LSP が切断された場合に IP パケットが宛先に IP スイッチングされるのを防ぐことができます。

MPLS エコー応答は、MPLS エコー要求に応じて送信されます。 応答は IP パケットとして送信され、IP、MPLS、または両方のスイッチング タイプの組み合わせを使用して転送されます。 MPLS エコー応答パケットの送信元アドレスは、エコー応答を生成するルータから取得されたアドレスです。 宛先アドレスは、MPLS エコー要求パケットを送信したルータの送信元アドレスです。 MPLS エコー応答の宛先ポートは、エコー要求の送信元ポートに設定されます。

MPLS LSP ping 動作では、サポートされているいずれかの Forwarding Equivalence Class（FEC; 転送等価クラス）エンティティを使用して、ping 送信元と各 FEC の出力ノード間の LSP の接続性が検証されます。 MPLS LSP ping 動作では、次の FEC タイプがサポートされています。

- LDP IPv4 プレフィックス（**target ipv4** コマンドで設定）
- MPLS TE トンネル（**target traffic-eng tunnel** コマンドで設定）
- 疑似回線（**target pseudowire** コマンドで設定）

手順の概要

1. **configure**
2. **ipsla operation** *operation-number*
3. **type mpls lsp ping**
4. **output interface** *type interface-path-id*
5. **target** {**ipv4** *destination-address destination-mask* | **traffic-eng tunnel** *tunnel-interface* | **pseudowire** *destination-address circuit-id*}
6. **lsp selector ipv4** *ip-address*
7. **force explicit-null**
8. **reply dscp** *dscp-bits*
9. **reply mode** {**control-channel** | **router-alert**}
10. **exp** *exp-bits*
11. **ttl** *time-to-live*
12. **exit**
13. **ipsla schedule operation** *operation-number*
14. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
15. 次のいずれかのコマンドを使用します。
    - **end**
    - **commit**
16. **show ipsla statistics** [*operation-number*]

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla operation 432`	IP SLA 動作を設定し、動作番号を指定します。範囲は 1 ～ 2048 です。
ステップ 3	**type mpls lsp ping** 例： `RP/0/RSP0/CPU0:router(config-ipsla-op)# type mpls lsp ping`	MPLS LSP ping 動作を設定し、IP SLA MPLS LSP ping コンフィギュレーション モードを開始します。

	コマンドまたはアクション	目的
ステップ 4	**output interface** *type interface-path-id* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-ping)# output interface pos 0/1/0/0`	（任意）LSP ping 動作で使用されるエコー要求出力インターフェイスを設定します。 （注） MPLS LSP ping 動作で使用されるターゲットとして疑似回線が指定されている場合は、**output interface** コマンドを使用できません。

ステップ 5	**target** {**ipv4** *destination-address destination-mask* | **traffic-eng tunnel** *tunnel-interface* | **pseudowire** *destination-address circuit-id*}
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-ping)# target ipv4 10.25.26.10 255.255.255.255`
	または
	`RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-ping)# target ipv4 10.25.26.10/32`
	または
	`RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-ping)# target traffic-eng tunnel 12`
	または
	`RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-trace)# target pseudowire 192.168.1.4 4211`
		MPLS LSP ping 動作のターゲット宛先を、LDP IPv4 アドレス、MPLS トラフィック エンジニアリング トンネル、または疑似回線として指定します。

ステップ 6	**lsp selector ipv4** *ip-address* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-ping)# lsp selector ipv4 127.0.0.2`	（任意）MPLS LSP ping 動作の LSP を選択するために使用されるローカル ホスト IPv4 アドレスを指定します。
ステップ 7	**force explicit-null** 例： `RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-ping)# force explicit-null`	（任意）エコー要求を送信するときに、LSP のラベル スタックに明示的な null ラベルを追加します。
ステップ 8	**reply dscp** *dscp-bits* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-ping)# reply dscp 2`	（任意）エコー応答パケットで使用する DiffServ コード ポイント（DSCP）値を指定します。有効な値は 0 ～ 63 です。 数値の代わりに、EF（緊急転送）や AF11（保証転送クラス AF11）などの予約されたキーワードを指定できます。

	コマンドまたはアクション	目的

ステップ 9	**reply mode** {**control-channel** | **router-alert**}
	
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-ping)# reply mode router-alert`
	または
	`RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-ping)# reply mode control-channel`
		（任意）MPLS LSP ping 動作の制御チャネルを経由してエコー応答パケットを送信するか、IP ルータ アラートを含む IPv4 UDP パケットとして応答するように、エコー要求を設定します。ルータ アラート応答モードでは、エコー応答パケットが宛先に戻る場合に、中間ホップごとに中継 LSR ルータによって特別な処理が実行されるように強制されます。
		
		（注） **control-channel** キーワードは、ターゲットが疑似回線に設定されている場合のみ使用できます。

ステップ 10	**exp** *exp-bits* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-ping)# exp 5`	（任意）エコー応答パケットのヘッダーで使用する MPLS 試験フィールド（EXP）値を指定します。有効な値の範囲は 0 ～ 7 です。
ステップ 11	**ttl** *time-to-live* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-ping)# ttl 200`	（任意）エコー要求パケットの MPLS ラベルで使用する存続可能時間（TTL）値を指定します。 有効な値は、1 ～ 255 です。

ステップ 12	**exit**
	
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-ping)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/CPU0:router(config)#`
		IP SLA MPLS LSP Ping コンフィギュレーション モードおよび IP SLA コンフィギュレーション モードを終了します。 グローバル コンフィギュレーション モードに戻ります。

ステップ 13	**ipsla schedule operation** *operation-number*
	
	例：
	`RP/0/RSP0/CPU0:router(config)# ipsla schedule operation 432`
	`RP/0/RSP0/CPU0:router(config-ipsla-sched)#`
		動作の開始時間をスケジューリングします。 基本スケジュールを設定できます。

ステップ 14	**start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*] 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00`	動作が開始される時刻を指定します。 ここでは、次のキーワードについて説明します。 • （任意）**pending** キーワードを使用して、動作を保留（未開始）状態にしておくように設定します。 デフォルト値は inactive です。 **start-time** コマンドが指定されていない場合、開始時刻が設定されるか、即時スタートを実

	コマンドまたはアクション	目的
		行するトリガーが発生するまで、情報は収集されません。 • （任意）**now** キーワードを使用して、動作を即時スタートする必要があることを示します。 • （任意）**after** キーワードおよび関連する引数を使用して時刻を指定します。ここで指定した時刻以降に、動作が情報の収集を開始します。

ステップ **15**	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

	コマンドまたはアクション	目的
ステップ 16	**show ipsla statistics** [*operation-number*] 例： `RP/0/RSP0/CPU0:router# show ipsla statistics 432`	現在の MPLS LSP ping 動作の IP SLA 統計情報を表示します。

MPLS LSP トレース動作の設定およびスケジューリング

MPLS LSP トレース動作は、エコー要求（UDP パケット）を各中継ラベル スイッチング ルータ（LSP）のコントロールプレーンに送信することにより、MPLS ネットワーク内のターゲット ルータへの LSP パスのホップバイホップ ルートをトレースします。 中継 LSR では、さまざまなチェックが実行され、LSP パスの中継 LSR であることが特定されます。 トレース動作では、ネットワーク接続のトラブルシューティングと、障害があるホップバイホップのローカライズを実行できます。

エコー要求パケットとエコー応答パケットが LSP を検証します。 MPLS LSP トレース動作の成功は、ラベル付きパケットを受信したときに MPLS エコー要求を処理する中継ルータに依存します。

中継ルータは、存続可能時間（TTL）が期限切れになった MPLS パケットまたは LSP の切断に対応して、中継ホップに関する情報を含む MPLS エコー応答を返します。 MPLS エコー応答の宛先ポートは、エコー要求の送信元ポートに設定されます。

MPLS LSP トレース動作では、各中継 LSR が、トレースされている Forwarding Equivalence Class（FEC;転送等価クラス）エンティティのタイプに関連する情報を返します。 この情報により、トレース動作では、ローカル フォワーディングの情報がルーティング プロトコルによって LSP パスとして特定された情報と一致するかどうかをチェックできます。

MPLS ラベルは、LSP で使用されている FEC のタイプに従って、パケットにバインドされます。 MPLS LSP トレース動作では、次の FEC タイプがサポートされています。

- LDP IPv4 プレフィックス（**target ipv4** コマンドで設定）
- MPLS TE トンネル（**target traffic-eng tunnel** コマンドで設定）

手順の概要

1. **configure**
2. **ipsla operation** *operation-number*
3. **type mpls lsp trace**
4. **output interface** *type interface-path-id*
5. 次のいずれかを実行します。
   - **target ipv4** *destination-address destination-mask*
   - **target traffic-eng tunnel** *tunnel-interface*
6. **lsp selector ipv4** *ip-address*
7. **force explicit-null**
8. **reply dscp** *dscp-bits*
9. **reply mode router-alert**
10. **exp** *exp-bits*
11. **ttl** *time-to-live*
12. **exit**
13. **ipsla schedule operation** *operation-number*
14. **start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*]
15. 次のいずれかのコマンドを使用します。
    - **end**
    - **commit**
16. **show ipsla statistics** [*operation-number*]

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla operation 432`	IP SLA 動作を設定し、動作番号を指定します。範囲は 1 ～ 2048 です。

	コマンドまたはアクション	目的
ステップ 3	**type mpls lsp trace** 例： `RP/0/RSP0/CPU0:router(config-ipsla-op)# type mpls lsp trace`	MPLS LSP トレース動作を設定し、IP SLA MPLS LSP トレース コンフィギュレーション モードを開始します。
ステップ 4	**output interface** *type interface-path-id* 例： `RP/0/RP0/CPU0:router(config-ipsla-mpls-lsp-ping)# output interface pos 0/1/0/0`	（任意）LSP トレース動作で使用されるエコー要求出力インターフェイスを設定します。

ステップ 5	次のいずれかを実行します。
	• **target ipv4** *destination-address destination-mask*
	• **target traffic-eng tunnel** *tunnel-interface*
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-trace)# target ipv4 10.25.26.10 255.255.255.255`
	`RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-trace)# target ipv4 10.25.26.10/32`
	または
	`RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-trace)# target traffic-eng tunnel 12`
		MPLS LSP トレース動作のターゲット宛先を、LDP IPv4 アドレスまたは MPLS トラフィック エンジニアリング トンネルとして指定します。

ステップ 6	**lsp selector ipv4** *ip-address* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-trace)# lsp selector ipv4 127.0.0.2`	（任意）IPv4 LSP ping 動作の LSP を選択するために使用されるローカル ホスト MPLS アドレスを指定します。
ステップ 7	**force explicit-null** 例： `RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-trace)# force explicit-null`	（任意）エコー要求を送信するときに、LSP のラベル スタックに明示的な null ラベルを追加します。
ステップ 8	**reply dscp** *dscp-bits* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-trace)# reply dscp 2`	（任意）エコー応答パケットで使用する DiffServ コード ポイント（DSCP）値を指定します。有効な値は 0 ～ 63 です。 数値の代わりに、EF（緊急転送）や AF11（保証転送クラス AF11）などの予約されたキーワードを指定できます。

	コマンドまたはアクション	目的
ステップ 9	**reply mode router-alert** 例： `RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-trace)# reply mode router-alert`	（任意）IP ルータ アラートを使用した IPv4 UDP パケットとして応答するようにエコー要求を設定します。 ルータ アラート応答モードでは、エコー応答パケットが宛先に戻る場合に、中間ホップごとに中継 LSR ルータによって特別な処理が実行されるように強制されます。
ステップ 10	**exp** *exp-bits* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-trace)# exp 5`	（任意）エコー応答パケットのヘッダーで使用する MPLS 試験フィールド（EXP）値を指定します。 有効な値の範囲は 0 ～ 7 です。
ステップ 11	**ttl** *time-to-live* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-trace)# ttl 20`	（任意）エコー要求パケットの MPLS ラベルで使用する存続可能時間（TTL）値を指定します。 有効な値は、1 ～ 255 です。

ステップ 12	**exit**
	
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-mpls-lsp-trace)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla-op)# exit`
	`RP/0/RSP0/CPU0:router(config-ipsla)# exit`
	`RP/0/RSP0/CPU0:router(config)#`
		IP SLA MPLS LSP トレース コンフィギュレーション モードおよび IP SLA コンフィギュレーション モードを終了します。 グローバル コンフィギュレーション モードに戻ります。

ステップ 13	**ipsla schedule operation** *operation-number*
	
	例：
	`RP/0/RSP0/CPU0:router(config)# ipsla schedule operation 432`
	`RP/0/RSP0/CPU0:router(config-ipsla-sched)#`
		動作の開始時間をスケジューリングします。 基本スケジュールを設定できます。

ステップ 14	**start-time** [*hh:mm:ss* {*day* | *month day*} | **now** | **pending** | **after** *hh:mm:ss*] 例： `RP/0/RSP0/CPU0:router(config-ipsla-sched)# start-time 01:00:00`	動作が開始される時刻を指定します。 ここでは、次のキーワードについて説明します。 • （任意）**pending** キーワードを使用して、動作を保留（未開始）状態にしておくように設定します。 デフォルト値は inactive です。 **start-time** コマンドが指定されていない場合、開始時刻が設定されるか、即時スタートを実行するトリガーが発生するまで、情報は収集されません。

	コマンドまたはアクション	目的
		• （任意）**now** キーワードを使用して、動作を即時スタートする必要があることを示します。 • （任意）**after** キーワードおよび関連する引数を使用して時刻を指定します。ここで指定した時刻以降に、動作が情報の収集を開始します。

ステップ 15	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーションファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーションファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。

ステップ 16	**show ipsla statistics** [*operation-number*] 例： `RP/0/RSP0/CPU0:router# show ipsla statistics 432`	トレース動作の現在の IP SLA 統計情報を表示します。

IP SLA 反応としきい値のモニタリングの設定

IP SLA でしきい値を設定して、しきい値違反を通知するようにするには、**ipsla reaction operation** コマンドと **ipsla reaction trigger** コマンドが必要です。 次の手順を実行して、IP SLA 反応としきい値のモニタリングを設定します。

IP SLA 反応のモニタ対象の要素の設定

IP SLA 反応は、モニタ対象の値が指定レベルを上回ったり下回ったりした場合や、モニタ対象のイベント（タイムアウトやに接続の切断など）が発生した場合にトリガーされるように設定されます。 これらのモニタ対象の値およびイベントは、モニタ対象の要素と呼ばれます。 特定の動作で反応が発生するように、反応の条件を設定できます。

利用できるモニタ対象の要素のタイプは、次の項に示されています。

接続の切断違反のトリガーの設定

モニタ対象の動作に接続の切断がある場合の反応を設定できます。

手順の概要

1. **configure**
2. **ipsla reaction operation** *operation-number*
3. **react** [**connection-loss**]
4. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla reaction operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla reaction operation 432`	IP SLA エージェントが制御するイベントに基づいた特定のアクションを設定します。 *operation-number* 引数は、設定されている反応に対する IP SLA 動作の数です。 範囲は 1 ～ 2048 です。

	コマンドまたはアクション	目的

ステップ 3	**react** [**connection-loss**]
	
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-react)# react connection-loss`
	`RP/0/RSP0/CPU0:router(config-ipsla-react-cond)#`
		反応をモニタする要素を指定します。
		
		connection-loss キーワードを使用して、モニタ対象の動作で接続の切断がある場合に反応が発生するように指定します。

ステップ 4	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

ジッター違反のトリガーの設定

ジッター値は送信元から宛先の値および宛先から送信元の値として計算されます。 各方向または両方向のジッター値が指定しきい値を上回るか下回る場合に、トラップなどのイベントをトリガーできます。 jitter-average をモニタ対象の要素として設定できます。

手順の概要

1. **configure**
2. **ipsla reaction operation** *operation-number*
3. **react** [**jitter-average** {**dest-to-source** | **source-to-dest**}]
4. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla reaction operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla reaction operation 432`	IP SLA エージェントが制御するイベントに基づいた特定のアクションを設定します。 *operation-number* 引数は、設定されている反応に対する IP SLA の動作数です。 範囲は 1 ～ 2048 です。

ステップ 3	**react** [**jitter-average** {**dest-to-source** | **source-to-dest**}]
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-react)# react jitter-average`
	`RP/0/RSP0/CPU0:router(config-ipsla-react-cond)#`
		反応をモニタする要素を指定します。
		反応は、平均ラウンドトリップ ジッター値が上限または下限のしきい値に違反している場合に発生します。 **jitter-average** キーワードには、次のオプションが用意されています。
		• **dest-to-source**：宛先から送信元（DS）のジッター平均を指定します。
		• **source-to-dest**：送信元から宛先（SD）のジッター平均を指定します。

ステップ 4	次のいずれかのコマンドを使用します。 • **end** • **commit** 例： `RP/0/RSP0/CPU0:router(config)# end`	設定変更を保存します。 • **end** コマンドを実行すると、変更をコミットするように要求されます。 `Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`

	コマンドまたはアクション	目的
	または `RP/0/RSP0/CPU0:router(config)# commit`	◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。 ◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。 ◦ cancel と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。 • 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

パケット損失違反のトリガーの設定

パケット損失値は送信元から宛先の値および宛先から送信元の値として計算されます。 各方向のパケット損失値が指定しきい値を上回るか下回る場合に、トラップなどのイベントをトリガーできます。 パケット損失をモニタ対象の要素として設定するには、このタスクを実行します。

手順の概要

1. **configure**
2. **ipsla reaction operation** *operation-number*
3. **react** [**packet-loss** [**dest-to-source** | **source-to-dest**]]
4. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla reaction operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla reaction operation 432`	IP SLA エージェントが制御するイベントに基づいた特定のアクションを設定します。 *operation-number* 引数は、設定されている反応に対する IP SLA 動作の数です。範囲は 1 ～ 2048 です。

ステップ 3	**react** [**packet-loss** [**dest-to-source** | **source-to-dest**]]
	
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-react)# react packet-loss dest-to-source`
	`RP/0/RSP0/CPU0:router(config-ipsla-react-cond)#`
		反応をモニタする要素を指定します。
		パケット損失値違反の反応が指定されます。 **packet-loss** キーワードには、次のオプションが用意されています。
		• **dest-to-source**：宛先から送信元（DS）のパケット損失違反を指定します。
		• **source-to-dest**：送信元から宛先（SD）のパケット損失違反を指定します。

ステップ 4	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。

	コマンドまたはアクション	目的
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

ラウンドトリップ違反のトリガーの設定

ラウンドトリップ時間（RTT）は、すべての IP SLA 動作のモニタ対象値です。 rtt 値が指定しきい値を上回るか下回る場合に、トラップなどのイベントをトリガーできます。 rtt をモニタ対象の要素として設定できます。

手順の概要

1. **configure**
2. **ipsla reaction operation** *operation-number*
3. **react** [**rtt**]
4. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla reaction operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla reaction operation 432`	IP SLA エージェントが制御するイベントに基づいた特定のアクションを設定します。 *operation-number* 引数は、設定されている反応に対する IP SLA 動作の数です。 範囲は 1 ～ 2048 です。
ステップ 3	**react** [**rtt**] 例： `RP/0/RSP0/CPU0:router(config-ipsla-react)# react rtt`	反応をモニタする要素を指定します。 **rtt** キーワードを使用して、ラウンドトリップ値が上限または下限のしきい値に違反する場合に発生する反応を指定します。

