



Cisco NCS 560 シリーズ ルータ (Cisco IOS XR リリース 7.0.x) システム モニタリング コンフィギュレーション ガイド

初版 : 2019 年 8 月 30 日

シスコシステムズ合同会社

〒107-6227 東京都港区赤坂9-7-1 ミッドタウン・タワー

<http://www.cisco.com/jp>

お問い合わせ先 : シスコ コンタクトセンター

0120-092-255 (フリーコール、携帯・PHS含む)

電話受付時間 : 平日 10:00~12:00、13:00~17:00

<http://www.cisco.com/jp/go/contactcenter/>

【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。

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: www.cisco.com/go/trademarks. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2019 Cisco Systems, Inc. All rights reserved.



目次

第 1 章

システム ログिंगの実装 1

システム ログिंगの実装 1

システム ログिंगの設定に関する前提条件 3

システム ログ機能の設定 3

ログング バッファへのログングの設定 3

リモート サーバへのログングの設定 3

端末回線へのログングの設定 5

コンソール端末へのログングの変更 5

タイム スタンプ形式の変更 5

重複 syslog メッセージの抑制 6

ローカルストレージデバイスへのシステム ログング メッセージのアーカイブ 6

プラットフォーム自動モニタリング 7

PAM イベント 8

PAM の無効化と再有効化 10

PAM でのデータ アーカイブ 11

PAM ツールによって収集されるファイル 12

第 2 章

アラーム ログ関連の実装 15

アラーム ログ関連の実装 15

アラーム ログ関連の実装に関する前提条件 15

アラーム ログ関連の実装に関する情報 15

アラーム ログングおよびデバッグ イベント管理システム 15

アラーム ログ関連の設定 17

ログング関連ルールの設定 17

ロギング関連ルール セットの設定	18
根本原因アラームと非根本原因アラームの関連	18
階層的な関連ルール フラグの設定	19
ロギング抑制ルールの設定	19
ロギング イベント バッファ設定の変更	20
ロギング関連バッファ設定の変更	20
バイステートアラームのアラーム ソース ロケーション表示フィールドのイネーブル化	20
SNMP 関連ルールの設定	21
SNMP 関連ルールセットの設定	21
アラーム ロギング関連の詳細	21

第 3 章

パフォーマンス管理の実装 27

パフォーマンス管理を実装する前提条件	27
パフォーマンス管理の実装に関する情報	28
PM 機能の概要	28
PM 統計情報サーバ	28
PM 統計情報収集機能	28
PM の利点	29
PM 統計情報収集の概要	29
PM 統計情報をエクスポートするためのバイナリ ファイル形式	30
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	31
バイナリ ファイルに適用されるファイルの命名規則	33
パフォーマンス管理の実装方法	34
PM 統計情報収集用の外部 TFTP サーバまたはローカル ディスクの設定	34
PM 統計情報収集テンプレートの設定	34
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	35
PM しきい値モニタリング テンプレートの設定	36
正規表現によるインスタンス フィルタリングの設定	37
パフォーマンス管理：詳細	37

第 4 章

Embedded Event Manager ポリシーの設定および管理 53

Embedded Event Manager ポリシーの設定および管理の前提条件	54
Embedded Event Manager ポリシーの設定および管理について	54
Event Management	54
システム イベント処理	54
Embedded Event Manager スクリプト	55
Embedded Event Manager ポリシー Tcl コマンド拡張カテゴリ	55
Embedded Event Manager 用のシスコ ファイル命名規則	56
Embedded Event Manager の組み込みアクション	57
アプリケーション固有の組み込みイベント管理	58
イベント検出とリカバリ	59
System Manager イベント ディテクタ	59
タイマー サービス イベント ディテクタ	60
syslog イベント ディテクタ	60
None イベント ディテクタ	61
Watchdog System Monitor イベント ディテクタ	61
分散イベント ディテクタ	62
Embedded Event Manager イベントのスケジューリングおよび通知	63
信頼性統計情報	63
Embedded Event Manager ポリシーの設定および管理方法	65
環境変数の設定	65
Embedded Event Manager ポリシーの登録	65
Tcl を使用した Embedded Event Manager ポリシーの記述方法	66
EEM Tcl スクリプトの登録と定義	66
EEM ポリシー実行の一時停止	67
EEM ポリシーを格納するディレクトリの指定	67
Tcl を使用した EEM ポリシーのプログラミング	68
EEM ユーザ Tcl ライブラリ索引の作成	73
EEM ユーザ Tcl パッケージ索引の作成	76
TCL を使用した EEM ポリシー：詳細	79