	コマンドまたはアクション	目的
	`RP/0/RSP0/CPU0:router(config-ipsla-react-cond)#`	

ステップ 4	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

タイムアウト違反のトリガーの設定

タイムアウト違反のトリガーを設定できます。

手順の概要

1. **configure**
2. **ipsla reaction operation** *operation-number*
3. **react** [**timeout**]
4. 次のいずれかのコマンドを使用します。
   • **end**
   • **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla reaction operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla reaction operation 432`	IP SLA エージェントが制御するイベントに基づいた特定のアクションを設定します。 *operation-number* 引数は、設定されている反応に対する IP SLA 動作の数です。 範囲は 1 ～ 2048 です。

ステップ 3	**react** [**timeout**]
	
	例：
	
	`RP/0/RSP0/CPU0:router(config-ipsla-react)# react timeout`
	`RP/0/RSP0/CPU0:router(config-ipsla-react-cond)#`
		反応をモニタする要素を指定します。
		
		timeout キーワードを使用して、モニタ対象の動作にタイムアウトがある場合に発生する反応を指定します。

ステップ 4	次のいずれかのコマンドを使用します。
	
	• **end**
	
	• **commit**
	
	例：
	
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

エラー検証違反のトリガーの設定

エラー検証違反がある場合の反応を指定できます。

手順の概要

1. **configure**
2. **ipsla reaction operation** *operation-number*
3. **react** [**verify-error**]
4. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla reaction operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla reaction operation 432`	IP SLA エージェントが制御するイベントに基づいた特定のアクションを設定します。 *operation-number* 引数は、設定されている反応に対する IP SLA 動作の数です。 範囲は 1 ～ 2048 です。

ステップ 3	**react** [**verify-error**]
	
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-react)# react verify-error`
	`RP/0/RSP0/CPU0:router(config-ipsla-react-cond)#`
		反応をモニタする要素を指定します。
		
		verify-error キーワードを使用して、エラー検証違反があるときに発生する反応を指定します。

ステップ 4	次のいずれかのコマンドを使用します。 • **end** • **commit**	設定変更を保存します。 • **end** コマンドを実行すると、変更をコミットするように要求されます。 `Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`

	コマンドまたはアクション	目的

	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

IP SLA 反応のしきい値違反タイプの設定

各モニタリング対象の要素では、次の項目を指定できます。

- しきい値をチェックするための条件
- 反応を発生させることができる条件の発生パターン（しきい値タイプなど）

たとえば、**threshold type immediate** コマンドを使用すると、対象の条件が確認されるとすぐに特定の要素で反応が発生するように指定できます。または、**threshold type consecutive** コマンドを使用すると、3 回連続して条件が確認されると反応が発生するように指定できます。

しきい値のタイプでは、イベントをトリガーするしきい値違反（またはしきい値違反の組み合わせ）のタイプを定義します。

この表では、しきい値違反タイプを一覧で示します。

表 *25* : *IP SLA* 反応のしきい値違反タイプ

しきい値違反のタイプ	説明
consecutive	違反が何回か連続して発生した後にのみイベントをトリガーします。たとえば、連続した違反タイプを使用すると、タイムアウトが 5 回連続して発生した後や、ラウンドトリップ時間が上限のしきい値を 5 回連続して上回った後にアクションが実行されるように設定できます。詳細については、連続した違反のイベントの生成,（322 ページ）を参照してください。
immediate	反応タイプ（応答時間など）の値が上限しきい値を上回るか、下限しきい値を下回る場合や、タイムアウト、接続の切断、verify-error イベントが発生した場合にイベントを即座にトリガーします。詳細については、各違反のイベントの生成,（320 ページ）を参照してください。
X / Y	y 回のプローブ動作以内に x 回の違反が発生すると（x回/y回）、イベントをトリガーします。詳細については、X / Y 違反のイベントの生成,（324 ページ）を参照してください。
averaged	プローブ動作の X 回の値の平均合計が、指定された上限しきい値を上回るか、下限しきい値を下回るときにイベントをトリガーします。詳細については、平均違反のイベントの生成,（326 ページ）を参照してください。

各違反のイベントの生成

指定された条件が満たされるたびに、トラップ生成したり、別の動作をトリガーしたりできます。

手順の概要

1. **configure**
2. **ipsla reaction operation** *operation-number*
3. **react** [**connection-loss** | **jitter-average** {**dest-to-source** | **source-to-dest**} | **packet-loss** [**dest-to-source** | **source-to-dest**] | **rtt** | **timeout** | **verify-error**]
4. **threshold type immediate**
5. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla reaction operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla reaction operation 432`	IP SLA エージェントが制御するイベントに基づいた特定のアクションを設定します。 *operation-number* 引数は、設定されている反応に対する IP SLA 動作の数です。 範囲は 1 ～ 2048 です。

ステップ 3	**react** [**connection-loss**	**jitter-average** {**dest-to-source**	**source-to-dest**}	**packet-loss** [**dest-to-source**	**source-to-dest**]	**rtt**	**timeout**	**verify-error**]
								例：
								`RP/0/RSP0/CPU0:router(config-ipsla-react)# react timeout`
								`RP/0/RSP0/CPU0:router(config-ipsla-react-cond)#`
									反応をモニタする要素を指定します。
									モニタ対象の動作にタイムアウトがあると、反応が指定されます。

ステップ 4	**threshold type immediate** 例： `RP/0/RSP0/CPU0:router(config-ipsla-react-cond)# threshold type immediate`	しきい値違反に対してただちにアクションを実行します。
ステップ 5	次のいずれかのコマンドを使用します。	設定変更を保存します。

	コマンドまたはアクション	目的

	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーションファイルに変更が保存され、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーションファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。

連続した違反のイベントの生成

連続した回数の違反が発生した後に、トラップ生成したり、別の動作をトリガーしたりできます。

手順の概要

1. **configure**
2. **ipsla reaction operation** *operation-number*
3. **react** [**connection-loss** | **jitter-average** {**dest-to-source** | **source-to-dest**} | **packet-loss** [**dest-to-source** | **source-to-dest**] | **rtt** | **timeout** | **verify-error**]
4. **threshold type consecutive** *occurrences*
5. 次のいずれかのコマンドを使用します。
   • **end**
   • **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla reaction operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla reaction operation 432`	IP SLA エージェントが制御するイベントに基づいた特定のアクションを設定します。 *operation-number* 引数は、設定されている反応に対する IP SLA 動作の数です。 範囲は 1 ～ 2048 です。

ステップ 3	**react** [**connection-loss** | **jitter-average** {**dest-to-source** | **source-to-dest**} | **packet-loss** [**dest-to-source** | **source-to-dest**] | **rtt** | **timeout** | **verify-error**]
	
	例：
	
	`RP/0/RSP0/CPU0:router(config-ipsla-react)# react connection-loss`
	`RP/0/RSP0/CPU0:router(config-ipsla-react-cond)#`
		反応をモニタする要素を指定します。
		
		モニタ対象の動作に接続の切断があると、反応が指定されます。

ステップ 4	**threshold type consecutive** *occurrences* 例： `RP/0/RSP0/CPU0:router(config-ipsla-react-cond)# threshold type consecutive 8`	連続した回数の違反が発生した後にアクションを実行します。 反応条件が連続した発生回数に対して設定されている場合、デフォルト値はありません。 発生回数は、しきい値タイプの指定時に設定されます。 連続した違反回数は 1 ～ 16 です。

ステップ 5	次のいずれかのコマンドを使用します。
	
	• **end**
	• **commit**
	
	例：
	
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィ

	コマンドまたはアクション	目的
		ギュレーションセッションは終了せず、設定変更もコミットされません。 • 設定変更を実行コンフィギュレーションファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。

X / Y 違反のイベントの生成

y 回のプローブ動作以内に x 回の違反が発生した後に（x 回/y 回）、トラップ生成したり、別の動作をトリガーしたりできます。 例として、**react** コマンドに **rtt** rtt キーワードを指定して使用します。

手順の概要

1. **configure**
2. **ipsla reaction operation** *operation-number*
3. **react** [**connection-loss** | **jitter-average** {**dest-to-source** | **source-to-dest**} | **packet-loss** [**dest-to-source** | **source-to-dest**] | **rtt** | **timeout** | **verify-error**]
4. **threshold type xofy** *X value Y value*
5. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla reaction operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla reaction operation 432`	IP SLA エージェントが制御するイベントに基づいた特定のアクションを設定します。 *operation-number* 引数は、設定されている反応に対する IP SLA 動作の数です。 範囲は 1 ～ 2048 です。

	コマンドまたはアクション	目的

ステップ 3	**react** [**connection-loss** | **jitter-average** {**dest-to-source** | **source-to-dest**} | **packet-loss** [**dest-to-source** | **source-to-dest**] | **rtt** | **timeout** | **verify-error**]
	
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-react)# react rtt`
	`RP/0/RSP0/CPU0:router(config-ipsla-react-cond)#`
		ラウンドトリップ値が上限しきい値または下限しきい値に違反している場合に反応が発生するように指定します。

ステップ 4	**threshold type xofy** *X value Y value* 例： `RP/0/RSP0/CPU0:router(config-ipsla-react-cond)# threshold type xofy 7 7`	モニタ対象の要素でしきい値違反などの反応条件が発生した場合、*y* 回のプローブ動作以内に *x* 回の違反が発生すると（*x* 回/*y*回）、**action** コマンドでの定義に従って、アクションが実行されます。デフォルトは、*x-value* および *y-value* の両方とも 5 です（**xofy** 5 5）。 各値の有効範囲は 1 ～ 16 です。

ステップ 5	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

平均違反のイベントの生成

プローブ動作の X の平均合計数が下限しきい値または上限しきい値に違反する場合、トラップ生成したり、別の動作をトリガーしたりできます。

手順の概要

1. **configure**
2. **ipsla reaction operation** *operation-number*
3. **react** [**connection-loss** | **jitter-average** {**dest-to-source** | **source-to-dest**} | **packet-loss** [**dest-to-source** | **source-to-dest**] | **rtt** | **timeout** | **verify-error**]
4. **threshold type average** *number-of-probes*
5. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla reaction operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla reaction operation 432`	IP SLA エージェントが制御するイベントに基づいた特定のアクションを設定します。 *operation-number* 引数は、設定されている反応に対する IP SLA 動作の数です。 範囲は 1 ～ 2048 です。

ステップ 3	**react** [**connection-loss**	**jitter-average** {**dest-to-source**	**source-to-dest**}	**packet-loss** [**dest-to-source**	**source-to-dest**]	**rtt**	**timeout**	**verify-error**]
								例：
								`RP/0/RSP0/CPU0:router(config-ipsla-react)# react packet-loss dest-to-source`
								`RP/0/RSP0/CPU0:router(config-ipsla-react-cond)#`
									反応をモニタする要素を指定します。
									パケット損失値違反の反応が指定されます。 **packet-loss** キーワードには、次のオプションが用意されています。
									• **dest-to-source**：宛先から送信元（DS）のパケット損失違反を指定します。
									• **source-to-dest**：送信元から宛先（SD）のパケット損失違反を指定します。

	コマンドまたはアクション	目的
ステップ 4	**threshold type average** *number-of-probes* 例： `RP/0/RSP0/CPU0:router(config-ipsla-react-cond)# threshold type average 8`	平均値がしきい値に違反した場合にアクションを実行します。

ステップ 5	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。

反応イベントの指定

反応条件が検出されるとき、**action** コマンドを使用して、発生するアクションのタイプを設定できます。 次のアクション タイプが設定されます。

- **logging**：**logging** キーワードが設定されると、反応が発生したことを示すメッセージがコンソールに生成されます。
- **trigger**：**trigger** キーワードが設定されると、1 回以上の他の動作を開始できます。 その結果、**ipsla reaction trigger** *op1 op2* コマンドで開始できる動作を制御できます。 このコマンド

は、*op1* がアクション タイプのトリガーを生成すると、動作 *op2* を開始できることを示します。

反応イベントを指定できます。 例として、**react** コマンドに **connection-loss** キーワードを指定して使用します。

手順の概要

1. **configure**
2. **ipsla reaction operation** *operation-number*
3. **react** [**connection-loss** | **jitter-average** {**dest-to-source** | **source-to-dest**} | **packet-loss** [**dest-to-source** | **source-to-dest**] | **rtt** | **timeout** | **verify-error**]
4. **action** [**logging** | **trigger**]
5. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla reaction operation** *operation-number* 例： `RP/0/RSP0/CPU0:router(config)# ipsla reaction operation 432`	IP SLA エージェントが制御するイベントに基づいた特定のアクションを設定します。 *operation-number* 引数は、設定されている反応に対する IP SLA 動作の数です。 範囲は 1 ～ 2048 です。

ステップ 3	**react** [**connection-loss**	**jitter-average** {**dest-to-source**	**source-to-dest**}	**packet-loss** [**dest-to-source**	**source-to-dest**]	**rtt**	**timeout**	**verify-error**]
								例：
								`RP/0/RSP0/CPU0:router(config-ipsla-react)# react connection-loss`
								`RP/0/RSP0/CPU0:router(config-ipsla-react-cond)#`
									モニタ対象の動作で接続の切断がある場合の反応を指定します。

	コマンドまたはアクション	目的
ステップ 4	**action** [**logging** | **trigger**] 例： `RP/0/RSP0/CPU0:router(config-ipsla-react-cond)# action logging`	**react** コマンドを設定した場合、またはしきい値イベントが発生した場合に実行されるアクションまたはアクションの組み合わせを指定します。次のアクションタイプが記述されます。 • **logging**：モニタ対象の要素で指定された違反タイプが発生した場合に、ロギングメッセージを送信します。IP SLA エージェントは syslog を生成し、SNMP に通知します。トラップを生成するかどうかは、SNMP エージェントによって決定されます。 • **trigger**：違反条件に一致した場合に保留からアクティブへの移行が発生する 1 つまたは複数の動作の動作ステートを決定します。トリガーされるターゲット動作は、**ipsla reaction trigger** コマンドを使用して指定します。ターゲット動作は、そのターゲット動作の lifetime 値で指定された存続期間が経過するまで継続します。トリガーされたターゲット動作は、存続期間が終了するまで、再度トリガーされることはありません。

ステップ 5	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。

	コマンドまたはアクション	目的
		• 設定変更を実行コンフィギュレーションファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。

送信元 PE ルータでの MPLS LSP モニタリング インスタンスの設定

このタスクを実行して、MPLS LSP モニタ（MPLSLM）インスタンスの動作パラメータを設定します。 IP SLA 測定統計情報は送信元 PE ルータに保存されます。

MPLS LSP モニタ ping またはトレース インスタンスを設定するには、次のタスクのいずれかを実行します。

MPLS LSP モニタリング ping インスタンスの設定

はじめる前に

（注） MPLS LSP モニタリングは PE ルータで設定されます。

手順の概要

1. **configure**
2. **ipsla**
3. **mpls discovery vpn**
4. **interval** *minutes*
5. **exit**
6. **mpls lsp-monitor**
7. **monitor** *monitor-id*
8. **type mpls lsp ping**
9. **vrf** *vrf-name*
10. **scan interval** *scan-interval*
11. **scan delete-factor** *factor-value*
12. **timeout** *milliseconds*
13. **datasize request** *size*
14. **lsp selector ipv4** *ip-address*
15. **force explicit-null**
16. **reply dscp** *dscp-bits*
17. **reply mode router-alert**
18. **ttl** *time-to-live*
19. **tag** *text*
20. **exp** *exp-bits*
21. **statistics hourly** [**buckets** *hours*]
22. 次のいずれかのコマンドを使用します。
    - **end**
    - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla** 例： `RP/0/RSP0/CPU0:router(config)# ipsla`	IP SLA コンフィギュレーション モードを開始し、IP サービス レベル契約を設定します。

	コマンドまたはアクション	目的
ステップ 3	**mpls discovery vpn** 例： `RP/0/RSP0/CPU0:router(config-ipsla)# mpls discovery vpn`	（任意）MPLS VPN BGP ネクスト ホップ ネイバー探索コンフィギュレーション モードを開始します。
ステップ 4	**interval** *minutes* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mpls-discovery-vpn)# interval 120`	（任意）有効ではなくなったルーティング エントリが MPLS VPN の BGP ネクスト ホップ ネイバー探索データベースから削除される間隔を指定します。 デフォルトの間隔は 60 分です。
ステップ 5	**exit** 例： `RP/0/RSP0/CPU0:router(config-ipsla-mpls-discovery-vpn)# exit`	MPLS ディスカバリ VPN コンフィギュレーション モードを終了します。

ステップ 6	**mpls lsp-monitor**
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla)# mpls lsp-monitor`
	`RP/0/RSP0/CPU0:router(config-ipsla-mplslm)#`
		MPLS LSP モニタ モードを開始します。 このモードから、LSP モニタ インスタンスの設定、LSP モニタ インスタンスの反応の設定、または LSP モニタ インスタンスのスケジューリングを実行できます。

ステップ 7	**monitor** *monitor-id*
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-mplslm)# monitor 1`
	`RP/0/RSP0/CPU0:router(config-ipsla-mplslm-def)#`
		MPLS LSP モニタ インスタンスを設定し、IP SLA MPLS LSP モニタ コンフィギュレーション モードを開始します。

ステップ 8	**type mpls lsp ping** 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-def)# type mpls lsp ping`	検出されたそれぞれの BGP ネクスト ホップ アドレスに対して、自動的に MPLS LSP ping 動作を作成し、対応するコンフィギュレーション モードを開始して、パラメータを設定します。
ステップ 9	**vrf** *vrf-name* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-ping)# vrf SANJOSE`	（任意）ping 動作で特定のバーチャル プライベート ネットワーク（VPN）ルーティングおよび転送（VRF）インスタンスのモニタリングをイネーブルにします。 VRF を指定しない場合、MPLS LSP モニタリング インスタンスはすべての VRF をモニタします。

	コマンドまたはアクション	目的
ステップ 10	**scan interval** *scan-interval* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-ping)# scan interval 300`	（任意）MPLS LSP モニタ インスタンスが BGP ネクスト ホップ ネイバーの更新のためにスキャン キューをチェックする間隔（分単位）を指定します。 デフォルトの間隔は 240 分です。 各間隔では、MPLS LSP モニタ インスタンス スキャン キューにリストされている新しく検出された BGP ネクスト ホップ ネイバーごとに、新しい IP SLA 動作が自動的に作成されます。
ステップ 11	**scan delete-factor** *factor-value* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-ping)# scan delete-factor 2`	（任意）有効ではなくなった BGP ネクスト ホップ ネイバーに対する IP SLA 動作を自動的に削除するまでに、MPLS LSP モニタ インスタンスがスキャン キューをチェックする回数を指定します。 デフォルトのスキャン ファクタは 1 です。 つまり、MPLS LSP モニタ インスタンスがスキャン キューで更新をチェックするたびに、有効ではなくなった BGP ネクスト ホップ ネイバーの IP SLA 動作が削除されます。 スキャン ファクタが 0 に設定されると、MPLS LSP モニタ インスタンスによって IP SLA 動作は削除されません。 この設定は推奨しません。
ステップ 12	**timeout** *milliseconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-ping)# timeout 50000`	（任意）各 MPLS LSP 動作が LSP 検証（LSPV）サーバからの応答を待機する時間の長さを指定します。 デフォルト値は 5000 ミリ秒です。
ステップ 13	**datasize request** *size* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-ping)# datasize request 512`	（任意）MPLS LSP エコー要求パケットのペイロード サイズを指定します。 デフォルト値は 100 バイトです。 （注） このコマンドは、MPLS LSP ping モードだけで利用できます。
ステップ 14	**lsp selector ipv4** *ip-address* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-ping)# lsp selector ipv4 127.10.10.1`	（任意）複数の LSP からラベル スイッチド パス（LSP）を選択するために使用するローカル ホスト IP アドレス（127.*x.x.x*）を指定します。 デフォルト値は 127.0.0.1 です。

	コマンドまたはアクション	目的
ステップ 15	**force explicit-null** 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-ping)# force explicit-null`	（任意）MPLS LSP エコー要求パケットのラベル スタックに、明示的な Null ラベルが追加されるかどうかを指定します。 これは、デフォルトではディセーブルになっています。
ステップ 16	**reply dscp** *dscp-bits* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-ping)# reply dscp 5`	（任意）MPLS LSP エコー応答パケットの IP ヘッダーで使用される DiffServ サービス コードポイント（DSCP）値を指定します。
ステップ 17	**reply mode router-alert** 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-ping)# reply mode router-alert`	（任意）MPLS LSP エコー応答パケットでルータ アラート オプションの使用をイネーブルにします。 これは、デフォルトではディセーブルになっています。
ステップ 18	**ttl** *time-to-live* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-ping)# ttl 200`	（任意）MPLS LSP 動作に使用されるエコー要求パケットの最大ホップ カウントを指定します。 デフォルト値は 255 です。
ステップ 19	**tag** *text* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-ping)# tag mplslm-tag`	（任意）MPLS LSP 動作のユーザ指定 ID を作成します。
ステップ 20	**exp** *exp-bits* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-ping)# exp 7`	（任意）エコー要求パケットの MPLS ヘッダーで使用される試験フィールド値を指定します。 デフォルト値は 0 です
ステップ 21	**statistics hourly** [**buckets** *hours*] 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-ping)# statistics hourly buckets 2`	（任意）MPLS LSP モニタリング インスタンスでの動作の統計情報収集パラメータを指定します。 時間のデフォルト値は 2 です。
ステップ 22	次のいずれかのコマンドを使用します。 • **end** • **commit**	設定変更を保存します。 • **end** コマンドを実行すると、変更をコミットするように要求されます。 `Uncommitted changes found, commit them`

	コマンドまたはアクション	目的

	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		`before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーションファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーションファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

次の作業

- 反応条件を設定します。
- MPLS LSP モニタリング インスタンス動作のスケジュールを設定します。

MPLS LSP モニタリング トレース インスタンスの設定

はじめる前に

（注） MPLS LSP モニタリングは PE ルータで設定されます。

手順の概要

1. **configure**
2. **ipsla**
3. **mpls discovery vpn**
4. **interval** *minutes*
5. **exit**
6. **mpls lsp-monitor**
7. **monitor** *monitor-id*
8. **type mpls lsp trace**
9. **vrf** *vrf-name*
10. **scan interval** *scan-interval*
11. **scan delete-factor** *factor-value*
12. **timeout** *milliseconds*
13. **lsp selector ipv4** *ip-address*
14. **force explicit-null**
15. **reply dscp** *dscp-bits*
16. **reply mode router-alert**
17. **ttl** *time-to-live*
18. **tag** *text*
19. **exp** *exp-bits*
20. **statistics hourly** [**buckets** *hours*]
21. 次のいずれかのコマンドを使用します。
    - **end**
    - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla** 例： `RP/0/RSP0/CPU0:router(config)# ipsla`	IP SLA コンフィギュレーション モードを開始し、IP サービス レベル契約を設定します。

	コマンドまたはアクション	目的
ステップ 3	**mpls discovery vpn** 例： `v:router(config-ipsla)# mpls discovery vpn`	（任意）MPLS VPN BGP ネクスト ホップ ネイバー探索をイネーブルにします。
ステップ 4	**interval** *minutes* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mpls-discovery-vpn)# interval 120`	（任意）有効ではなくなったルーティング エントリが MPLS VPN の BGP ネクスト ホップ ネイバー探索データベースから削除される間隔を指定します。 デフォルトの間隔は 60 分です。
ステップ 5	**exit** 例： `RP/0/RSP0/CPU0:router(config-ipsla-mpls-discovery-vpn)# exit`	MPLS ディスカバリ VPN コンフィギュレーション モードを終了します。

ステップ 6	**mpls lsp-monitor**
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla)# mpls lsp-monitor`
	`RP/0/RSP0/CPU0:router(config-ipsla-mplslm)#`
		MPLS LSP モニタ モードを開始します。 このモードから、LSP モニタ インスタンスの設定、LSP モニタ インスタンスの反応の設定、または LSP モニタ インスタンスのスケジューリングを実行できます。

ステップ 7	**monitor** *monitor-id*
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-mplslm)# monitor 1`
	`RP/0/RSP0/CPU0:router(config-ipsla-mplslm-def)#`
		MPLS LSP モニタ インスタンスを設定し、IP SLA MPLS LSP モニタ コンフィギュレーション モードを開始します。

ステップ 8	**type mpls lsp trace** 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplsm-def)# type mpls lsp trace`	検出されたそれぞれの BGP ネクスト ホップ アドレスに対して自動的に MPLS LSP トレース動作を作成し、対応するコンフィギュレーション モードを開始して、パラメータを設定します。
ステップ 9	**vrf** *vrf-name* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-trace)# vrf SANJOSE`	（任意）traceroute 動作で特定のバーチャル プライベート ネットワーク（VPN）ルーティングおよび転送（VRF）インスタンスのモニタリングをイネーブルにします。 VRF を指定しない場合、MPLS LSP モニタリング インスタンスはすべての VRF をモニタします。

	コマンドまたはアクション	目的
ステップ 10	**scan interval** *scan-interval* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-trace)# scan interval 300`	（任意）MPLS LSP モニタ インスタンスが BGP ネクスト ホップ ネイバーの更新のためにスキャン キューをチェックする間隔（分単位）を指定します。 デフォルトの間隔は 240 分です。 各間隔では、MPLS LSP モニタ インスタンス スキャン キューにリストされている新しく検出された BGP ネクスト ホップ ネイバーごとに、新しい IP SLA 動作が自動的に作成されます。
ステップ 11	**scan delete-factor** *factor-value* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-trace)# scan delete-factor 2`	（任意）有効ではなくなった BGP ネクスト ホップ ネイバーに対する IP SLA 動作を自動的に削除するまでに、MPLS LSP モニタ インスタンスがスキャン キューをチェックする回数を指定します。 デフォルトのスキャン ファクタは 1 です。 つまり、MPLS LSP モニタ インスタンスがスキャン キューで更新をチェックするたびに、有効ではなくなった BGP ネクスト ホップ ネイバーの IP SLA 動作が削除されます。 スキャン ファクタが 0 に設定されると、MPLS LSP モニタ インスタンスによって IP SLA 動作は削除されません。 この設定は推奨しません。
ステップ 12	**timeout** *milliseconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-trace)# timeout 50000`	（任意）各 MPLS LSP 動作が LSP 検証（LSPV）サーバからの応答を待機する時間の長さを指定します。 デフォルト値は 5000 ミリ秒です。
ステップ 13	**lsp selector ipv4** *ip-address* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-trace)# lsp selector ipv4 127.10.10.1`	（任意）複数の LSP からラベル スイッチドパス（LSP）を選択するために使用するローカル ホスト IP アドレス（127.*x.x.x*）を指定します。 デフォルト値は 127.0.0.1 です。

	コマンドまたはアクション	目的
ステップ 14	**force explicit-null** 例： `RP/0/RSP0/CPU0RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-trace)# force explicit-null`	（任意）MPLS LSP エコー要求パケットのラベル スタックに、明示的な Null ラベルが追加されるかどうかを指定します。これは、デフォルトではディセーブルになっています。
ステップ 15	**reply dscp** *dscp-bits* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-trace)# reply dscp 5`	（任意）MPLS LSP エコー応答パケットの IP ヘッダーで使用される DiffServ サービス コードポイント（DSCP）値を指定します。
ステップ 16	**reply mode router-alert** 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-trace)# reply mode router-alert`	（任意）MPLS LSP エコー応答パケットでルータ アラート オプションの使用をイネーブルにします。これは、デフォルトではディセーブルになっています。
ステップ 17	**ttl** *time-to-live* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-trace)# ttl 40`	（任意）MPLS LSP 動作に使用されるエコー要求パケットの最大ホップ カウントを指定します。デフォルト値は 30 です。
ステップ 18	**tag** *text* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-trace)# tag mplslm-tag`	（任意）MPLS LSP 動作のユーザ指定 ID を作成します。
ステップ 19	**exp** *exp-bits* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-trace)# exp 7`	（任意）エコー要求パケットの MPLS ヘッダーで使用される試験フィールド値を指定します。デフォルト値は 0 です
ステップ 20	**statistics hourly** [**buckets** *hours*] 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-trace)# statistics hourly buckets 2`	（任意）MPLS LSP モニタリング インスタンスでの動作の統計情報収集パラメータを指定します。時間のデフォルト値は 2 です。
ステップ 21	次のいずれかのコマンドを使用します。 • **end** • **commit**	設定変更を保存します。 • **end** コマンドを実行すると、変更をコミットするように要求されます。 `Uncommitted changes found,`

	コマンドまたはアクション	目的

	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		`commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーションファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

次の作業

- 反応条件を設定します。
- MPLS LSP モニタリング インスタンス動作のスケジュールを設定します。

送信元 PE ルータでの MPLS LSP モニタリング インスタンスの反応条件の設定

このタスクを実行して、MPLS LSP モニタリング インスタンスの反応条件を設定します。

はじめる前に

MPLS LSP モニタリング インスタンスは、反応条件を設定する前に定義してください。

手順の概要

1. **configure**
2. **ipsla**
3. **mpls lsp-monitor**
4. **reaction monitor** *monitor-id*
5. **react** {**connection-loss** | **timeout**}
6. **action logging**
7. **threshold type** {**consecutive** *occurrences* | **immediate**}
8. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla** 例： `RP/0/RSP0/CPU0:router(config)# ipsla`	IP SLA コンフィギュレーション モードを開始し、IP サービス レベル契約を設定します。

ステップ 3	**mpls lsp-monitor**
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla)# mpls lsp-monitor`
	`RP/0/RSP0/CPU0:router(config-ipsla-mplslm)#`
		MPLS LSP モニタ モードを開始します。 このモードから、LSP モニタ インスタンスの設定、LSP モニタ インスタンスの反応の設定、または LSP モニタ インスタンスのスケジューリングを実行できます。

	コマンドまたはアクション	目的

ステップ 4	**reaction monitor** *monitor-id*
	
	例：
	
	`RP/0/RSP0/CPU0:router(config-ipsla-mplslm)# reaction monitor 2`
	`RP/0/RSP0/CPU0:router(config-ipsla-mplslm-react)#`
		MPLS LSP モニタ インスタンスの反応を設定し、IP SLA MPLS LSP モニタ反応のコンフィギュレーション モードを開始します。

ステップ 5	**react** {**connection-loss** | **timeout**} 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-react)# react connection-loss`	一方向の接続が切断されたり、モニタ対象の動作にタイムアウトが発生したりすると、反応が発生するように指定します。 自動的に作成された動作に条件が当てはまると、反応は適用されます。
ステップ 6	**action logging** 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-react-cond)# action logging`	反応条件およびしきい値の結果として、イベントがログに記録されるように指定します。
ステップ 7	**threshold type** {**consecutive** *occurrences* | **immediate**} 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-react-cond)# threshold type consecutive`	指定した回数の違反が連続して発生した場合や、違反が発生すると即座に指定されたアクションが実行されるように指定します。 *occurrences* の有効な値の範囲は 1 ～ 16 です。

ステップ 8	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーションセッションが終了して、ルータがEXECモードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータがEXECモードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッ

	コマンドまたはアクション	目的
		ションは終了せず、設定変更もコミットされません。 • 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

次の作業

- MPLS LSP モニタリング インスタンス動作のスケジュールを設定します。

送信元PEルータでのMPLSLSPモニタリングインスタンスのスケジュール設定

このタスクを実行して、MPLS LSP モニタリング インスタンスでの動作のスケジュールを設定します。

手順の概要

1. **configure**
2. **ipsla**
3. **mpls lsp-monitor**
4. **schedule monitor** *monitor-id*
5. **frequency** *seconds*
6. **schedule period** *seconds*
7. **start-time** *hh*:*mm*:*ss* [*day* | *month day*]
8. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla** 例： `RP/0/RSP0/CPU0:router(config)# ipsla`	IP SLA コンフィギュレーション モードを開始し、IP サービス レベル契約を設定します。

ステップ 3	**mpls lsp-monitor**
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla)# mpls lsp-monitor`
	`RP/0/RSP0/CPU0:router(config-ipsla-mplslm)#`
		MPLS LSP モニタ モードを開始します。 このモードから、LSP モニタ インスタンスの設定、LSP モニタ インスタンスの反応の設定、または LSP モニタ インスタンスのスケジューリングを実行できます。

ステップ 4	**schedule monitor** *monitor-id*
	例：
	`RP/0/RSP0/CPU0:router(config-ipsla-mplslm)# schedule monitor 2`
	`RP/0/RSP0/CPU0:router(config-ipsla-mplslm-sched)#`
		IP SLA MPLS LSP モニタ スケジュール コンフィギュレーション モードを開始して、MPLS LSP モニタ インスタンスのスケジュールを設定します。

ステップ 5	**frequency** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-sched)# frequency 600`	（任意）スケジュール期間が実行される頻度を指定します。デフォルト値はスケジュール期間と同じです。スケジュール期間は **schedule period** コマンドを使用して指定されます。 MPLS LSP MPLS LSP モニタ インスタンスの開始時刻のスケジュールを設定する前に、この値を指定する必要があります。
ステップ 6	**schedule period** *seconds* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-sched)# schedule period 300`	その期間ですべての動作が実行されるようにスケジュールを設定する時間を秒単位で指定します。 すべての動作のスケジュールは、スケジュール期間中を通して均等の間隔で設定されます。 動作のセット全体が実行される頻度を指定するには、**frequency** コマンドを指定します。 頻度の値は、スケジュール期間以上である必要があります。 MPLS LSP MPLS LSP モニタ インスタンスの開始時刻のスケジュールを設定する前に、この値を指定する必要があります。

	コマンドまたはアクション	目的
ステップ 7	**start-time** *hh*:*mm*:*ss* [*day* | *month day*] 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-sched)# start-time 11:45:00 July 4`	MPLS LSP モニタ インスタンスが情報の収集を開始するときを指定します。 スケジュールを設定した時刻を指定する必要があります。指定しない場合、情報が収集されません。

ステップ 8	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

LSP パス ディスカバリ

このタスクを実行して、LSP パス ディスカバリ（LPD）およびエコー間隔、パス、スキャンなどの必要なパラメータを設定します。

手順の概要

1. **configure**
2. **ipsla**
3. **mpls lsp-monitor**
4. **monitor** *monitor-id*
5. **type mpls lsp ping**
6. **path discover**
7. **echo interval** *time*
8. **echo maximum lsp selector ipv4** *host address*
9. **echo multipath bitmap-size** *size*
10. **echo retry** *count*
11. **echo timeout** *value*
12. **path retry** *range*
13. **path secondary frequency** {**both** | **connection-loss** | **timeout**} *value*}
14. **scan period** *value*
15. 次のいずれかのコマンドを使用します。
    - **end**
    - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**ipsla** 例： `RP/0/RSP0/CPU0:router(config)# ipsla`	IP SLA コンフィギュレーション モードを開始し、IP サービス レベル契約を設定します。
ステップ 3	**mpls lsp-monitor** 例： `RP/0/RSP0/CPU0:router(config-ipsla)# mpls lsp-monitor`	MPLS LSP モニタ モードを開始します。 このモードから、LSP モニタ インスタンスの設定、LSP モニタ インスタンスの反応の設定、または LSP モニタ インスタンスのスケジューリングを実行できます。

	コマンドまたはアクション	目的
ステップ 4	**monitor** *monitor-id* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm)# monitor 2`	MPLS LSP モニタ インスタンスを設定します。
ステップ 5	**type mpls lsp ping** 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-def)# type mpls lsp ping`	ラベル スイッチド パス（LSP）のエンドツーエンドの接続と MPLS ネットワークの整合性を検証します。
ステップ 6	**path discover** 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-ping)# path discover`	LSP パス ディスカバリをイネーブルにします。
ステップ 7	**echo interval** *time* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-lpd)# echo interval 777`	パス ディスカバリ中に送信される MPLS LSP エコー要求のインターバル（ミリ秒単位）を設定します。 範囲は 0 ～ 3600000 です。 デフォルトは 0 です。
ステップ 8	**echo maximum lsp selector ipv4** *host address* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-lpd)# echo maximum lsp selector ipv4 host_one 127.100.100.100`	パス ディスカバリ中に使用される最大セレクタ値であるローカル ホスト IP アドレス（127.x.x.x）を設定します。 デフォルトは 127.255.255.255 です。
ステップ 9	**echo multipath bitmap-size** *size* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-lpd)# echo multipath bitmap-size 50`	パス ディスカバリ中に MPLS LSP エコー要求のダウンストリーム マッピングで送信されるセレクタの最大数を設定します。 範囲は 1 ～ 256 です。 デフォルトは 32 です。
ステップ 10	**echo retry** *count* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-lpd)# echo retry 3`	パス ディスカバリ中に送信される MPLS LSP エコー要求のタイムアウト リトライ回数を設定します。 範囲は 0 ～ 10 です。 デフォルトは 3 です。
ステップ 11	**echo timeout** *value* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-lpd)# echo timeout 300`	パス ディスカバリ中のエコー要求のタイムアウト値を設定します。 範囲は 0 ～ 3600（ミリ秒単位）です。 デフォルトは 5 です。

	コマンドまたはアクション	目的
ステップ 12	**path retry** *range* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-lpd)# path retry 12`	MPLS LSP パスの再試行範囲を設定します。 範囲は 1 ～ 16 です。 デフォルトは 1 です。
ステップ 13	**path secondary frequency** {**both** | **connection-loss** | **timeout**} *value*} 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-lpd)# path secondary frequency both 600`	次の secondary frequency をイネーブルにします。 • タイムアウトおよび接続の切断の両方 • 接続の切断のみ • タイムアウトのみ （注） デフォルト値はありません。
ステップ 14	**scan period** *value* 例： `RP/0/RSP0/CPU0:router(config-ipsla-mplslm-lsp-lpd)# scan period 60`	MPLS LSP スキャン期間の値を設定します。 範囲は 0 ～ 7200 分です。 デフォルトは 5 です。