第 5 章

IP サービス レベル契約の実装	83
IP サービス レベル契約テクノロジーの概要	83
サービス レベル契約	84
IP サービス レベル契約の利点	85
IP サービス レベル契約を実装する前提条件	86
IP サービス レベル契約の実装の制限	86
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	87
IP SLA レスポンダおよび IP SLA 制御プロトコル	88
IP SLA の応答時間の計算	88
IP SLA 動作のスケジューリング	89
IP サービス レベル契約の動作タイプ	89
IP SLA の VRF サポート	90
IP SLA : 予防的しきい値モニタリング	90
IP SLA 反応コンフィギュレーション	91
IP SLA しきい値モニタリングおよび通知	91
Two-Way Active Measurement Protocol (TWAMP)	92
TWAMP エンティティ	92
TWAMP プロトコル	93
ルータでの TWAMP の制限事項	94
ルータでの TWAMP の設定	94
TWAMP の確認	95
TWAMP Light	95
Cisco NCS 540 シリーズ ルータ上の TWAMP Light の制限事項	96
NCS 540 シリーズ ルータでの TWAMP Light の設定	96
NCS 540 シリーズ ルータでの TWAMP Light の検証	96
MPLS LSP モニタリング	97
MPLS LSP モニタリングのしくみ	97
BGP ネクストホップ ネイバー探索	98
IP SLA LSP ping 動作と LSP traceroute 動作	99
MPLS LSP モニタリングの予防的しきい値モニタリング	100

LSP ヘルス モニタの複数動作スケジューリング	100
LSP パス ディスカバリ	100
IP サービス レベル契約の実装方法	101
UDP ジッター動作を使用した IP サービス レベルの設定	101
宛先デバイスでの IP SLA レスポンダのイネーブル化	101
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	103
送信元デバイスで UDP ジッター動作を設定する前提条件	104
送信元デバイスでの基本 UDP ジッター動作の設定およびスケジューリング	104
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	106
UDP エコー動作のための IP SLA の設定	111
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	111
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	111
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	114
ICMP エコー動作の設定	117
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	117
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	120
ICMP パスエコー動作の設定	123
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	123
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	126
ICMP パスジッター動作の設定	129
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	130
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	133
IP SLA MPLS LSP ping 動作およびトレース動作の設定	136
MPLS LSP ping 動作の設定およびスケジューリング	137
MPLS LSP トレース動作の設定およびスケジューリング	140
IP SLA 反応としきい値のモニタリングの設定	144
IP SLA 反応のモニタ対象の要素の設定	144
IP SLA 反応のしきい値違反タイプの設定	148
反応イベントの指定	152

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	154
MPLS LSP モニタリング ping インスタンスの設定	154
MPLS LSP モニタリング トレース インスタンスの設定	158
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	162
送信元 PE ルータでの MPLS LSP モニタリング インスタンスのスケジュール設定	163
LSP パス ディスカバリ	165
IP サービス レベル契約を実装するための設定例	167
IP サービス レベル契約の設定：例	167
IP SLA 反応としきい値のモニタリングの設定：例	168
IP SLA MPLS LSP モニタリングの設定：例	169
LSP パス ディスカバリの設定：例	170