ステップ 15	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。

	コマンドまたはアクション	目的
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

トラッキング タイプの設定（rtr）

IPSLA は、ipsla オブジェクト タイプです。 オブジェクト トラッカーは、ipsla 操作の戻りコードをトラッキングして、トラック状態の変化を判断します。

手順の概要

1. configure
2. **track** *track-name*
3. **type rtr** *ipsla operation id* **reachability**
4. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	configure 例： `RP/0/RSP0/CPU0:router# 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 2	**track** *track-name* 例： `RP/0/RSP0/CPU0:router(config)# track t1`	トラック コンフィギュレーション モードを開始します。
ステップ 3	**type rtr** *ipsla operation id* **reachability** 例： `RP/0/RSP0/CPU0:routertype rtr 100 reachability`	到達可能性のためにトラッキングする必要がある ipsla 操作 id を設定します。

	コマンドまたはアクション	目的

ステップ 4	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーションファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。

IP サービス レベル契約を実装するための設定例

IP サービス レベル契約の設定：例

次の例では、UDP エコー動作とスケジュールを設定する方法を示します。

```
configure
ipsla operation 432
  type udp echo
    destination address 12.25.26.10
    destination port 11111
    frequency 300
 exit
exit
ipsla schedule operation 432
  life 30
  ageout 3600
  recurring
  start-time after 01:00:00
end

show ipsla statistics 432
```

IP SLA 反応としきい値のモニタリングの設定：例

次の例では、IP SLA 反応およびしきい値モニタリングを設定する方法を示します。 次の作業を実行できます。

- true または false の条件をアクティブ化する属性の反応を設定します。たとえば、1、5、6 です。
- しきい値を受け入れる属性の反応を設定します。
- 追加の threshold type オプションを設定します。
- アクション タイプのロギングまたはトリガーを設定します。

```
configure
ipsla operation 1
  type icmp echo
    timeout 5000
    destination address 223.255.254.254
    frequency 10
    statistics interval 30
    buckets 3
end

configure
ipsla operation 2
  type icmp path-echo
    destination address 223.255.254.254
    frequency 5
end

configure
ipsla reaction operation 1
  react timeout
   action trigger
   threshold type immediate
 exit
exit
  react rtt
   action logging
   threshold lower-limit 4 upper-limit 5
end
```

動作 1 はタイムアウトの発生をチェックします。 適用される場合、動作 1 はトリガー イベントを生成します。 **rtt** キーワードが 5 を超えると、エラーがログに記録されます。

動作 1 によってトリガー イベントが生成されると、動作 2 が開始されます。 次の例では、**ipsla reaction trigger** コマンドを使用して、反応トリガー動作を設定する方法を示します。

```
configure
ipsla reaction trigger 1 2
end
```

IP SLA MPLS LSP モニタリングの設定：例

次の例では、IP SLA MPLS LSP モニタリングの設定方法を説明します。

```
ipsla
 mpls lsp-monitor
```

```
 monitor 1
  type mpls lsp ping
   vrf SANJOSE
   scan interval 300
   scan delete-factor 2
   timeout 10000
   datasize request 256
   lsp selector ipv4 127.0.0.10
   force explicit-null
   reply dscp af
   reply mode router-alert
   ttl 30
   exp 1
   statistics hourly
    buckets 1
   !
  !
 !
 reaction monitor 1
  react timeout
   action logging
   threshold type immediate
  !
  react connection-loss
   action logging
   threshold type immediate
  !
 !
 schedule monitor 1
  frequency 300
  schedule period 120
  start-time 11:45:00 July 4
 !
!
mpls discovery vpn
 interval 600
!
!
```

LSP パス ディスカバリの設定：例

次の例では、LSP パス ディスカバリの設定方法を説明します。

```
configure
ipsla
 mpls lsp-monitor
  monitor 1
   type mpls lsp ping
    path discover
     path retry 12
     path secondary frequency both 12
```

その他の関連資料

次の項では、IP サービス レベル契約に関連する参考資料を紹介します。

関連資料

関連項目	マニュアル タイトル
IP サービス レベル契約のコマンド	『*Cisco ASR 9000 Series Aggregation Services Router System Monitoring Command Reference*』の「*IP* サービス レベル契約コマンド」モジュール
ユーザ グループとタスク ID に関する情報	『*Cisco ASR 9000 Series Aggregation Services Router System Security Configuration Guide*』の「*Configuring AAA Services*」モジュール

標準	タイトル
この機能でサポートされる新規の標準または変更された標準はありません。また、既存の標準のサポートは変更されていません。	—

MIB	MIB のリンク
—	Cisco IOS XR ソフトウェアを使用して MIB を検索およびダウンロードするには、http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml にある Cisco MIB Locator を使用し、[Cisco Access Products] メニューからプラットフォームを選択します。

RFC

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

シスコのテクニカル サポート

説明	リンク
シスコのテクニカル サポート Web サイトでは、製品、テクノロジー、ソリューション、技術的なヒント、およびツールへのリンクなどの、数千ページに及ぶ技術情報が検索可能です。Cisco.com に登録済みのユーザは、このページから詳細情報にアクセスできます。	*http://www.cisco.com/cisco/web/support/index.html*

第 5 章

ロギング サービスの実装

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新しいタスクと改訂された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。

（注） Cisco IOS XR ソフトウェアでのロギング サービスおよびこのモジュールの一覧で示されているロギング コマンドの詳細については、このモジュールの「関連資料，（388 ページ）」の項を参照してください。

ロギング サービスの実装の機能履歴

リリース	変更内容
リリース 3.7.2	この機能が導入されました。

ロギング サービス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）にロギング サービスを実装するには、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。

ロギング サービスの実装に関する情報

ロギング サービスを実装するには、次の概念を理解する必要があります。

システム ロギング プロセス

デフォルトでは、ルータは syslog メッセージを syslog プロセスに送信するように設定されます。syslog プロセスでは、ロギング バッファ、端末回線、syslog サーバのなどの syslog メッセージの宛先に対するメッセージの分配を制御します。 syslog プロセスはデフォルトでメッセージをコンソール端末にも送信します。

システム ロギング メッセージの形式

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 -group -severity -message -code* : *message-text*

次にサンプルの syslog メッセージを示します。

```
RP/0/RSP0/CPU0:Nov 28 23:56:53.826 : config[65710]: %SYS-5-CONFIG_I : Configured from console
 by console
```

この表では、Cisco IOS XR ソフトウェアの syslog メッセージの一般的な形式を説明します。

表 *26*：一般的な *syslog* メッセージ形式

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。

フィールド	説明
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。タイムスタンプの形式の修正，（374 ページ）の項を参照してください。
process-name	syslog メッセージを生成したプロセスのプロセスです。
[*pid*]	syslog メッセージを生成したプロセスのプロセス ID（pid）です。
%message -group- severity -message -code	syslog メッセージに関連付けられているメッセージ 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。

重複メッセージの抑制

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。 抑制とロギング プロセスは、ロギング ヒストリおよび外部 syslog サーバで同じです。

重複メッセージの抑制がイネーブルの場合、次の 2 つのタイプのイベントは異なる方法で処理されます。

- 新規メッセージ

  新規メッセージはいつでも即座にログに記録されます。

- 繰り返されるメッセージ

  繰り返されるメッセージは抑制対象です。 繰り返されるメッセージの抑制は、新しいメッセージが発生すると中断されます（メッセージの抑制の中断，（358 ページ）を参照）。

この機能の設定については、重複 syslog メッセージの抑制，（378 ページ）を参照してください。

メッセージの抑制の中断

抑制間隔のシーケンスは、異なるイベントが発生したときに中断されます。 すべての新しいイベントが最初に発生すると即座に記録されるため、新しいイベントによって現在の抑制間隔は中断されます。 これによりメッセージキューがクリアされるため、以前のシーケンスからの別のメッセージは新しいイベントとして扱われ、抑制シーケンスは再度開始されます。

繰り返しメッセージの要約は 0、30、120、および 600 秒間隔でのみ生成され、新しいイベントの発生時には生成されません。 各間隔で、任意の数のメッセージが抑制されているかどうかが確認されます。 その間隔で抑制されているメッセージが存在する場合は、その要約が表示され、メッセージキューがクリアされ、抑制シーケンスが最初から開始されます。 したがって、新しいイベントが抑制間隔の前に発生した場合、メッセージ キューおよびカウンタはクリアされ、間隔の時点では要約する抑制はありません。

次に例を示します。

```
RP/0/RP0/CPU0:router#show running-config logging
Mon Dec  3 12:30:26.346 UTC
logging archive
 device harddisk
 severity debugging
 file-size 4
 archive-size 100
!
logging console disable
logging 223.255.254.248
logging 223.255.254.249
logging suppress duplicates

RP/0/RP0/CPU0:router#
RP/0/RP0/CPU0:router#
RP/0/RP0/CPU0:router#
RP/0/RP0/CPU0:router#
RP/0/RP0/CPU0:router#run
Mon Dec  3 12:30:39.798 UTC
#
# while true
> do
> logger -s alert -c 1 "LOGGING SUPRESS DUPLICATE TESTING "
> done

RP/0/RP0/CPU0:router#
RP/0/RP0/CPU0:router#

//********************************************** MSGS ON REMOTE
SERVER************************************************************
Message from syslogd@[12.24.50.70.2.2] at Mon Dec  3 05:01:59 2012 ...
[12.24.50.70.2.2] 663231: last message repeated 2 times
Dec  3 05:01:58 [12.24.50.70.2.2] 663230: RP/0/RP1/CPU0:Dec  3 12:52:09.773 : logger[65786]:
 %OS-SYSLOG-1-LOG_ALERT_OWNER_PLANE :  LOGGING SUPRESS DUPLICATE TESTING
Dec  3 05:01:59 [12.24.50.70.2.2] 663231: last message repeated 2 times

Message from syslogd@[12.24.50.70.2.2] at Mon Dec  3 05:02:30 2012 ...
[12.24.50.70.2.2] 663232: last message repeated 110 times
Dec  3 05:02:30 [12.24.50.70.2.2] 663232: last message repeated 110 times

Message from syslogd@[12.24.50.70.2.2] at Mon Dec  3 05:04:31 2012 ...
[12.24.50.70.2.2] 663233: last message repeated 348 times
Dec  3 05:04:31 [12.24.50.70.2.2] 663233: last message repeated 348 times

Message from syslogd@[12.24.50.70.2.2] at Mon Dec  3 05:14:32 2012 ...
[12.24.50.70.2.2] 663234: last message repeated 1797 times
Dec  3 05:14:32 [12.24.50.70.2.2] 663234: last message repeated 1797 times
```

syslog メッセージの宛先

コンソール端末への syslog メッセージのロギングは、デフォルトでイネーブルです。 コンソール端末へのロギングをディセーブルにするには、グローバル コンフィギュレーション モードで **logging console disable** コマンドを使用します。 コンソールへのロギングを再度イネーブルにするには、グローバル コンフィギュレーション モードで **logging console** コマンドを使用します。

syslog メッセージは、コンソール以外の宛先に送信できます。たとえば、ロギング バッファ、syslog サーバ、コンソール以外の終端回線（vtys など）です。

この表では、syslog の宛先を指定するために使用するコマンドを一覧で示します。

表 *27*：*syslog* の宛先を設定するために使用するコマンド

コマンド	説明
logging buffered	syslog メッセージの宛先としてロギング バッファを指定します。
logging {*hostname* | *ip-address*}	syslog サーバ ホストを syslog メッセージの宛先として指定します。
logging monitor	コンソール以外の端末回線を syslog メッセージの宛先として指定します。

logging buffered コマンドでは、ロギング メッセージをロギング バッファにコピーします。 循環バッファであるため、バッファがいっぱいになると、古いメッセージが新しいメッセージで置き換えられます。 バッファに記録された syslog メッセージを表示するには、**show logging** コマンドを使用します。最初に表示されるメッセージは、バッファ内で最も古いメッセージです。ロギング バッファの現在の内容をクリアするには、**clear logging** コマンドを使用します。 ロギング バッファへのロギングをディセーブルにするには、グローバル コンフィギュレーション モードで **no logging buffered** コマンドを使用します。

logging コマンドは、ロギング メッセージを受信する syslog サーバ ホストを識別します。 このコマンドを何度も発行すると、ロギング メッセージを受信する syslog サーバのリストが作成されます。 指定された IP アドレスやホスト名を持つ syslog サーバを、利用可能な syslog サーバのリストから削除するには、グローバル コンフィギュレーション モードで **no logging** コマンドを使用します。

logging monitor コマンドは、vtys などのコンソール端末以外の端末回線への syslog メッセージのロギングをグローバルにイネーブルにします。 コンソール以外の端末回線へのロギングをディセーブルにするには、グローバル コンフィギュレーション モードで **no logging monitor** コマンドを使用します。

コンソール以外の宛先に syslog メッセージを送信するためのガイドライン

ロギング プロセスはコンソール端末以外の宛先に syslog メッセージを送信し、プロセスはデフォルトでイネーブルです。 ロギング バッファ、端末回線、syslog サーバへのロギングはイネーブルです。

現在の端末セッションのロギング

logging monitor コマンドは、コンソール端末以外の端末回線への syslog メッセージのロギングをグローバルにイネーブルにします。 **logging monitor** コマンドをイネーブルにしたら、**terminal monitor** コマンドを使用して、端末セッション中に syslog メッセージを表示します。

端末セッション中、syslog メッセージの端末へのロギングをディセーブルにするには、EXEC モードで **terminal monitor disable** コマンドを使用します。 **terminal monitor disable** コマンドは、現在の端末セッションのロギングのみをディセーブルにします。

現在の端末セッションの syslog メッセージのロギングを再度イネーブルにするには、EXEC モードで **terminal monitor** コマンドを使用します。

（注） **terminal monitor** コマンドおよび **terminal monitor disable** コマンドはローカルで設定され、端末セッションが終了すると有効ではなくなります。

syslog サーバに送信された syslog メッセージ

syslog サーバに送信された syslog メッセージを管理しやすくするために、Cisco IOS XR ソフトウェアには次の機能が搭載されています。

- UNIX システム ファシリティ
- ホスト名プレフィックス ロギング
- ソース インターフェイス ロギング

UNIX システム ロギング ファシリティ

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。 これらの UNIX システム ファシリティの詳細については、ご使用の UNIX オペレーティング システムのオペレータ マニュアルを参照してください。 syslog の形式は、Berkeley Standard Distribution（BSD）UNIX バージョン 4.3 と互換性があります。

この表では、*type* 引数に指定できるファシリティ タイプ キーワードを説明します。

表 *28* : ロギング ファシリティ タイプのキーワード

ファシリティ タイプのキーワード	説明
auth	認可システムを示します。
cron	cron ファシリティを示します。
daemon	システム デーモンを示します。
kern	カーネルを示します。
local0–7	ローカルで定義されたメッセージ用に予約されています。
lpr	回線プリンタ システムを示します。
mail	メール システムを示します。
news	USENET ニュースを示します。
sys9	システムの使用を示します。
sys10	システムの使用を示します。
sys11	システムの使用を示します。
sys12	システムの使用を示します。
sys13	システムの使用を示します。
sys14	システムの使用を示します。
syslog	システム ログを示します。
user	ユーザ プロセスを示します。
uucp	UNIX から UNIX へのコピー システムを示します。

ホスト名プレフィックス ロギング

syslog サーバに送信されたシステム ロギング メッセージを管理しやすくするために、Cisco IOS XR ソフトウェアではホスト名プレフィックス 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付加するには、グローバル コンフィギュレーション モードで、**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 メッセージが syslog サーバに送信されるとき、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出るかにかかわらず、すべての syslog メッセージに同じ IP アドレスを含めるように設定するには、グローバル コンフィギュレーション モードで **logging source-interface** コマンドを使用します。

UNIX syslog デーモンの設定

4.3 BSD UNIX システム上で syslog デーモンを設定するには、/etc/syslog.conf ファイルに次のような行を含めます。

```
local7.debug /usr/adm/logs/cisco.log
```

debugging キーワードでは syslog レベルを指定します。他のキーワードの一般的な説明については、表 32：syslog メッセージの重大度, （365 ページ）を参照してください。 **local7** キーワードでは、使用するロギング ファシリティを指定します。他のキーワードの一般的な説明については、表 28：ロギング ファシリティ タイプのキーワード, （360 ページ）を参照してください。

syslog デーモンは、次のフィールドで指定されたファイルに、このレベルまたはより重大なレベルのメッセージを送信します。 このファイルは、syslog デーモンに書き込み権限がある既存ファイルである必要があります。

ローカル ストレージ デバイスでのロギング 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。

アーカイブ属性の設定

ロギング アーカイブを作成して、ロギング メッセージを収集および保存する方法を指定するには、グローバル コンフィギュレーション モードで **logging archive** コマンドを使用します。 **logging archive** コマンドでは、ロギング アーカイブ サブモードを開始します。このモードでは、syslog をアーカイブするための属性を設定できます。

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 *29*：*syslog* アーカイブ属性を設定するために使用するコマンド

コマンド	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。
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の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。詳細については、重大度，（364ページ）を参照してください。

ストレージ ディレクトリのアーカイブ

デフォルトでは、syslog アーカイブは <device>/var/log のディレクトリに格納されます。 個別のアーカイブ ファイルはアーカイブが作成された年月日に基づいてサブディレクトリに保存されます。 たとえば、2006/02/26 に作成されたアーカイブ ファイルは、次のディレクトリに保存されます。

```
harddisk:/var/log/2006/02/26
```

重大度

宛先に送信される syslog メッセージの重大度を指定して、ロギングの宛先に送信されるメッセージの数を制限できます（重大度の定義は表 32：syslog メッセージの重大度, （365 ページ）を参照）。

この表では、syslog メッセージの重大度を制御するコマンドを一覧で示します。

表 *30*：*syslog* メッセージの重大度を制御するために使用するコマンド

コマンド	説明
logging buffered [*severity*]	ロギング バッファに送信する syslog メッセージを重大度に基づいて制限します。
logging console [*severity*]	コンソール端末に送信する syslog メッセージを重大度に基づいて制限します。
logging monitor [*severity*]	端末回線に送信する syslog メッセージを重大度に基づいて制限します。
logging trap [*severity*]	syslog サーバに送信する syslog メッセージを重大度に基づいて制限します。
severity *severity*	syslog アーカイブに送信する syslog メッセージを重大度に基づいて制限します。

logging buffered、**logging console**、**logging monitor**、**logging traps** コマンドでは、指定した重大度番号以下の syslog メッセージがそれぞれの宛先に送信されないように制限します。この番号は *severity* 引数で指定します。

（注） 重大度の番号が小さい syslog メッセージは、より重要なイベントであることを示します。 重大度の定義については、表 32：syslog メッセージの重大度, （365 ページ）を参照してください。

ロギング ヒストリ表

snmp-server enable traps syslog コマンドで、簡易ネットワーク管理プロトコル（SNMP）ネットワーク管理ステーション（NMS）に送信するようにsyslogメッセージ トラップをイネーブルにしている場合、ルータの履歴テーブルに送信および保存されるメッセージのレベルを変更できます。また履歴テーブルに保存されるメッセージ数も変更できます。