第 1 章

システム ロギングの実装

このモジュールでは、ロギングサービスをルータに実装する必要があるタスクを説明します。Cisco IOS XR ソフトウェアには基本ロギング サービスが用意されています。ロギング サービスでは、システムロギング (syslog) メッセージモニタリングおよびトラブルシューティングのロギング情報を収集し、取得したロギング情報のタイプを選択できます。

システム ロギング実装の機能履歴

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

- [システム ロギングの実装 \(1 ページ\)](#)

システム ロギングの実装

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

syslog メッセージ形式

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

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

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

表 1: *syslog* メッセージの形式

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

syslog メッセージの重大度

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

表 2: *syslog* メッセージの重大度

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

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

システム ロギングの設定に関する前提条件

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

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

システム ログ機能の設定

必要に応じてシステム ロギングを設定するには、この項のタスクを実行します。

ロギング バッファへのロギングの設定

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

設定例

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

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

リモート サーバへのロギングの設定

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

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

表 3: Syslog メッセージを管理するための機能

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

設定例

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

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

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

関連項目

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

端末回線へのロギングの設定

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

設定例

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

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

コンソール端末へのロギングの変更

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

設定例

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

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

タイムスタンプ形式の変更

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

設定例

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

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

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

重複 syslog メッセージの抑制

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

設定例

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

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

ローカルストレージデバイスへのシステム ロギング メッセージのアーカイブ

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

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

表 4: syslog アーカイブ属性を設定するために使用するコマンド

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

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

設定例

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

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

プラットフォーム自動モニタリング

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

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

PAM イベント

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

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

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

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

以下の項では、PAM の主なイベントについて説明します。

クラッシュのモニタリング

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

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

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

トレースバックのモニタリング

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

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

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

メモリ使用率モニタリング

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

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

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

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

CPUモニタリング

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

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

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

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

ファイルシステムモニタリング

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

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

```

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

```

PAM の無効化と再有効化

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

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

- Cisco IOS XR コマンドライン インターフェイスから :

```

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

```

- ルータ シェル プロンプトから :

```

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

```

PAM の無効化

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

ローカル RP の場合：

```
Router# process shutdown pam_manager
```

すべての RP の場合：

```
Router# process shutdown pam_manager location all
```

PAM の再有効化

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

ローカル RP の場合：

```
Router# process start pam_manager
```

すべての RP の場合：

```
Router# process start pam_manager location all
```



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

PAM でのデータ アーカイブ

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

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

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

PAM ツールによって収集されるファイル

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

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

イベント名	PAM によって収集されるコマンドおよびファイル
プロセスのクラッシュ	<ul style="list-style-type: none"> • show install active • show platform • show version • コア (gz) ファイル • core.txt ファイル
プロセスのトレースバック	<ul style="list-style-type: none"> • show dll • show install active • show logging • show platform • show version
メモリ リーク	<ul style="list-style-type: none"> • show install active • show platform • show version • コア (gz) ファイル • 実行中のダンプコア • 連続メモリ使用量のスナップショット
ログギング イベントの表示	<ul style="list-style-type: none"> • show install active • show logging • show platform • show version • コア (gz) ファイル • core.txt ファイル

イベント名	PAM によって収集されるコマンドおよびファイル
CPU ホッグ	<ul style="list-style-type: none">• follow process• pstack• show dll• show install active• show platform• show version• top -H• コア (gz) ファイル• CPU 使用率のスナップショット
ディスク使用量	<ul style="list-style-type: none">• show install active• show platform• show version• コンソール ログ• コア (gz) ファイル• ディスク使用率のスナップショット



第 2 章

アラーム ログ関連の実装

このモジュールでは、アラーム ログ関連の設定に関する概念とタスクについて説明します。アラーム ログ関連は、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 ログを拡張します。

- [アラーム ログ関連の実装 \(15 ページ\)](#)

アラーム ログ関連の実装

アラーム ログ関連は、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 ログを拡張します。このモジュールでは、アラーム ログ関連の設定とアラーム ログのモニタリングに関連する概念とタスクについて説明します。