SNMP トラップは宛先への到達が保証されていないため、メッセージは履歴テーブルに格納されます。デフォルトでは、syslogトラップがイネーブルでない場合でも、レベル警告および上記（表 32：syslog メッセージの重大度，（365 ページ） を参照）の 1 つのメッセージが履歴テーブルに保存されます。

この表では、ロギング ヒストリ表の重大度と表のサイズのデフォルトを変更するために使用されるコマンドを一覧で示します。

表 *31*：ロギング ヒストリ表のコマンド

コマンド	説明
logging history *severity*	履歴ファイルに保存されてSNMPサーバに送信されるsyslogメッセージのデフォルトの重大度を変更します。
logging history size *number*	履歴テーブルに保存できるsyslogメッセージの数を変更します。

（注） 表 32：syslog メッセージの重大度，（365 ページ） に、level キーワードおよび重大度を示します。 SNMP を使用する場合、重大度の値は +1 を使用します。 たとえば、**emergency** は 0 ではなく 1 になり、**critical** は 2 ではなく 3 になります。

Syslog メッセージの重大度の定義

この表では、*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に順番に一覧で示します。

表 *32*：*syslog* メッセージの重大度

重大度のキーワード	レベル	説明	syslog 定義
emergencies	0	システムが使用不可	LOG_EMERG
alerts	1	即時処理が必要	LOG_ALERT

重大度のキーワード	レベル	説明	syslog 定義
critical	2	クリティカルな状態	LOG_CRIT
errors	3	エラー状態	LOG_ERR
warnings	4	警告状態	LOG_WARNING
notifications	5	正常だが注意を要する状態	LOG_NOTICE
informational	6	情報メッセージだけ	LOG_INFO
debugging	7	デバッグ メッセージ	LOG_DEBUG

syslog 重大度コマンドのデフォルト

この表では、*severity* 引数をサポートするコマンドのデフォルトの重大度設定を一覧で示します。

表 *33*：重大度コマンドのデフォルト

コマンド	デフォルトの重大度キーワード	レベル
logging buffered	debugging	7
logging console	informational	6
logging history	warnings	4
logging monitor	debugging	7
logging trap	informational	6

ロギング サービスの実装方法

ここでは、次の手順について説明します。

システム ロギング メッセージ宛先の設定

このタスクでは、コンソール端末以外の宛先へのロギングを設定する方法を説明します。

概念の情報については、syslog メッセージの宛先, （359 ページ）の項を参照してください。

手順の概要

1. **configure**
2. **logging buffered** [*size* | *severity*]
3. **logging monitor** [*severity*]
4. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**
5. **terminal monitor**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**logging buffered** [*size*	*severity*] 例： `RP/0/RSP0/CPU0:router(config)# logging buffered severity warnings`	ロギング バッファを syslog メッセージの宛先として指定し、ロギング バッファのサイズを設定し、ロギング バッファに送信される syslog メッセージを重大度に基づいて制限します。 • *size* 引数のデフォルト値は 4096 バイトです。 • *severity* 引数のデフォルト値は **debugging** です。 • *severity* 引数のキーワード オプションは、**emergencies**、**alerts**、**critical**、**errors**、**warnings**、**notifications**、**informational**、**debugging** です。 • デフォルトでは、*severity* 引数の重大度や *size* 引数のバッファ サイズを指定せずにこのコマンドを入力すると、重大度が **debugging** に、バッファ サイズが 4096 バイトに設定されます。
ステップ 3	**logging monitor** [*severity*] 例： `RP/0/RSP0/CPU0:router(config)# logging monitor critical`	コンソール端末以外の端末回線を syslog メッセージの宛先として指定し、端末回線に送信されるメッセージの数を重大度に基づいて制限します。 • *severity* 引数のキーワード オプションは、**emergencies**、**alerts**、**critical**、**errors**、**warnings**、**notifications**、**informational**、**debugging** です。 • デフォルトでは、*severity* 引数の重大度を指定せずにこのコマンドを入力すると、重大度は**debugging** に設定されます。

	コマンドまたはアクション	目的

ステップ 4	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。

ステップ 5	**terminal monitor** 例： `RP/0/RSP0/CPU0:router# terminal monitor`	現在の端末セッションにおける syslog メッセージの表示をイネーブルにします。 （注） 現在の端末の syslog メッセージのロギングは、**terminal monitor disable** コマンドでディセーブルにできます。 • 現在のセッションのメッセージのロギングが **terminal monitor disable** コマンドでディセーブルにされている場合、現在のセッションの syslog メッセージの表示を再度イネーブルにするには、このコマンドを使用します。 （注） このコマンドは EXEC モード コマンドであるため、ローカルで設定され、現在のセッションが終了するとイネーブルではなくなります。

リモート サーバへのロギングの設定

このタスクでは、リモート syslog サーバへのロギングを設定する方法を説明します。

はじめる前に

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。

手順の概要

1. configure
2. **logging** {*ip-address* | *hostname*}
3. **logging trap** [*severity*]
4. **logging facility** [*type*]
5. **logging hostnameprefix** *hostname*
6. **logging source-interface** *type interface-path-id*
7. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	configure 例： `RP/0/RSP0/CPU0:router#`	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 2	**logging** {*ip-address* | *hostname*} 例： `RP/0/RSP0/CPU0:router(config)# logging 10.3.32.154`	syslog サーバ ホストを syslog メッセージの宛先として指定します。 • このコマンドを何度も発行すると、ロギング メッセージを受信する syslog サーバのリストが作成されます。
ステップ 3	**logging trap** [*severity*] 例： `RP/0/RSP0/CPU0:router(config)#`	syslog サーバに送信する syslog メッセージを重大度に基づいて制限します。 • デフォルトでは、*severity* 引数の重大度を指定せずにこのコマンドを入力すると、重大度は **informational** に設定されます。
ステップ 4	**logging facility** [*type*] 例： `RP/0/RSP0/CPU0:router(config)# logging facility kern`	（任意）syslog ファシリティを設定します。 • デフォルトでは、*type* 引数にファシリティ タイプを指定せずにこのコマンドを入力すると、ファシリティは **local-7** に設定されます。

	コマンドまたはアクション	目的
ステップ 5	**logging hostnameprefix** *hostname* 例： `RP/0/RSP0/CPU0:router(config)#`	（任意）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。 ヒント　ホスト名プレフィックス ロギングは、syslog サーバで受信される syslog メッセージを並べ替えるときに便利です。
ステップ 6	**logging source-interface** *type interface-path-id* 例： `RP/0/RSP0/CPU0:router(config)#`	（任意）syslog 送信元アドレスを設定します。 • 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出るかにかかわらず、ルータから送信されるすべての 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、このコマンドを使用します。

ステップ 7	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

ロギング ヒストリ表の設定

このタスクでは、ロギング ヒストリ表を設定する方法を説明します。

概念の情報については、重大度, （364 ページ）の項を参照してください。

はじめる前に

SNMP NMS へのメッセージのロギングは、**snmp-server enable traps syslog** コマンドでイネーブルにします。 SNMP の詳細については、関連資料, （388 ページ）の項を参照してください。

手順の概要

1. **configure**
2. **logging history** *severity*
3. **logging history size** *number*
4. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**
5. **show logging history**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**logging history** *severity* 例： `RP/0/RSP0/CPU0:router(config)# logging history errors`	履歴ファイルに保存されて SNMP サーバに送信される syslog メッセージのデフォルトの重大度を変更します。 • デフォルトでは、重大度が **warnings** 以下の syslog メッセージが履歴ファイルに格納され、SNMP サーバに送信されます。
ステップ 3	**logging history size** *number* 例： `RP/0/RSP0/CPU0:router(config)# logging history size 200`	履歴テーブルに保存できる syslog メッセージの数を変更します。 • デフォルトでは、1 つの syslog メッセージが履歴テーブルに格納されます。 （注） 履歴テーブルが一杯になると（メッセージの数がこのコマンドで指定した最大数に達すると）、新しいメッセージを保存できるよう、最も古いメッセージがテーブルから削除されます。

	コマンドまたはアクション	目的

ステップ 4	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。

ステップ 5	**show logging history** 例： `RP/0/RSP0/CPU0:router# show logging history`	（任意）syslog 履歴テーブルの状態についての情報を表示します。

コンソール端末およびロギング バッファへのロギングの修正

このタスクでは、コンソール端末およびロギング バッファのロギングの設定を変更する方法を説明します。

（注） デフォルトでは、ロギングはイネーブルです。

手順の概要

1. **configure**
2. **logging buffered** [*size* | *severity*]
3. **logging console** [*severity*]
4. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**logging buffered** [*size* | *severity*] 例： `RP/0/RSP0/CPU0:router(config)# logging buffered size 60000`	ロギング バッファを syslog メッセージの宛先として指定し、ロギング バッファのサイズを設定し、ロギング バッファに送信される syslog メッセージを重大度に基づいて制限します。 • *size* 引数のデフォルトは 4096 バイトです。 • *severity* 引数のデフォルトは **debugging** です。 • *severity* 引数のキーワード オプションは、**emergencies**、**alerts**、**critical**、**errors**、**warnings**、**notifications**、**informational**、**debugging** です。 • デフォルトでは、*severity* 引数の重大度や *size* 引数のバッファ サイズを指定せずにこのコマンドを入力すると、重大度が **debugging** に、バッファ サイズが 4096 バイトに設定されます。
ステップ 3	**logging console** [*severity*] 例： `RP/0/RSP0/CPU0:router(config)# logging console alerts`	コンソール端末に送信するメッセージを重大度に基づいて制限します。 • デフォルトでは、syslog メッセージは **informational** の重大度でコンソール端末のログに記録されます。 • *severity* 引数のキーワード オプションは、**emergencies**、**alerts**、**critical**、**errors**、**warnings**、**notifications**、**informational**、**debugging** です。 • *severity* 引数の重大度を指定せずにこのコマンドを入力すると、重大度は **informational** に設定されます。

	コマンドまたはアクション	目的
		（注） **logging console disable** コマンドでコンソール端末へのロギングがディセーブルにされている場合に、ロギングを再度イネーブルにするには、このコマンドを使用します。

ステップ 4	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーションファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。

タイムス タンプの形式の修正

このタスクでは、syslog メッセージおよびデバッグ メッセージのタイム スタンプ形式を変更する方法を説明します。

手順の概要

1. **configure**
2. 次のいずれかを実行します。
   - **service timestamps log datetime** [**localtime**] [**msec**] [**show-timezone**]
   - **service timestamps log uptime**
3. 次のいずれかを実行します。
   - **service timestamps debug datetime** [**localtime**] [**msec**] [**show-timezone**]
   - **service timestamps debug uptime**
4. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	次のいずれかを実行します。
	• **service timestamps log datetime** [**localtime**] [**msec**] [**show-timezone**]
	• **service timestamps log uptime**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# service timestamps log datetime localtime msec`
	または
	`RP/0/RSP0/CPU0:router(config)# service timestamps log uptime`
		syslog メッセージのタイム スタンプ形式を変更します。
		• デフォルトでは、タイム スタンプはイネーブルです。 デフォルトのタイム スタンプ形式は month day HH:MM:SS です。
		• **service timestamps log datetime** コマンドを発行すると、日時のタイム スタンプが付くように syslog メッセージが設定されます。
		◦任意で **localtime** キーワードを指定すると、タイム スタンプにローカル タイムゾーンが含まれます。
		◦任意で **msec** キーワードを指定すると、タイム スタンプにミリ秒が含まれます。
		◦任意で **show-timezone** キーワードを指定すると、タイム スタンプにタイムゾーン情報が含まれます。
		• **service timestamps log uptime** コマンドを発行すると、ルータが最後にリブートされたときから経過した時間のタイムスタンプが付くように syslog メッセージが設定されます。

	コマンドまたはアクション	目的
		◦ **service timestamps log uptime** コマンドでは、タイム スタンプを HHHH:MM:SS に設定します。これはルータが最後にリブートされたときからの時間を示します。

ステップ 3	次のいずれかを実行します。
	• **service timestamps debug datetime** [**localtime**] [**msec**] [**show-timezone**]
	• **service timestamps debug uptime**
	例：
	`RP/0/RSP0/CPU0:router(config)# service timestamps debug datetime msec show-timezone`
	または
	`RP/0/RSP0/CPU0:router(config)# service timestamps debug uptime`
		デバッグ メッセージのタイム スタンプ形式を変更します。
		• デフォルトでは、タイム スタンプはイネーブルです。 デフォルトのタイム スタンプ形式は month day HH:MM:SS です。
		• **service timestamps log datetime** コマンドを発行すると、日時のタイム スタンプが付くようにデバッグ メッセージが設定されます。
		◦ 任意で **localtime** キーワードを指定すると、タイム スタンプにローカル タイムゾーンが含まれます。
		◦ 任意で **msec** キーワードを指定すると、タイム スタンプにミリ秒が含まれます。
		◦ 任意で **show-timezone** キーワードを指定すると、タイム スタンプにタイムゾーン情報が含まれます。
		• **service timestamps log uptime** コマンドを発行すると、ネットワーク デバイスが最後にリブートされたときから経過した時間のタイム スタンプが付くようにデバッグ メッセージが設定されます。
		ヒント キーワードや引数を指定せずに **service timestamps** コマンドを入力すると、**service timestamps debug uptime** コマンドと同じように機能します。

ステップ 4	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。

	コマンドまたはアクション	目的
		◦cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。 • 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

タイム スタンプのディセーブル化

このタスクでは、syslog メッセージにタイム スタンプが含まれないようにする方法を説明します。

手順の概要

1. **configure**
2. 次のいずれかを実行します。
   - **service timestamps disable**
   - **no service timestamps** [**debug** | **log**] [**datetime** [**localtime**] [**msec**] [**show-timezone**]] | **uptime**]
3. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	次のいずれかを実行します。	syslog メッセージにタイム スタンプが含まれないようにします。

	コマンドまたはアクション	目的
	• **service timestamps disable** • **no service timestamps** [**debug** | **log**] [**datetime** [**localtime**] [**msec**] [**show-timezone**]] | **uptime**]	（注） どちらのコマンドでも syslog メッセージのタイム スタンプをディセーブルにできますが、**service timestamps disable** コマンドを指定すると、設定にコマンドが保存されます。**service timestamps** コマンドの **no** 形式を指定すると、設定からコマンドが削除されます。

ステップ 3	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。

重複 syslog メッセージの抑制

このタスクでは、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説明します。

手順の概要

1. **configure**
2. **logging suppress duplicates**
3. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**logging suppress duplicates** 例： `RP/0/RSP0/CPU0:router(config)# logging suppress duplicates`	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します。 **注意** デバッグセッション中にこのコマンドがイネーブルの場合、切り離して解決しようとしている問題に関する重要な情報を見落とす可能性があります。こうした場合は、このコマンドをディセーブルにすることを検討してください。

ステップ 3	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。

	コマンドまたはアクション	目的
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。

リンクステータス syslog メッセージのロギングのディセーブル化

このタスクでは、論理リンクおよび物理リンクのりんくすてーたす syslog メッセージのロギングをディセーブルにする方法を説明します。

リンクステータス メッセージのロギングをイネーブルにした場合、ルータで大量のリンクステータス（アップダウン）のシステム ロギング メッセージが生成される場合があります。 リンクステータス syslog メッセージのロギングをディセーブルにすると、ログに記録されるメッセージの数を減らすことができます。

手順の概要

1. **configure**
2. **logging events link-status disable**
3. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**logging events link-status disable** 例： `RP/0/RSP0/CPU0:router(config)# logging events link-status disable`	ソフトウェア（論理）リンクおよび物理リンクのリンクステータス syslog メッセージのロギングをディセーブルにします。 • 物理リンクでは、リンクステータス syslog メッセージのロギングはデフォルトでイネーブルです。

	コマンドまたはアクション	目的
		• 論理リンクと物理リンクの両方で、リンクステータス syslog メッセージをイネーブルにするには、**logging events link-status software-interfaces** コマンドを使用します。 • リンクステータス syslog メッセージを物理リンクでのみイネーブルするには、**no logging events link-status** コマンドを使用します。

ステップ 3	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

システム ロギング メッセージの表示

このタスクでは、ロギング バッファに保存されている syslog メッセージを表示する方法を説明します。

（注） コマンドは任意の順序で入力できます。

手順の概要

1. **show logging**
2. **show logging location** *node-id*
3. **show logging process** *name*
4. **show logging string** *string*
5. **show logging start** *month day hh*:*mm*:*ss*
6. **show logging end** *month day hh*:*mm*:*ss*

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**show logging** 例： `RP/0/RSP0/CPU0:router# show logging`	ロギング バッファに保存されているすべての syslog メッセージを表示します。
ステップ 2	**show logging location** *node-id* 例： `RP/0/RSP0/CPU0:router# show logging location 0/1/CPU0`	指定されたノードからの syslog メッセージを表示します。
ステップ 3	**show logging process** *name* 例： `RP/0/RSP0/CPU0:router# show logging process init`	指定したプロセスに関連する syslog メッセージを表示します。
ステップ 4	**show logging string** *string* 例： `RP/0/RSP0/CPU0:router# show logging string install`	指定したストリングを含む syslog メッセージを表示します。
ステップ 5	**show logging start** *month day hh*:*mm*:*ss* 例： `RP/0/RSP0/CPU0:router# show logging start december 1 10:30:00`	指定した日時以降に生成されたロギング バッファ内の syslog メッセージを表示します。
ステップ 6	**show logging end** *month day hh*:*mm*:*ss* 例： `RP/0/RSP0/CPU0:router# show logging end december 2 22:16:00`	指定した日時以前に生成されたロギング バッファ内の syslog メッセージを表示します。

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

このタスクでは、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方法を説明します。

はじめる前に

（注）

- ローカル ストレージデバイスには、アーカイブ ファイルを格納するために利用できる十分な領域が必要です。 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。
- ストレージデバイスの syslog メッセージのアーカイブは、設定された重大度レベルのメッセージが 10 MB または 20480 メッセージを超えたときにのみ開始されます。 たとえば、設定された重大度レベルが緊急の場合、重大度レベルが緊急の syslog メッセージが 10 MB または 20480 個記録された場合にのみアーカイブが開始されます。

手順の概要

1. configure
2. **logging archive**
3. **device** {**disk0** | **disk1** | **harddisk**}
4. **frequency** {**daily** | **weekly**}
5. **severity** *severity*
6. **archive-length** *weeks*
7. **archive-size** *size*
8. **file-size** *size*
9. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	configure 例： `RP/0/RSP0/CPU0:router# 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。

	コマンドまたはアクション	目的
ステップ 2	**logging archive** 例： `RP/0/RSP0/CPU0:router(config)# logging archive`	ロギング アーカイブ コンフィギュレーション モードを開始します。
ステップ 3	**device** {**disk0** | **disk1** | **harddisk**} 例： `RP/0/RSP0/CPU0:router(config-logging-arch)# device disk1`	syslog のロギングに使用するデバイスを指定します。 • この手順は必須です。 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。 • デフォルトでは、ログは <device>/var/log のディレクトリに作成されます
ステップ 4	**frequency** {**daily** | **weekly**} 例： `RP/0/RSP0/CPU0:router(config-logging-arch)# frequency weekly`	（任意）ログを収集する頻度を日次または週次で指定します。デフォルトでは、ログは毎日収集されます。
ステップ 5	**severity** *severity* 例： `RP/0/RSP0/CPU0:router(config-logging-arch)# severity warnings`	（任意）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。 重大度は次のとおりです。 • emergencies • alerts • critical • errors • warnings • notifications • informational • debugging 詳細については、Syslog メッセージの重大度の定義, （365 ページ）の項を参照してください。