アラーム ログ関連の実装に関する前提条件

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

アラーム ログ関連の実装に関する情報

アラーム ログおよびデバッグ イベント管理システム

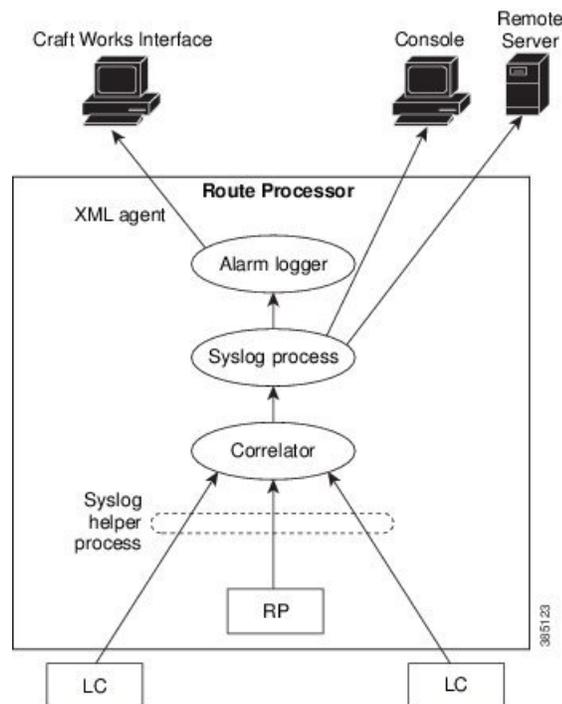
Cisco IOS XR ソフトウェアのアラーム ログおよびデバッグ イベント管理システム (ALDEMS) は、システム サーバおよびアプリケーションから転送されるアラーム メッセージをモニタリングし、格納するために使用されます。また、ALDEMS は、単一の根本原因のために転送されたアラーム メッセージ同士を関連します。

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

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

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

図 1: ALDEMS コンポーネント通信



コリレータ

コリレータは、ルータ上のノード全体に分散されたシステムログ (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。ログング関連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グング関連バッファに保存されます。相関メッセージの各セットには、コリレータにより相関 ID がタグ付けされます。

システム ロギング プロセス

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

アラーム ロガー

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



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

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

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

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

アラーム ログ関連の設定

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

ロギング 関連ルールの設定

ロギング 関連を使用して、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。関連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 バッファから関連メッセージをすべて取得して、発生した関連イベントを表示できます。関連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関連が発生します。関連ルールを特定のコンテキスト

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

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

設定例

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

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

ロギング相関ルールセットの設定

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

設定例

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

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

根本原因アラームと非根本原因アラームの相関

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

設定例

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

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

階層的な相関ルール フラグの設定

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

設定例

次に、階層的な相関ルールのフラグを設定する例を示します。

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

ロギング抑制ルールの設定

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

設定例

次の例に、ロギング抑制ルールの設定方法を示します。

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

ログイベントバッファ設定の変更

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

設定例

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

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

ログ関連バッファ設定の変更

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

設定例

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

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

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

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

設定例

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

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

SNMP 関連ルールの設定

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

設定例

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

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

SNMP 関連ルールセットの設定

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

設定例

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

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

アラーム ロギング関連の詳細

アラーム ロギング関連を使用して、システム パフォーマンスに影響を及ぼすイベントの最上位ルートメッセージを分離できます。たとえば、ラインカードの活性挿抜 (OIR) を示す元のメッセージが分離され、根本原因メッセージのみ表示され、同じイベントに関連するすべての後続メッセージが関連になる場合があります。関連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 バッファから関連メッセージをすべて取得して、発生した関連イベントを表示できます。

関連ルール

関連ルールを設定して、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。関連ルールは、アラームログおよびデバッグイベント管理システム (ALDEMS) に、不要なメッセージの蓄積に起因する不要なストレスを与えないようにします。各関連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グイベントバッファ内のアラームローガーに転送され、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、ログ関連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関連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グコリレータバッファ内の一致する関連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、ログ関連ルールのメッセージが定義されます。

- メッセージカテゴリ
- メッセージグループ
- メッセージコード

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

根本原因メッセージ、ステートフル関連および非ステートフル関連を分離するために、ルールには2つのタイプの関連が設定されています。非ステートフル関連は、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関連バッファにバッファされた状態で残ります。ステートフル関連は、バイステート根本原因アラームがクリアされると、関連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、すべての関連された非根本原因アラームはsyslogに送信され、関連バッファからは削除されます。ステートフル関連は、疑われる根本原因がすでに存在しないにもかかわらず、存在し続けている非根本原因状態を検出する場合に役立ちます。

アラーム重大度とフィルタリング

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

アラームは、次の表に示されている重大度に応じてフィルタリングできます。

表 5: イベント ログिंगのアラーム重大度

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

バイステート アラーム

バイステート アラームは、アクティブから非アクティブへのインターフェイス ステートの変化、ラインカードの活性挿抜 (OIR)、またはコンポーネントの温度の変化など、システムハードウェアに関連するステート変更によって生成されます。デフォルトでは、バイステートアラーム イベントはログング 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 ログング メカニズムを登録できます。

バイステート アラーム通知により、次のことがわかります。

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

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

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

次に、バイステート アラーム メッセージの例を示します。

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

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

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

アラームソースのロケーションを表示した場合、一般形式は次のようになります。

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

次に、アラームソースのロケーションが表示されている場合の例を示します。

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

階層的な相関

階層的な相関は、次の条件が満たされた場合に有効になります。

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

次に、2つの階層的な相関ルールの例を示します。

ルール 1	カテ ゴ リ	グル ー プ	コー ド
Root Cause 1	Cat 1	Group 1	Code 1
Non-root Cause 2	Cat 2	Group 2	Code 2
ルール 2			
Root Cause 2	Cat 2	Group 2	Code 2
Non-root Cause 3	Cat 3	Group 3	Code 3

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

Cause 1 -> Cause 2 -> Cause 3

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



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

コンテキスト関連フラグ

コンテキスト関連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、コンテキスト関連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 にはコンテキスト関連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 にコンテキスト関連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 にコンテキスト関連フラグが設定され、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、コレ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、コンテキスト関連動作の例を示します。

- Rule 1 には、根本原因 A が含まれ、非根本原因が関連付けられている。
- Rule 1 にはコンテキスト関連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 にコンテキスト関連フラグが設定され、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、関連されません。



第 3 章

パフォーマンス管理の実装

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

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

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

- [パフォーマンス管理を実装する前提条件](#) (27 ページ)
- [パフォーマンス管理の実装に関する情報](#) (28 ページ)
- [PM 機能の概要](#) (28 ページ)
- [PM の利点](#) (29 ページ)
- [PM 統計情報収集の概要](#) (29 ページ)
- [パフォーマンス管理の実装方法](#) (34 ページ)