	コマンドまたはアクション	目的
ステップ 6	**archive-length** *weeks* 例： `RP/0/RSP0/CPU0:router(config-logging-arch)# archive-length 6`	（任意）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。 デフォルトでは、アーカイブ ログは 4 週間保存されます。
ステップ 7	**archive-size** *size* 例： `RP/0/RSP0/CPU0:router(config-logging-arch)# archive-size 50`	（任意）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。 デフォルトのアーカイブ サイズは 20 MB です。
ステップ 8	**file-size** *size* 例： `RP/0/RSP0/CPU0:router(config-logging-arch)# file-size 10`	（任意）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。 デフォルトでは、最大ファイルサイズは 1 メガバイトです。

ステップ 9	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。

	コマンドまたはアクション	目的

ロギング サービスを実装するための設定例

ここでは、次の設定例について説明します。

コンソール端末およびロギング バッファへのロギングの設定：例

次の例では、ロギング バッファへのロギングがイネーブルであり、コンソール端末に送信される syslog メッセージの重大度は **critical** 重大度以下の syslog メッセージに制限され、ロギング バッファのサイズが 60,000 バイトに設定されているロギングの設定を示します。

```
!
logging console critical
logging buffered 60000
!
```

syslog メッセージの宛先の設定：例

次の例では、ロギングがコンソール端末以外の宛先に設定されているロギングの設定を示します。この設定は次のようになります。

- コンソール端末以外の宛先に対するロギングはイネーブル。
- **warnings** 重大度以下の syslog メッセージは syslog サーバ ホストに送信される。
- **critical** 重大度より低い syslog メッセージは端末回線に送信される。
- ロギング バッファのサイズは 60,000 バイトに設定される。
- IP アドレス 172.19.72.224 の syslog サーバ ホストが syslog メッセージの受信先として設定される。

```
!
logging trap warnings
logging monitor critical
logging buffered 60000
logging 172.19.72.224
!
```

ロギング ヒストリ表の設定：例

次の例では、ロギング ヒストリ表のサイズが 200 エントリであり、ロギング ヒストリ表に送信される syslog メッセージの重大度が **errors** の重大度のメッセージに制限されているロギングの設定を示します。

```
logging history size 200
logging history errors
```

タイム スタンプの修正：例

次の例では、month date HH:MM:SS タイムゾーンの形式に従うようにタイム スタンプが設定されているタイム スタンプの設定を示します。

```
service timestamps log datetime show-timezone
```

次の例では、HH:MM:SS ミリ秒のタイムゾーンの形式に従うようにタイム スタンプが設定されているタイム スタンプの設定を示します。

```
service timestamps log datetime msec show-timezone
```

ロギング アーカイブの設定：例

次の例では、ロギング アーカイブを設定する方法とアーカイブ属性を定義する方法を示します。

```
RP/0/RSP0/CPU0:router# configure
RP/0/RSP0/CPU0:router(config)# logging archive
RP/0/RSP0/CPU0:router(config-logging-arch)# device disk1
RP/0/RSP0/CPU0:router(config-logging-arch)# frequency weekly
RP/0/RSP0/CPU0:router(config-logging-arch)# severity warnings
RP/0/RSP0/CPU0:router(config-logging-arch)# archive-length 6
RP/0/RSP0/CPU0:router(config-logging-arch)# archive-size 50
RP/0/RSP0/CPU0:router(config-logging-arch)# file-size 10
```

次の作業

アラーム ログ相関を設定するには、『*Cisco ASR 9000* シリーズ アグリゲーション サービス ルータ システム モニタリング コンフィギュレーション ガイド』の「アラームおよびアラーム ログ相関の実装とモニタリング」モジュールを参照してください。

その他の関連資料

次の項では、Cisco IOS XR ソフトウェアへのロギング サービスの実装に関連する参考資料を紹介します。

関連資料

関連項目	マニュアル タイトル
ロギング サービス コマンド リファレンス	『*Cisco ASR 9000 Series Aggregation Services Router System Monitoring Command Reference*』の「*Logging Services Commands*」モジュール
オンボード障害ロギング（OBFL）コンフィギュレーション	『*Cisco ASR 9000* シリーズ アグリゲーション サービス ルータ システム モニタリング コンフィギュレーション ガイド』の「*Onboard Failure Logging Commands*」モジュール
オンボード障害ロギング（OBFL）コマンド	『*Cisco ASR 9000 Series Aggregation Services Router System Monitoring Command Reference*』の「*Onboard Failure Logging Commands*」モジュール
アラームおよびロギング相関コマンド	『*Cisco ASR 9000 Series Aggregation Services Router System Monitoring Command Reference*』の「*Alarm Management and Logging Correlation Commands*」モジュール
アラームおよびロギング相関コンフィギュレーションおよびモニタリング タスク	『*Cisco ASR 9000* シリーズ アグリゲーション サービス ルータ システム モニタリング コンフィギュレーション ガイド』の「アラームおよびアラーム ログ相関の実装とモニタリング」モジュール
SNMP コマンド	『*Cisco ASR 9000 Series Aggregation Services Router System Monitoring Command Reference*』の「*SNMP Commands*」モジュール
SNMP の設定作業	『*Cisco ASR 9000* シリーズ アグリゲーション サービス ルータ システム モニタリング コンフィギュレーション ガイド』の「*Implementing SNMP*」モジュール
Cisco IOS XR のスタートアップ マニュアル	『*Cisco ASR 9000 Series Aggregation Services Router Getting Started Guide*』
ユーザ グループとタスク ID に関する情報	『*Cisco ASR 9000 Series Aggregation Services Router System Security Command Reference*』の「*Configuring AAA Services*」モジュール

標準

標準	タイトル
この機能でサポートされる新規の標準または変更された標準はありません。また、既存の標準のサポートは変更されていません。	—

MIB

MIB	MIB のリンク
—	Cisco IOS XR ソフトウェアを使用して MIB を検索およびダウンロードするには、http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml にある Cisco MIB Locator を使用し、[Cisco Access Products] メニューからプラットフォームを選択します。

RFC

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

シスコのテクニカル サポート

説明	リンク
シスコのテクニカル サポート Web サイトでは、製品、テクノロジー、ソリューション、技術的なヒント、およびツールへのリンクなどの、数千ページに及ぶ技術情報が検索可能です。Cisco.com に登録済みのユーザは、このページから詳細情報にアクセスできます。	*http://www.cisco.com/cisco/web/support/index.html*

第 6 章

オンボード障害ロギング

OBFLは、現場交換可能ユニット（FRU）のブート、環境、および重大なハードウェアデータを収集し、その情報をFRUの不揮発性メモリに保存します。 この情報は、障害やエラーが発生したときにトラブルシューティング、テスティング、診断に使用され、ハードウェアのトラブルシューティングと根本原因の分離解析をより正確に行えるようにします。 保存されているOBFLデータは障害時に取得でき、カードがブートしなくてもアクセスできます。

OBFLはデフォルトでオンになっているため、カードをインストールするとただちにデータが収集され保存されます。 問題が発生すると、このデータから過去の環境条件、アップタイム、ダウンタイム、エラー、その他の動作状態に関する情報が読み取られます。

注意 OBFL はすべてのカードでデフォルトでアクティブになっています。 OBFL データは FRU の問題の診断および解決に使用されるため、明確な理由なしにOBFL を非アクティブ化しないでください。

（注） OBFL コマンド、コンソール ロギング、アラーム、ロギング相関については関連資料，（388 ページ）を参照してください。

OBFL の実装の機能履歴

リリース	変更内容
リリース 3.7.2	この機能が導入されました。

- 次の作業，401 ページ
- その他の関連資料，401 ページ

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。

OBFL について

ロギング サービスを実装するには、次の概念を理解する必要があります。

データ収集タイプ

OBFL はベースラインによって収集される情報とイベントによって生成される情報の両方を、サポートされているそれぞれの OBFL 対応カードの不揮発性メモリに収集して格納します。 次のようなデータが収集されます。

- FRU パーツ シリアル番号
- OS のバージョン
- ブート時間
- 実行時間合計（使用された時間）
- ブート ステータス
- ブート時の温度と電圧
- 温度と電圧の履歴
- その他のボード特有のエラー

このデータは 2 つの異なる方法で収集されます。ベースライン データ収集とイベントによって生成されるデータ収集です。

ベースライン データ収集

ベースラインデータは、ハードウェアやソフトウェアの障害とは関係なく保存されます。 次の内容が含まれています。

データ型	詳細
インストール	初期ブート時および過去 9 回のブートで格納されたシャーシ名とスロット番号です。
温度	吸気口と高温ポイントの温度は、ブート後 10 分記録されます。
実行時間	はじめにインストールされてからの実行時間の合計です。 これは 30 分間隔でローカル ルータの時計に基づきます。

イベントによって生成されるデータ収集

イベントによって生成されるデータには、カードエラーイベントもあります。障害イベントは、カード クラッシュ、メモリ エラー、ASIC リセット、および類似症状を示すハードウェア障害です。

データ型	詳細	
環境エラー	温度エラー	吸気口と高温ポイントの温度エラー
	電圧エラー	+5、MBUS +5、+3.3、+2.2 電圧エラー
		次のような温度または電圧のイベントが発生すると、環境の読み取りがログに記録されます。 • 通常の範囲を超過 • 10% を超える変更 • 範囲内で 5 分を超えて返される リブートすると、これらの環境読み込みは 1 つの環境履歴レコードに統合されます。このレコードには、環境読み込みの連続したセットの期間と通常範囲の超過量が示されます。

データ型	詳細	
カレンダー時間	ディセーブル	OBFL ロギングがグローバル コンフィギュレーション モードや管理コンフィギュレーション モードで **hw-module** {**all** | **subslot** *node-id*} **logging onboard disable** コマンドを使ってディセーブルにされた時刻です。
	クリア済み	OBFL ロギングが EXEC モードや管理 EXEC コンフィギュレーション モードで**clear logging onboard** コマンドを使ってクリアされた時刻です。
	0 にリセット	ラインカードの合計実行時間が EXEC モードや管理 EXEC モードで **clear logging onboard** コマンドを使ってゼロにリセットされた時刻です。

サポート対象のカードとプラットフォーム

OBFL データ収集はサポートされています。

OBFL データ ストレージに利用できる十分な不揮発性メモリが搭載されている FRU では OBFL がサポートされます。 たとえば、プロセッサでは OBFL がサポートされています。

表 *34*：カード タイプごとの *OBFL* サポート

カード タイプ	Cisco ASR 9000 Series Router
ルートスイッチ プロセッサ（RSP）	サポートあり
電源装置カード：AC 整流モジュールおよび DC 電源入力モジュール（PEM）	未サポート
ファン コントローラ カード	サポートあり
共有ポート アダプタ（SPA）	未サポート

Syslog メッセージの重大度の定義

デフォルトでは、アラート メッセージおよび緊急メッセージの OBFL データが収集されます。

OBFL の実装方法

OBFL ロギングはルータに設定されています。 新しいノードを挿入し、そのスロットまたはすべてのスロットで OBFL をイネーブルにすると、新しいノードで OBFL がイネーブルになります。ルータからカードを取り外して別のルータに挿入すると、そのカードは新しいルータで OBFL 設定を有効にします。

ここでは、次の手順について説明します。

OBFL のイネーブル化またはディセーブル化

OBFL はすべてのノードでデフォルトでイネーブルであり、指定したノードまたはすべてのノードでディセーブルにするまでアクティブです。

注意 OBFL データは FRU の問題の診断および解決に使用されるため、明確な理由なしに OBFL を非アクティブ化しないでください。

OBFL をイネーブルまたはディセーブルにする以外に設定要件はありません。

手順の概要

1. admin
2. configure
3. **hw-module** {**all** | **subslot** *node-id*} **logging onboard** [**disable** | **severity** {**alerts** | **emergencies**}]
4. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	admin 例： `RP/0/RSP0/CPU0:router# admin`	管理 EXEC モードを開始します。

	コマンドまたはアクション	目的

ステップ 2	configure
	
	例：
	`RP/0/RSP0/CPU0:router(admin)# configure`
	
	`RP/0/RSP0/CPU0:router(admin-config)#`
		管理コンフィギュレーション モードを開始します。

ステップ 3	**hw-module** {**all** | **subslot** *node-id*} **logging onboard** [**disable** | **severity** {**alerts** | **emergencies**}] 例： `RP/0/RSP0/CPU0:router(admin-config)# hw-module all logging onboard severity alerts`	OBFL ストレージ デバイスのログに記録される syslog メッセージの重大度を設定します。 • OBFL ストレージ デバイスのログに記録される syslog メッセージの重大度を指定するには、**severity** キーワードを使用します。 • 緊急 syslog メッセージとアラート syslog メッセージの両方をログに記録するように指定するには、**alerts** キーワードを使用します。 デフォルトは **alerts** キーワードです。 • 緊急 syslog メッセージだけをログに記録するように指定するには、**emergencies** キーワードを使用します。

ステップ 4	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。
		
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。

メッセージの重大度の設定

このタスクを実行して、メッセージの重大度を設定します。

手順の概要

1. admin
2. configure
3. **hw-module** {**all** | **subslot** *node-id*} **logging onboard** [**disable** | **severity** {**alerts** | **emergencies**}]
4. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	admin 例： `RP/0/RSP0/CPU0:router# admin`	管理 EXEC モードを開始します。

ステップ 2	configure
	
	例：
	`RP/0/RSP0/CPU0:router(admin)# configure`
	
	`RP/0/RSP0/CPU0:router(admin-config)#`
		管理コンフィギュレーション モードを開始します。

ステップ 3	**hw-module** {**all**	**subslot** *node-id*} **logging onboard** [**disable**	**severity** {**alerts**	**emergencies**}] 例： `RP/0/RSP0/CPU0:router(admin-config)# hw-module all logging onboard severity alerts`	OBFL ストレージ デバイスのログに記録される syslog メッセージの重大度を設定します。 • OBFL ストレージ デバイスのログに記録される syslog メッセージの重大度を指定するには、**severity** キーワードを使用します。 • 緊急 syslog メッセージとアラート syslog メッセージの両方をログに記録するように指定するには、**alerts** キーワードを使用します。 デフォルトは **alerts** キーワードです。 • 緊急 syslog メッセージだけをログに記録するように指定するには、**emergencies** キーワードを使用します。

	コマンドまたはアクション	目的

ステップ 4	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーションセッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーションファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。

OBFL のモニタリングおよびメンテナンス

OBFL のステータスと OBFL が収集したデータを表示するには、この項に説明されているコマンドを使用します。 これらのコマンドは EXEC または管理 EXEC モードで入力します。

手順の概要

1. admin
2. **show logging onboard** [**all** | *cbc* {**dump-all** | **dump-range** {*start-address* | *end-address* | **most-recent** {**fans** *fan-tray-slot* | [**location** *node-id*]} | **diagnostic** | **environment** | **error** | **temperature** | **uptime** | **verbose** | **voltage**] [**continuous** | **historical** | **static-data**] [**detail** | **raw** | **summary**] [**location** *node-id*]
3. **show processes include obfl**
4. **show running-config**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	admin 例： `RP/0/RSP0/CPU0:router# admin`	管理 EXEC モードを開始します。
ステップ 2	**show logging onboard** [**all** | *cbc* {**dump-all** | **dump-range** {*start-address* | *end-address* | **most-recent** {**fans** *fan-tray-slot* | [**location** *node-id*]} | **diagnostic** | **environment** | **error** | **temperature** | **uptime** | **verbose** | **voltage**] [**continuous** | **historical** | **static-data**] [**detail** | **raw** | **summary**] [**location** *node-id*] 例： `RP/0/RSP0/CPU0:router(admin)# show logging onboard uptime`	すべてのノードまたは指定したノードの保存されている OBFL データを表示します。 『*Cisco ASR 9000 Series Aggregation Services Router System Monitoring Command Reference*』の「*Onboard Failure Logging Commands*」モジュールを参照してください。
ステップ 3	**show processes include obfl** 例： `RP/0/RSP0/CPU0:router# show processes include obfl`	OBFL 環境モニタ プロセスが動作していることを確認します。
ステップ 4	**show running-config** 例： `RP/0/RSP0/CPU0:router# show running-config`	OBFL コンフィギュレーションのステータスを表示します。

OBFL データのクリア

特定のカードまたはすべてのカードの OBFL データをすべて消去するには、次のコマンドを使用します。

clear logging onboard [**all** | **cbc** {**dump-all** | **dump-range** {*start-address* | *end-address* | **most-recent** {**fans** *fan-tray-slot* | [**location** *node-id*]} | **corrupted-files** | **diagnostic** | **environment** | **error** | **poweron-time** | **temperature** | **uptime** | **voltage**] [**location** *node-id*]

⚠

注意 **clear logging onboard** コマンドは、1 つのノードまたはすべてのノードから、すべての OBFL データを完全に削除します。 OBFLデータはFRUの問題の診断および解決に使用されるため、明確な理由なしに OBFL ログをクリアしないでください。

注意 OBFL がカード上でアクティブに実行されている場合、**clear logging onboard** コマンドを発行すると、後で破損したログや不完全なログが生成されることがあります。OBFL は必ず、このコマンドを発行する前にディセーブルにしてください。

詳細については、『*Cisco ASR 9000 Series Aggregation Services Router System Monitoring Command Reference*』の「*Onboard Failure Logging Commands* 」モジュールを参照してください。

OBFL の設定例

ここでは、次の設定例について説明します。

OBFL のイネーブル化とディセーブル化：例

次の例は、OBFL をディセーブルにする方法を示しています。

```
RP/0/RSP0/CPU0:router(admin-config)# hw-module all logging onboard disable
```

次の例は、OBFL を再度イネーブルにする方法を示しています。

```
RP/0/RSP0/CPU0:router(admin-config)# no hw-module all logging onboard disable
```

次の例は、OBFL がイネーブルになっており、メッセージの重大度がデフォルトにリセットされていることを示しています。

```
RP/0/RSP0/CPU0:router(admin-config)# no hw-module all logging onboard
```

メッセージの重大度の設定：例

次の例は、重大度が 0（emergency）に設定されている syslog メッセージだけをストレージ デバイスに保存する方法を示しています。

```
RP/0/RSP0/CPU0:router(admin-config)# hw-module subslot 0/2/CPU0 logging onboard severity
emergencies
```

次の例は、重大度が 0（emergency）および 1（alert）に設定されている syslog メッセージをストレージ デバイスに保存する方法を示しています。

```
RP/0/RSP0/CPU0:router(admin-config)# hw-module subslot 0/2/CPU0 logging onboard severity
           alerts
```

OBFL メッセージのクリア：例

次の例では、システム内のすべてのノードの OBFL メッセージがすべてクリアされます。

```
RP/0/RSP0/CPU0:router(admin)# clear logging onboard
```

OBFL データの表示：例

次の例では、OBFL 機能からアップタイム情報を表示する方法を示します。

```
RP/0/RSP0/CPU0:router(admin)# show logging onboard uptime detail location 0/7/cpu0

--------------------------------------------------------------------------------
UPTIME CONTINUOUS DETAIL INFORMATION (Node: node0_7_CPU0)
--------------------------------------------------------------------------------
The first record     : 01/05/2009 00:58:41
The last record      : 01/17/2007916:07:13
Number of records    :        478
File size            :      15288 bytes
Current reset reason : 0x00
Current uptime       :    0 years  0 weeks 0 days  3 hours  0 minutes
--------------------------------------------------------------------------------
Time Stamp          |
MM/DD/YYYY HH:MM:SS | Users operation
--------------------------------------------------------------------------------
01/05/2009 01:44:35   File cleared by user request.
--------------------------------------------------------------------------------
```

次の作業

アラーム ログ相関を設定するには、『*Cisco ASR 9000* シリーズ アグリゲーション サービス ルータ システム モニタリング コンフィギュレーション ガイド』の「アラームおよびアラーム ログ相関の実装とモニタリング」モジュールを参照してください。

その他の関連資料

次の項では、Cisco IOS XR ソフトウェアへのロギング サービスの実装に関連する参考資料を紹介します。

関連資料

関連項目	マニュアル タイトル
ロギング サービス コマンド リファレンス	『*Cisco ASR 9000 Series Aggregation Services Router System Monitoring Command Reference*』の「*Logging Services Commands*」モジュール

関連項目	マニュアル タイトル
オンボード障害ロギング（OBFL）コンフィギュレーション	『*Cisco ASR 9000* シリーズ アグリゲーション サービス ルータ システム モニタリング コンフィギュレーション ガイド』の「*Onboard Failure Logging Commands*」モジュール
オンボード障害ロギング（OBFL）コマンド	『*Cisco ASR 9000 Series Aggregation Services Router System Monitoring Command Reference*』の「*Onboard Failure Logging Commands*」モジュール
アラームおよびロギング相関コマンド	『*Cisco ASR 9000 Series Aggregation Services Router System Monitoring Command Reference*』の「*Alarm Management and Logging Correlation Commands*」モジュール
アラームおよびロギング相関コンフィギュレーションおよびモニタリング タスク	『*Cisco ASR 9000* シリーズ アグリゲーション サービス ルータ システム モニタリング コンフィギュレーション ガイド』の「アラームおよびアラーム ログ相関の実装とモニタリング」モジュール
SNMP コマンド	『*Cisco ASR 9000 Series Aggregation Services Router System Monitoring Command Reference*』の「*SNMP Commands*」モジュール
SNMP の設定作業	『*Cisco ASR 9000* シリーズ アグリゲーション サービス ルータ システム モニタリング コンフィギュレーション ガイド』の「*Implementing SNMP*」モジュール
Cisco IOS XR のスタートアップ マニュアル	『*Cisco ASR 9000 Series Aggregation Services Router Getting Started Guide*』
ユーザ グループとタスク ID に関する情報	『*Cisco ASR 9000 Series Aggregation Services Router System Security Command Reference*』の「*Configuring AAA Services*」モジュール

標準

標準	タイトル
この機能でサポートされる新規の標準または変更された標準はありません。また、既存の標準のサポートは変更されていません。	—

MIB

MIB	MIB のリンク
—	Cisco IOS XR ソフトウェアを使用して MIB を検索およびダウンロードするには、http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml にある Cisco MIB Locator を使用し、[Cisco Access Products] メニューからプラットフォームを選択します。

RFC

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

シスコのテクニカル サポート

説明	リンク
シスコのテクニカル サポート Web サイトでは、製品、テクノロジー、ソリューション、技術的なヒント、およびツールへのリンクなどの、数千ページに及ぶ技術情報が検索可能です。Cisco.com に登録済みのユーザは、このページから詳細情報にアクセスできます。	*http://www.cisco.com/cisco/web/support/index.html*

第 7 章

パフォーマンス管理の実装

Cisco IOS XR ソフトウェアでのパフォーマンス管理（PM）には、次のタスクを実行するためのフレームワークが用意されています。

- 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。

（注） Cisco IOS XR ソフトウェアでの PM およびこのモジュールの一覧で示されている PM コマンドの詳細については、このモジュールの関連資料，（448 ページ）の項を参照してください。

パフォーマンス管理の実装の機能履歴

リリース	変更内容
リリース 3.7.2	この機能が導入されました。

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。

  オプションの PIE インストールの詳細については、『*Cisco ASR 9000 Series Aggregation Services Router Getting Started Guide*』を参照してください。

- 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。

Cisco IOS XR ソフトウェアのパフォーマンス管理の実装に関する情報

パフォーマンス管理を実装するには、次の概念を理解する必要があります。

PM 機能の概要

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。

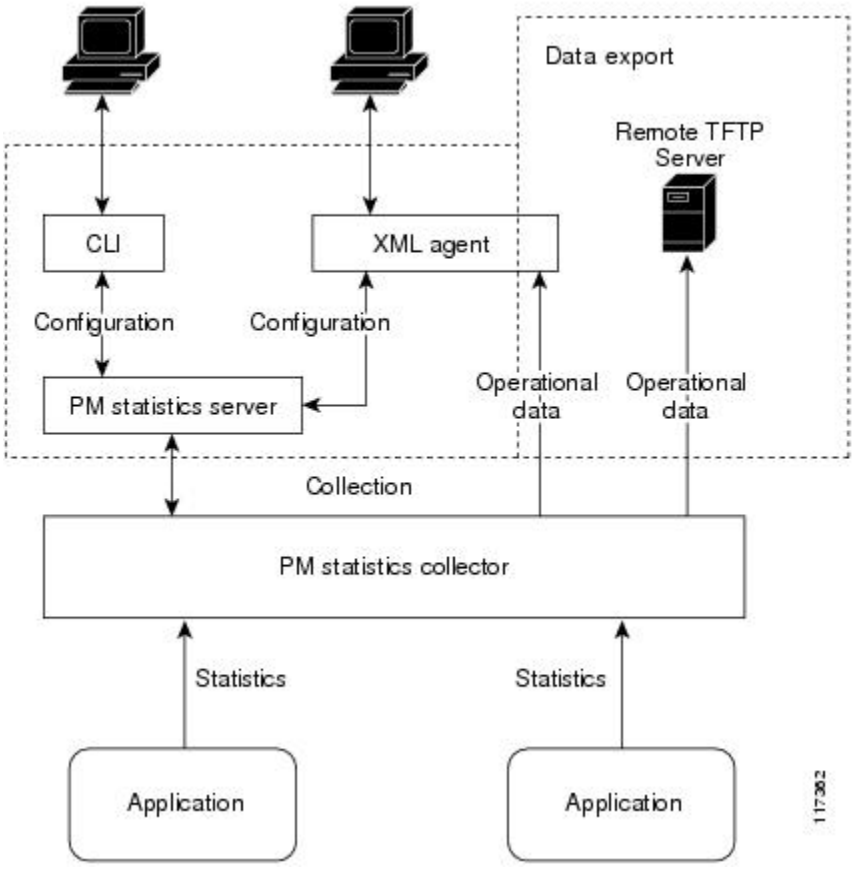
PM 統計情報収集機能

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 7：PM コンポーネントの通信, （407 ページ）は、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 *7*：*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 *35*：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 データ レート	Interfaces
インターフェイス汎用カウンタ	Interfaces
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 43：属性と値，（423 ページ）を参照してください。

PM 統計情報収集テンプレート

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。

PM 統計情報収集テンプレートを作成するガイドライン

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。

- PM 統計情報収集テンプレートを作成するには、**performance-mgmt statistics** コマンドを使用します。
- 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。
- テンプレートの設定時には、テンプレートの名前を付ける必要があります。 **default** 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設定できます。または、**template** キーワードと *template-name* 引数を指定して、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、次のデフォルト設定が含まれています。
  - 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。
- サンプル間隔およびサンプル サイズをテンプレートに設定します。
  - 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** キーワードおよび *minutes* 引数を指定して、サンプル間隔を設定できます。 範囲は 1 ～ 60 分です。 デフォルトは 10 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** キーワードおよび *minutes* 引数を指定して、サンプル サイズを設定できます。 範囲は 1 ～ 60 サンプルです。 デフォルトは 5 サンプルです。
- 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）。 たとえば、サンプル間隔の頻度が 10 分に設定されており、サンプル サイズが 5 回のサンプリング動作に設定されているとします。 この場合、5 回のサンプリング動作の合計は、10 分ごとに 1 回のサンプリング動作の頻度で実行されます。 このサイクルはサンプリング サイクルと呼ばれます。 これらのサンプルから収集されるデータが含まれるバイナリ ファイルは、50（5 x 10）分ごとに TFTP サーバにエクスポートされます。 このサイクルはエクスポート サイクルと呼ばれます。

注意 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。

PM 統計情報収集テンプレートをイネーブル化およびディセーブル化するガイドライン

PM 統計情報テンプレートのイネーブル時は、次のガイドラインに従ってください。

- PM 統計情報収集テンプレートをイネーブルするには、**performance-mgmt apply statistics** コマンドを使用します。
- 特定のエンティティで一度にイネーブルにできるのは 1 つの PM 統計情報収集テンプレートだけです。

（注） **performance-mgmt enable statistics** コマンドを使って PM 統計情報収集テンプレートをイネーブルにすると、データ収集は 1 つサンプリング サイクルを開始します。

- 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 統計情報収集をイネーブルまたはディセーブルにするときには、**location** キーワードと *node-id* 引数を使って場所を指定するか、**location all** キーワードを指定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

  location キーワードと *node-id* 引数は、指定したノードの PM 統計情報収集をイネーブルにします。 *node-id* 引数は *rack/slot/module* の形式で入力します。 **location all** キーワードでは、すべてのノードの PM 統計情報収集をイネーブルにします。
- 特定の期間の特定のエンティティに関する PM 統計情報の収集は 1 件だけイネーブルにできるため、PM 統計情報の収集をディセーブルにする際は、**default** キーワードまたは **template** キーワードと *template-name* 引数でテンプレート名を指定する必要はありません。

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 *36*：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 37：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，（412 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>	
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>	
OSPF（5）	v2protocol（1）	インスタンス <process_instance>	

エンティティ	サブエンティティ	キー	統計情報カウンタ
	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 36：バイナリ形式の値とキー，（411 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 *37*：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

エンティティ	サブエンティティ	統計情報カウンタ
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
	データ レート カウンタ（2）	InputDataRate、InputPacketRate、OutputDataRate、OutputPacketRate、InputPeakRate、InputPeakPkts、OutputPeakRate、OutputPeakPkts、Bandwidth
BGP（3）	ピア（1）	InputMessages、OutputMessages、InputUpdateMessages、OutputUpdateMessages、ConnEstablished、ConnDropped、ErrorsReceived、ErrorsSent

エンティティ	サブエンティティ	統計情報カウンタ
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

エンティティ	サブエンティティ	統計情報カウンタ
	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>

PM エンティティ インスタンス モニタリングの概要

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。

（注） 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 XML インターフェイスからのみ取得できます。

この表では、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 *38* : *BGP* エンティティ インスタンス モニタリング

エンティティ	コマンドについて
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イネーブルするには、グローバルコンフィギュレーション モードで **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/RSP0/CPU0:routerrouter(config)# performance-mgmt apply monitor bgp 10.12.0.4 default`

この表では、インターフェイス 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 *39* : インターフェイス エンティティ インスタンス モニタリング

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 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 モニタリングをイネーブルするには、グローバル コンフィギュレーション モードで **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/RSP0/CPU0:router(config)# performance-mgmt apply monitor interface data-rates gigabitethernet 0/2/0/0 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 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 モニタリングをイネーブルするには、グローバル コンフィギュレーション モードで **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/RSP0/CPU0:router(config)# performance-mgmt apply monitor interface generic-counters gigabitethernet 0/2/0/0 default`

この表では、MPLS 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 *40*：*MPLS* エンティティ インスタンス モニタリング

エンティティ	コマンドの説明
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 モニタリングをイネーブルするには、グローバル コンフィギュレーション モードで **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/RSP0/CPU0:router(config)# performance-mgmt apply monitor mpls ldp 10.34.64.154 default`

この表では、ノード 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 *41* : ノード エンティティ インスタンス モニタリング

エンティティ	コマンドの説明
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 モニタリングをイネーブルするには、グローバル コンフィギュレーション モードで **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/RSP0/CPU0:router(config)# performance-mgmt apply monitor node cpu location 0/RP1/CPU0 default`
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 モニタリングをイネーブルするには、グローバル コンフィギュレーション モードで **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/RSP0/CPU0:router(config)# performance-mgmt apply monitor node memory location 0/RP1/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 モニタリングをイネーブルするには、グローバル コンフィギュレーション モードで **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/RSP0/CPU0:router(config)# performance-mgmt apply monitor node process location p 0/RP1/CPU0 275 default`

PM しきい値モニタリングの概要

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。

PM しきい値モニタリング テンプレートを作成するガイドライン

PM しきい値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。

- PM しきい値テンプレートを作成するには、 **performance-mgmt thresholds** コマンドを使用します。
- エンティティを *entity* 引数で指定します。
- 1 つのエンティティに複数の PM しきい値テンプレートを定義できます。だたし、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 つだけであることに注意してください。
- エンティティのテンプレートを設定するときにその名前を指定します。 **default** キーワードを使用して、テンプレートをデフォルトのテンプレートとして設定できます。または、 **template** キーワードと *template-name* 引数を指定して、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 テンプレートのデフォルト設定は 10 分のサンプル間隔です。
- しきい値違反をモニタリングするエンティティに関連付けられる属性を *attribute* 引数で指定します。

（注） 各エンティティに関連付けられる属性のリストについては、表 43：属性と値, （423 ページ）を参照してください。

- **sample-interval** キーワードおよび *interval* 引数を指定して、PM しきい値モニタリングのサンプル間隔を設定します。 サンプル間隔は、属性のインスタンスがしきい値条件に一致するかどうかを判断するまでに PM システムが待機する頻度（分単位）を設定します。
- モニタリングする属性のしきい値条件を指定します。 しきい値条件は、属性、動作、しきい値から構成されます。 しきい値条件は属性のすべてのインスタンスに適用されます。

（注） PM しきい値テンプレートには、複数のしきい値条件を含めることができます。モニタする各しきい値条件を定義し、 **performance-mgmt thresholds** コマンドを使用して、指定したテンプレートにそのしきい値条件を適用する必要があります。

- しきい値条件で実行される動作を指定します。 サポートされる動作は次のとおりです。
  - **EQ**：等しい
  - **GE**：より大きいまたは等しい
  - **GT**：より大きい
  - **LE**：より小さいまたは等しい
  - **LT**：より小さい
  - **NE**：等しくない
  - **RG**：範囲外
- 値を *value* 引数で指定します。 *value* 引数を指定すると、PM システムはしきい値条件を絶対値として認識し、各サンプル間隔の後に属性のインスタンスのいずれかがしきい値条件と一致するかどうかを判断します。 **RG** キーワードを指定して範囲外の動作を指定する場合、範囲を指定する値のペアを指定する必要があります。
- 任意の **percent** キーワードを指定する場合、 *value* 引数は 0 から 100 のパーセンテージとして表す必要があります。 *value* 引数と **percent** キーワードを指定して値をパーセンテージとして表すと、しきい値条件は現在および以前のサンプル間で属性のそれぞれのインスタンス値の差分をパーセンテージで比較します。
- 任意で **rearm toggle** キーワードや **rearm window** キーワードおよび *window-size* 引数を指定できます。
  - **rearm toggle** ：属性のインスタンスがしきい値条件と一致したときに、属性のインスタンスのシステム ロギング メッセージを抑制します。 連続するサンプル間隔では、属性のインスタンスがしきい値条件に一致しないようになるまで、属性のそのインスタンスのシステム ロギング メッセージは抑制されます。
  - **rearm window** *window-size* ：属性のインスタンスがしきい値条件に一致する場合、*window-size* 引数に指定された間隔で、システム ロギング メッセージを抑制します。

（注） しきい値条件が満たされているかどうかを PM システムで決定する方法の詳細については、表 42：しきい値条件が満たされているかどうかを PM システムで決定する方法, （420 ページ）を参照してください。

この表では、しきい値条件が一致するかどうかを PM システムが決定する方法を説明します。

表 *42*：しきい値条件が満たされているかどうかを *PM* システムで決定する方法

しきい値条件が次の項目で構成されている場合	結果
属性、動作、特定の値	しきい値条件は絶対値です。これは、各サンプル間隔の経過後に、PM システムは属性のインスタンスがしきい値条件と完全に一致するかどうかを決定するためです。 • たとえば、エンティティのしきい値条件で、インスタンスの属性が 2000 よりも大きいかどうかをチェックするように設定されているとします。 サンプル間隔が経過したら、それに応じてPMシステムでは属性のインスタンスが条件に一致するかどうかを決定します。 • PMシステムでは、サンプル間隔が経過した後にしきい値条件に一致する属性の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。 • 属性のインスタンスがしきい値条件に一致しない場合、システム ロギング メッセージはそのサンプル間隔では生成されません。

しきい値条件が次の項目で構成されている場合	結果
属性、動作、パーセンテージで表される値	しきい値条件は相対値です。これは、比較に使用されるしきい値が、以前のサンプルのパーセンテージとして使用されるためです。 • たとえば、エンティティのしきい値条件で、以前のサンプルのしきい値よりもインスタンスの属性が 51 パーセント以上増加するかどうかをチェックするように設定されているとします。 サンプル間隔が経過した後、属性のインスタンスの値が 250 であるとします。 属性の任意のインスタンスが以前のしきい値よりも 51 パーセント以上大きい場合にシステム ロギング メッセージを生成するようにしきい値条件が設定されているため、PM システムでは次のサンプル間隔で、属性のその特定のインスタンスが 375（250 + 125（250 の 50 %））よりも大きいかどうかをチェックします。 （注） PM システムでは属性のすべてのインスタンスに対してしきい値条件を一致させます。そのため、しきい値条件のこのタイプのしきい値は、属性のインスタンスの値に対して相対的です。 • PM システムでは、サンプル間隔が経過した後にしきい値条件に一致する属性の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。 • 属性のインスタンスがしきい値条件に一致しない場合、システム ロギング メッセージはそのサンプル間隔では生成されません。

しきい値条件が次の項目で構成されている場合	結果
属性、動作、特定の値、 **rearm toggle** キーワード	サンプル間隔の経過後に、属性のインスタンスがしきい値条件と一致する場合、属性のそのインスタンスにシステム ロギング メッセージが生成されるように変更されます。 ただし、最初の一致に続く連続するサンプル間隔で、その属性の同じインスタンスがしきい値条件に一致する場合、インスタンスがサンプル間隔のしきい値条件に一致しなくなるまで、属性のそのインスタンスのシステム ロギング メッセージは表示されません。
属性、動作、特定の値、**rearm window** キーワードおよび *window-size* 引数	属性のインスタンスがしきい値条件と一致する場合、システム ロギング メッセージが生成されるようにしきい値条件が変更されます。ただし、属性のインスタンスがしきい値条件に一致すると、属性のそのインスタンスのシステム ロギング メッセージは *window-size* 引数で指定した間隔の回数だけ抑制されます。

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 *43*：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。
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。