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

パフォーマンス管理の実装に関する情報

PM 機能の概要

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

- PM 統計情報サーバ
- PM 統計情報収集機能

PM 統計情報サーバ

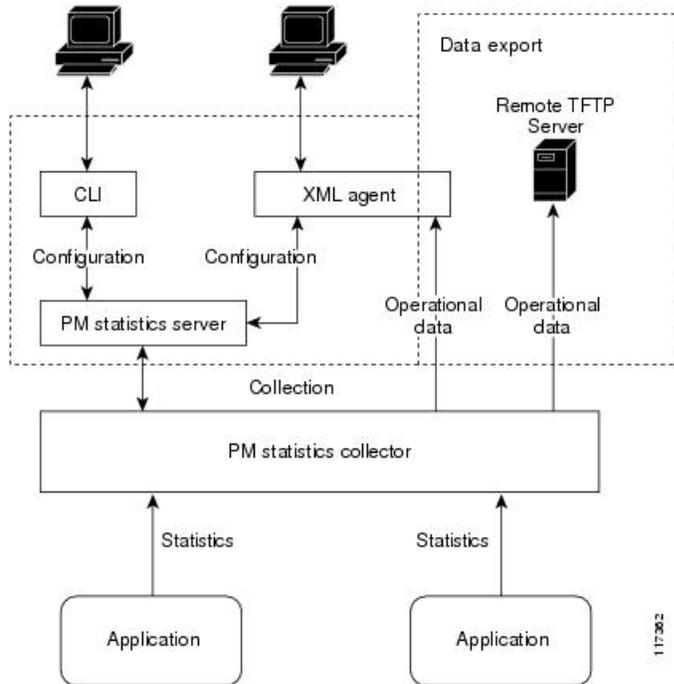
PM 統計情報サーバは統計情報収集、エンティティインスタンスモニタリング収集、しきい値モニタリングのフロントエンドです。コマンドラインインターフェイス（CLI）またはXMLスキームから設定されたすべてのPM 統計情報収集およびしきい値条件は、PM 統計情報サーバによって処理され、PM 統計情報機能に分散されます。

PM 統計情報収集機能

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

[図 2 : PM コンポーネントの通信 \(29 ページ\)](#) 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 2: PM コンポーネントの通信



PM の利点

PM システムには次の利点があります。

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

PM 統計情報収集の概要

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

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

表 6: エンティティ クラスおよび関連付けられているインスタンス

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



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



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

PM 統計情報をエクスポートするためのバイナリ ファイル形式

次のサンプルでは、バイナリ ファイル形式を説明します。

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

```

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

```

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

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

表 7: バイナリ形式の値とキー

エンティティ	サブエンティティ	キー	統計情報カウンタ
ノード (1)	CPU (1)	CPU キー <Node ID>	「表 8 : エンティティとサブエンティティでサポートされる統計情報カウンタ (32 ページ)」を参照してください
	メモリ (2)	メモリ キー <Node ID>	
	プロセス (3)	ノード プロセス キー <NodeProcessID>	
インターフェイス (2)	汎用カウンタ (1)	汎用カウンタ キー <ifName>	
	データ レート カウンタ (2)	データ レート カウンタ キー <ifName>	
	基本カウンタ (3)	基本カウンタ キー <ifName>	
BGP (3)	ピア (1)	ピア キー <IpAddress>	
MPLS (4)	予約済み (1)	—	
	予約済み (2)	—	
	LDP (4)	LDP セッション キー <IpAddress>	
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 7: バイナリ形式の値とキー (31 ページ)」の各エンティティとサブエンティティに関連付けられている数字) は、TFTP ファイルに表示されるエンティティ ID とサブエンティティ ID を表します。

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

表 8: エンティティとサブエンティティでサポートされる統計情報カウンタ

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

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

バイナリファイルに適用されるファイルの命名規則

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

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

パフォーマンス管理の実装方法

PM 統計情報収集用の外部 TFTP サーバまたはローカルディスクの設定

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

設定例

次の例では、PM 統計情報収集用に外部の TFTP サーバを設定します。

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

次の例では、PM 統計情報収集用にローカルディスクを設定します。

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

PM 統計情報収集テンプレートの設定

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

PM 統計情報収集テンプレートを設定するガイドライン

PM 統計情報テンプレートの作成時は、次のガイドラインに従ってください。

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

- 5つのサンプリング動作のサンプル サイズ。
- サンプル間隔は、サンプリングサイクル中に実行されるサンプリング動作の頻度を設定します。 **sample-interval** コマンドを使用して、サンプルの間隔を設定できます。範囲は1～60分です。
- **sample size** では、データを TFTP サーバにエクスポートする前に実行されるサンプリング動作の数を設定します。 **sample-size** コマンドを使用して、サンプルのサイズを設定できます。範囲は1～60サンプルです。