エンティティ	属性	説明	値
	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		

エンティティ	属性	説明	値
		送信されたマルチキャスト パケット。	範囲は 0 ～ 4294967295 です。
	OutOctets	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。
	AddressWithdrawMsgsRcd	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。

エンティティ	属性	説明	値
	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。
	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。

エンティティ	属性	説明	値
	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。
	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。
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。

エンティティ	属性	説明	値
	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。
	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。

エンティティ	属性	説明	値
	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。

PM しきい値モニタリング テンプレートをイネーブル化およびディセーブル化するガイドライン

PM しきい値モニタリング テンプレートをイネーブルにするときには、次のガイドラインに従います。

- PM しきい値モニタリング テンプレートをイネーブルにするには、**performance-mgmt apply thresholds** コマンドを使用します。
- テンプレートをイネーブルにすると、**performance-mgmt apply thresholds** 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、しきい値モニタリングは継続されます。
- 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。
- 次のエンティティの PM しきい値モニタリング テンプレートをイネーブルまたはディセーブルにするときには、**location** キーワードと *node-id* 引数を使って場所を指定するか、**location all** キーワードを指定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

  location キーワードと *node-id* 引数では、指定したノードの PM 統計情報収集をイネーブルまたはディセーブルにします。 *node-id* 引数は *rack/slot/module* の形式で入力します。 **location all** キーワードでは、すべてのノードの PM 統計情報収集をイネーブルまたはディセーブルにします。
- 1 つのエンティティに付き一度にイネーブルにできる PM しきい値のモニタリングテンプレートは 1 つだけであるため、PM 統計情報収集をディセーブルにするとき、**default** キーワードや **template** キーワードおよび *template-name* 引数でテンプレート名を指定する必要はありません。

パフォーマンス管理の実装方法

ここでは、次の手順について説明します。

PM 統計情報収集の外部 TFTP サーバの設定

このタスクでは、PM 統計情報収集の外部 TFTP サーバを設定する方法を説明します。

（注） PM 統計情報収集の PM 統計情報収集テンプレートをイネーブルにする前にこのタスクを実行します。 PM 統計情報収集テンプレートをイネーブルにする方法の詳細については、PM 統計情報収集テンプレートのイネーブル化とディセーブル化, （438ページ）を参照してください。

はじめる前に

このタスクを実行する前に、TFTP サーバにアクセスして、接続できる必要があります。

手順の概要

1. **configure**
2. **performance-mgmt resources tftp-server** *ip-address* **directory** *dir-name*
3. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**performance-mgmt resources tftp-server** *ip-address* **directory** *dir-name* 例： `RP/0/RSP0/CPU0:routerRP/0/RSP0/CPU0:router(config)# performance-mgmt resources tftp-server 10.3.40.161 directory mypmdata/datafiles`	PM データ収集用の IP アドレスおよびディレクトリ パスを設定します。 • *dir-name* 引数にはディレクトリ パス名全体を含めます。 （注） TFTP サーバにコピーされるファイルは、ファイル名にタイムスタンプが含まれ、それによってファイル名が一意になります。このため、ユーザが事前に TFTP サーバ ホストでファイルを手動作成する必要がないよう、使用する TFTP サーバはデータ転送時のファイルの作成をサポートしている必要があります。

	コマンドまたはアクション	目的

ステップ 3	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

PM 統計情報収集テンプレートの作成

このタスクでは、PM 統計情報収集テンプレートを作成する方法を説明します。

手順の概要

1. **configure**
2. **performance-mgmt statistics** *entity* {**default** | **template** *template-name*} [**sample-size** *size*] [**sample-interval** *minutes*]
3. 次のいずれかのコマンドを使用します。
   • **end**
   • **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**performance-mgmt statistics** *entity* {**default** | **template** *template-name*} [**sample-size** *size*] [**sample-interval** *minutes*] 例： `RP/0/RSP0/CPU0:router(config)# performance-mgmt statistics interface data-rates default`	指定したエンティティの PM 統計情報収集テンプレートを作成します。 • PM 統計情報収集テンプレートを作成するエンティティを指定するには、*entity* 引数を使用します。 • 指定したエンティティの PM 統計情報テンプレートにデフォルトのテンプレートを適用するには、**default** キーワードを使用します。デフォルト テンプレートには、10 分のデフォルト サンプル間隔および 5 つのサンプリング動作のデフォルト サンプル サイズが含まれています。 • テンプレートに固有名を指定するには、**template** キーワードと *template-name* 引数を使用します。 • **sample-size** キーワードと *size* 引数は、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。範囲は 1 ～ 60 サンプルです。デフォルト値は 5 サンプルです。 • **sample-interval** キーワードと *minutes* 引数では、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。範囲は1～60 分です。デフォルトは 10 分です。 （注） PM統計情報収集の作成の詳細は、「PM統計情報収集テンプレートを作成するガイドライン，（409 ページ）」の項を参照してください。

ステップ 3	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。

	コマンドまたはアクション	目的
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。 • 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

次の作業

PM 統計情報収集テンプレートを作成したら、テンプレートをイネーブルにして、PM 統計情報収集を開始する必要があります。 PM統計情報収集テンプレートのイネーブル化については、PM統計情報収集テンプレートのイネーブル化とディセーブル化, （438 ページ）の項を参照してください。

PM 統計情報収集テンプレートのイネーブル化とディセーブル化

このタスクでは、PM統計情報収集テンプレートをイネーブルおよびディセーブルにする方法を説明します。

はじめる前に

このタスクを実行する前に PM 統計情報収集テンプレートを作成する必要があります。または、事前に定義されたテンプレート（デフォルト）を使用できます。 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。

詳細については、PM 統計情報収集の外部 TFTP サーバの設定, （435 ページ）およびPM 統計情報収集テンプレートの作成, （436 ページ）のタスクを参照してください。

手順の概要

1. **configure**
2. 次のいずれかを実行します。
   - **performance-mgmt apply statistics** {*entity* | **interface** { **data-rates** | **generic-counters**} *type interface-path-id* } [**location** {**all** | *node-id*}] {*template-name* | **default**}
   - **no performance-mgmt apply statistics** {*entity* | **interface** { **data-rates** | **generic-counters**} *type interface-path-id* } [**location** {**all** | *node-id*}]
3. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	次のいずれかを実行します。 • **performance-mgmt apply statistics** {*entity* | **interface** { **data-rates** | **generic-counters**} *type interface-path-id* } [**location** {**all** | *node-id*}] {*template-name* | **default**} • **no performance-mgmt apply statistics** {*entity* | **interface** { **data-rates** | **generic-counters**} *type interface-path-id* } [**location** {**all** | *node-id*}] 例： `RP/0/RSP0/CPU0:router(config)# performance-mgmt apply statistics mpls ldp default`	PM統計情報収集テンプレートをイネーブルまたはディセーブルにします。 • 特定のエンティティで一度にイネーブルにできるのは 1 つの PM 統計情報収集テンプレートだけです。 • 次のエンティティの PM 統計情報収集をイネーブルにするときには、**location** キーワードと *node-id* 引数を使って場所を指定するか、**location all** キーワードを指定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 **location** キーワードと *node-id* 引数は、指定したノードの PM 統計情報収集をイネーブルにします。 *node-id* 引数は *rack*/*slot*/*module* の形式で入力します。 **location all** は、すべてのノードの PM 統計情報収集をイネーブルにします。 • 特定の期間の特定のエンティティに関する PM 統計情報の収集は 1 件だけイネーブルにできるため、PM 統計情報の収集をディセーブルに

	コマンドまたはアクション	目的
	または `RP/0/RSP0/CPU0:router(config)# no performance-mgmt apply statistics mpls ldp`	する際は、**default** キーワードまたは **template** キーワードと *template-name* 引数でテンプレート名を指定する必要はありません。 （注） **performance-mgmt apply statistics** コマンドを使って PM 統計情報収集テンプレートをイネーブルにすると、データ収集は 1 つサンプリング サイクルを開始します。 • 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 統計情報収集をディセーブルにするときには、**location** キーワードと *node-id* 引数を使って場所を指定するか、**location all** キーワードを指定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 **location** キーワードと *node-id* 引数では、指定したノードの PM 統計情報収集をディセーブルにします。 *node-id* 引数は *rack*/*slot*/*module* の形式で入力します。 **location all** キーワードでは、すべてのノードの PM 統計情報収集をディセーブルにします。 • 特定の期間の特定のエンティティに関する PM 統計情報の収集は 1 件だけイネーブルにできるため、PM 統計情報の収集をディセーブルにする際は、**default** キーワードまたは **template** キーワードと *template-name* 引数でテンプレート名を指定する必要はありません。

ステップ 3	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。

	コマンドまたはアクション	目的
		◦cancel と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。 • 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

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

このタスクでは、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方法を説明します。

はじめる前に

このタスクを実行する前に、エンティティの PM 統計情報収集テンプレートを作成する必要があります。

手順の概要

1. **configure**
2. **performance-mgmt apply monitor** {*entity instance* | **interface** { **data-rates** | **generic-counters**} *type interface-path-id* } {*template-name* | **default**}
3. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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。

	コマンドまたはアクション	目的
ステップ 2	**performance-mgmt apply monitor** {*entity instance* | **interface** { **data-rates** | **generic-counters**} *type interface-path-id* } {*template-name* | **default**} 例： `RP/0/RSP0/CPU0:router(config)# performance-mgmt apply monitor node cpu 0/RSP1/CPU0 default`	指定したインスタンスのエンティティ インスタンス モニタリングをイネーブルにします。 • モニタするエンティティとインスタンスの名前を指定するには、*entity* 引数および *instance* 引数をそれぞれに使用します。 • モニタされるエンティティ インスタンスに関連付けられるテンプレートを指定するには、**default** キーワードまたは *template-name* 引数のいずれかを使用します。

ステップ 3	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 設定変更を実行コンフィギュレーション ファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

PM しきい値モニタリング テンプレートの作成

このタスクでは、PM しきい値モニタリング テンプレートを作成する方法を説明します。

手順の概要

1. **configure**
2. **performance-mgmt thresholds** {*entity* | **interface** {**data-rates** | **generic-counters**} *type interface-path-id* } {**template** *name* } *attribute operation value* [*value2*] [**percent**] [**rearm** {**toggle** | **window** *window-size*}]
3. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	**performance-mgmt thresholds** {*entity* | **interface** {**data-rates** | **generic-counters**} *type interface-path-id* } {**template** *name* } *attribute operation value* [*value2*] [**percent**] [**rearm** {**toggle** | **window** *window-size*}]
	例：
	`RP/0/RSP0/CPU0:router(config)# performance-mgmt thresholds node cpu template cpu_thresh1`
	`RP/0/RSP0/CPU0:router(config-threshold-bgp)# AverageCPUUsed GT 25 percent`
		PM しきい値モニタリング テンプレートを作成します。
		（注） PM しきい値モニタリングテンプレートの作成の詳細については、PM しきい値モニタリングテンプレートを作成するガイドライン，（419ページ）の項を参照してください。

ステップ 3	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。

	コマンドまたはアクション	目的
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータがEXECモードに戻ります。変更はコミットされません。 ◦ cancel と入力すると、現在のコンフィギュレーション セッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。 • 設定変更を実行コンフィギュレーションファイルに保存し、コンフィギュレーションセッションを継続するには、**commit** コマンドを使用します。

次の作業

PM しきい値モニタリングテンプレートを作成したら、テンプレートをイネーブルにして、PM しきい値モニタリングを開始する必要があります。 PM 統計情報しきい値モニタリング テンプレートのイネーブル化については、PM しきい値モニタリングテンプレートのイネーブル化とディセーブル化, （444 ページ）のタスクを参照してください。

PM しきい値モニタリング テンプレートのイネーブル化とディセーブル化

このタスクでは、PM しきい値モニタリング テンプレートをイネーブルおよびディセーブルにする方法を説明します。

はじめる前に

このタスクを実行する前に PM しきい値テンプレートを作成する必要があります。 詳細については、PM しきい値モニタリングテンプレートの作成, （442 ページ）のタスクを参照してください。

手順の概要

1. **configure**
2. 次のいずれかを実行します。
   - **performance-mgmt apply thresholds** {*entity* | **interface** {**data-rates** | **generic-counters**} *type interface-path-id* } [**location** {**all** | *node-id*}] {*template-name* | **default**}
   - **no performance-mgmt apply thresholds** {*entity*| **interface** { **data-rates** | **generic-counters**} *type interface-path-id* } [**location** {**all** | *node-id*}]
3. 次のいずれかのコマンドを使用します。
   - **end**
   - **commit**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	**configure** 例： `RP/0/RSP0/CPU0:router# 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 2	次のいずれかを実行します。 • **performance-mgmt apply thresholds** {*entity* | **interface** {**data-rates** | **generic-counters**} *type interface-path-id* } [**location** {**all** | *node-id*}] {*template-name* | **default**} • **no performance-mgmt apply thresholds** {*entity*| **interface** { **data-rates** | **generic-counters**} *type interface-path-id* } [**location** {**all** | *node-id*}] 例： `RP/0/RSP0/CPU0:router(config)# performance-mgmt enable thresholds node cpu location all template20`	指定したテンプレートの PM しきい値モニタリング テンプレートをイネーブルまたはディセーブルにします。 • 1 つのエンティティで一度にイネーブルにできるのは、1 つの PM しきい値モニタリング テンプレートだけです。 • 次のエンティティの PM 情報収集しきい値のモニタリング テンプレートをイネーブルにするときには、**location** キーワードと *node-id* 引数を使って場所を指定するか、**locationall** キーワードを指定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 **location** キーワードと *node-id* 引数は、指定したノードの PM しきい値モニタリング テンプレートをイネーブルにします。 *node-id* 引数は *rack/slot/module* の形式で入力します。 **location all** キーワードは、すべてのノードの PM しきい値モニタリング テンプレートをイネーブルにします。

	コマンドまたはアクション	目的
	または `RP/0/RSP0/CPU0:router(config)# no performance-mgmt apply thresholds node cpu location all`	• 1 つのエンティティに付き一度にイネーブルにできる PM しきい値のモニタリング テンプレートは 1 つだけであるため、PM 統計情報収集をディセーブルにするとき、**default** キーワードや **template** キーワードおよび *template-name* 引数でテンプレート名を指定する必要はありません。 • テンプレートをイネーブルにすると、**performance-mgmt apply thresholds** 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、しきい値モニタリングは継続されます。 • 次のエンティティの PM 情報収集しきい値のモニタリング テンプレートをディセーブルにするときには、**location** キーワードと *node-id* 引数を使って場所を指定するか、**location all** キーワードを指定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 **location** キーワードと *node-id* 引数は、指定したノードの PM しきい値モニタリング テンプレートをディセーブルにします。 *node-id* 引数は *rack*/*slot*/*module* の形式で入力します。 **location all** キーワードは、すべてのノードの PM しきい値モニタリング テンプレートをディセーブルにします。 • 1 つのエンティティに付き一度にイネーブルにできる PM しきい値のモニタリング テンプレートは 1 つだけであるため、PM 統計情報収集をディセーブルにするとき、**default** キーワードや *template-name* 引数でテンプレート名を指定する必要はありません。

ステップ 3	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	
	例：
	`RP/0/RSP0/CPU0:router(config)# end`
	または
	`RP/0/RSP0/CPU0:router(config)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、変更をコミットするように要求されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		◦ yes と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		◦ no と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。

	コマンドまたはアクション	目的
		◦cancelと入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーションセッションは終了せず、設定変更もコミットされません。 • 設定変更を実行コンフィギュレーションファイルに保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

パフォーマンス管理を実装するための設定例

ここでは、次の設定例について説明します。

PM 統計情報収集テンプレートの作成およびイネーブル化：例

次の例では、TFTPサーバリソースを設定する方法と、PM統計情報収集テンプレートをイネーブルにする方法を示します。 この例では、次のPMテンプレート収集テンプレートが作成され、イネーブルになります。

- 汎用インターフェイス カウンタ エンティティに10というサンプル サイズおよび5というサンプル間隔が設定された「template1」という名前のテンプレート。
- ノード メモリ エンティティに30というサンプル サイズおよび2というサンプル間隔が設定された「template2」という名前のテンプレート。 テンプレートはグローバルでイネーブルになります。
- ノード プロセス エンティティに10というサンプル サイズおよび5というサンプル間隔が設定された「template3」という名前のテンプレート。 テンプレートはノード0/0/CPU0でイネーブルになります。

```
performance-mgmt resources tftp-server 10.30.62.154 directory pm/pm_data/pmtest
performance-mgmt statistics interface generic-counters template template1
 sample-size 10
 sample-interval 5
!
performance-mgmt statistics node memory template template2
 sample-size 30
 sample-interval 2
!
performance-mgmt statistics node process template template3
 sample-size 10
 sample-interval 5
!
performance-mgmt apply statistics interface generic-counters template1
performance-mgmt apply statistics node memory global template2
```

```
performance-mgmt apply statistics node process 0/0/CPU0 template3
```

PM しきい値モニタリング テンプレートの作成およびイネーブル化：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 75 % より大きいかどうかを決定します。 テンプレートのサンプル間隔は 5 分に設定されています。テンプレートはグローバルにイネーブルです。

```
performance-mgmt thresholds node cpu template template20
 AverageCpuUsed GT 75
 sample-interval 5
!
performance-mgmt apply thresholds node cpu global template20
```

その他の関連資料

ここでは、パフォーマンス管理の実装に関する参考資料について説明します。

関連資料

関連項目	マニュアル タイトル
パフォーマンス管理コマンド	『*Cisco ASR 9000 Series Aggregation Services Router System Monitoring Command Reference*』の「Cisco ASR 9000 シリーズ ルータ」モジュールの「Performance Management Commands」
Cisco IOS XR ソフトウェアの XML API 参考資料	『*Cisco IOS XR XML API Guide*』
Cisco IOS XR ソフトウェアのスタートアップ参考資料	『*Cisco ASR 9000 Series Aggregation Services Router Getting Started Guide*』
ユーザ グループとタスク ID に関する情報	『*Cisco ASR 9000 Series Aggregation Services Router System Security Configuration Guide*』の「Cisco ASR 9000 シリーズ ルータ」モジュールの「Configuring AAA Services」

標準

標準	タイトル
この機能でサポートされる新規の標準または変更された標準はありません。また、既存の標準のサポートは変更されていません。	—

MIB

MIB	MIB のリンク
—	Cisco IOS XR ソフトウェアを使用して MIB を検索およびダウンロードするには、http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml にある Cisco MIB Locator を使用し、[Cisco Access Products] メニューからプラットフォームを選択します。

RFC

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

シスコのテクニカル サポート

説明	リンク
シスコのテクニカル サポート Web サイトでは、製品、テクノロジー、ソリューション、技術的なヒント、およびツールへのリンクなどの、数千ページに及ぶ技術情報が検索可能です。Cisco.com に登録済みのユーザは、このページから詳細情報にアクセスできます。	*http://www.cisco.com/cisco/web/support/index.html*

索 引

A

B

C

D

E

F

I

J

L

M

O

P

R

S

T

U

V

X

あ

い

え

お

か

き

け

こ

さ

し

す

せ

そ

た

て

と

の

は

ひ

ふ

へ

ほ

め

も

ゆ

よ

り

る

れ

ろ