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

- エクスポート サイクルでは、PM 統計情報収集データが TFTP サーバにエクスポートされる頻度を決定します。エクスポート サイクルは、サンプル間隔にサンプル サイズを掛け合わせて計算します (サンプル間隔 x サンプル サイズ = エクスポート サイクル)。
- テンプレートをイネーブルにすると、**performance-mgmt apply statistics** コマンドの **no** 形式でテンプレートをディセーブルにするまで、サンプリングとエクスポートのサイクルは継続されます。
- 次のエンティティの PM 統計情報収集を有効または無効にするときは、**location** コマンドを使用してノードを指定するか、または **location all** コマンドを使用してすべてのノードの PM 統計情報収集を有効にする必要があります。
 - ノード CPU
 - ノード メモリ
 - ノード プロセス

設定例

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

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

PM エンティティ インスタンス モニタリングのイネーブル化

エンティティ インスタンス モニタリングでは、特定のエンティティ インスタンスに関連付けられている属性から統計情報を収集します。エンティティ インスタンスのモニタリングがイ

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

設定例

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

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

PM しきい値モニタリングテンプレートの設定

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

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

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

設定例

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

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

正規表現によるインスタンス フィルタリングの設定

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

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



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

設定例

次に、正規表現グループを定義する例を示します。

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

パフォーマンス管理：詳細

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

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

表 9: 属性と値

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

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

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

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

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

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

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

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

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

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

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

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

表 10: エンティティインスタンスとモニタリングコマンド

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

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

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	<p>インターフェイス汎用カウンタのエンティティ インスタンスのエンティティ インスタンス モニタリングを有効にするには、 performance-mgmt apply monitor interface generic-counters コマンドを使用します。</p> <p>構文：</p> <pre> performance-mgmt apply monitor interface generic-counters type interface-path-id {template-name default} </pre> <p>RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor interface generic-counters HundredGigE 0/0/1/0 default</p>
MPLS LDP	<p>MPLS LDP エンティティ インスタンスのエンティティ インスタンス モニタリングを有効にするには、 performance-mgmt apply monitor mpls ldp コマンドを使用します。</p> <p>構文：</p> <pre> performance-mgmt apply monitor mpls ldp ip-address {template-name default} </pre> <p>RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor mpls ldp 10.34.64.154 default</p>
ノード CPU	<p>ノード CPU エンティティ インスタンスのエンティティ インスタンス モニタリングを有効にするには、 performance-mgmt apply monitor node cpu コマンドを使用します。</p> <p>構文：</p> <pre> performance-mgmt apply monitor node cpu location node-id {template-name default} </pre> <p>RP/0/RP0/CPU0:Router(config)# performance-mgmt apply monitor node cpu location 0/RP0/CPU0 default</p>

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



第 4 章

Embedded Event Manager ポリシーの設定および管理

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

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

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

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

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

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

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

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

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

Event Management

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

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

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

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

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

システム イベント処理

イベント通知を受信すると、EEM は次のアクションを実行します。

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

Embedded Event Manager スクリプト

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

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

EEM スクリプトの作成には、次の手順が含まれます。

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

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

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

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

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

Embedded Event Manager 用のシスコ ファイル命名規則

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

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

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

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

表 12: 2文字の省略形の指定

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

Embedded Event Manager の組み込みアクション

EEM の組み込みアクションは、EEM ハンドラが動作するときにハンドラから要求できます。次の表に、各 EEM ハンドラ要求またはアクションを示します。

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

Embedded Event Manager の組み込みアクション	説明
syslog へのメッセージの記録	メッセージを <code>syslog</code> 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	指定されたチャンネルハンドラにコマンドを書き込み、 cli_exec コマンド拡張を使用してコマンドを実行します。
syslog メッセージの生成	action_syslog Tcl コマンド拡張を使用して、メッセージをログに記録します。
手動による EEM ポリシーの実行	event manager run コマンドをモードでポリシーを実行している間に、ポリシー内で EEM ポリシーを実行します。

Embedded Event Manager の組み込みアクション	説明
アプリケーション固有のイベントのパブリッシュ	<code>event_publish appl Tcl</code> 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	<code>EEM action_reload</code> 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 <code>sys_reqinfo_xxx</code> 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も識別され、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、サブスクリバがない場合でも、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、サブスクリ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リブする EEM スクリプトは、次の順序で処理されます。

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

イベント検出とリカバリ

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

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

- システム EEM イベント
- アプリケーション定義イベント

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

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

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

次のイベント ディテクタがサポートされています。

System Manager イベント ディテクタ

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

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

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

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

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

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

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

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

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

タイマー サービス イベント ディテクタ

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

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

syslog イベント ディテクタ

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

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

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

表 15: syslog イベント デテクタイベント 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 of 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 デテクタ

None イベント デテクタは、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 デテクタ

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

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

表 16: Watchdog System Monitor イベント デテクタ要求

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

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

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 デテクタ

EEM イベント デテクタと通信し、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 デテクタが必要です。分散イベント デテクタでは、EEM 通信チャンネルへのローカルハードウェア

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

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

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

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

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

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

信頼性統計情報

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

ハードウェアカードの信頼性メトリック データ

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

ハードウェアカードによって保持されるデータは、次のとおりです。

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

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

プロセス信頼性メトリック データ

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

プロセスの終了には次の場合が含まれます。

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

プロセスが保持するデータは次のとおりです。

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

Embedded Event Manager ポリシーの設定および管理方法

環境変数の設定

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

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

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

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

設定例

次の例では、一連の EEM 環境変数を定義する方法を示します。

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

No.	Name	Value
1	_email_to	beta@cisco.com
2	_cron_entry	0-59/2 0-23/1 * * 0-7
3	_email_from	beta@cisco.com

```
RP/0/RP0/CPU0:Router#
```

Embedded Event Manager ポリシーの登録

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

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

EEM ポリシーの登録時に以下を指定する必要があります。

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



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

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

設定例

次に、ユーザ定義の EEM ポリシーを登録する例を示します。

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

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

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

この項では、次の作業について説明します。

EEM Tcl スクリプトの登録と定義

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



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

設定例

次に、EEM ポリシーを登録して定義する例を示します。

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



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

EEM ポリシー実行の一時停止

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

設定例

次に、すべての EEM ポリシーの実行を一時停止する例を示します。

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

EEM ポリシーを格納するディレクトリの指定

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

設定例

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

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

Tcl を使用した EEM ポリシーのプログラミング

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

手順

	コマンドまたはアクション	目的
ステップ 1	show event manager policy available [system user] 例： <pre>RP/0/RP0/CPU0:Router# show event manager policy available</pre>	登録可能な EEM ポリシーを表示します。
ステップ 2	画面に表示されたサンプルポリシーの内容を、テキストエディタにカットアンドペーストします。	—
ステップ 3	必要な <code>event_register</code> 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 <code>event_register</code> 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。 <ul style="list-style-type: none"> • <code>event_register_appl</code> • <code>event_register_counter</code> • <code>event_register_stat</code> • <code>event_register_wdssysmon</code> • <code>event_register_oir</code> • <code>event_register_process</code> • <code>event_register_syslog</code> • <code>event_register_timer</code> • <code>event_register_timer_subscriber</code> • <code>event_register_hardware</code> • <code>event_register_none</code>
ステップ 4	適切な名前空間を、 <code>::cisco</code> 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 <code>::cisco</code> を使用できま

	コマンドまたはアクション	目的
		<p>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。</p> <ul style="list-style-type: none"> • ::cisco::eem <ul style="list-style-type: none"> • 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 <ul style="list-style-type: none"> • SMTP ライブラリ <p>(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。</p>
<p>ステップ 5</p>	<p>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。</p>	<p>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</p>

	コマンドまたはアクション	目的
		<p>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。</p> <p>(注) show event manager environment コマンドを使用して、システムの Embedded Event Manager 環境変数セットを表示できます。</p> <p>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。</p> <ul style="list-style-type: none"> • _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。	<p>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。</p> <ul style="list-style-type: none"> • 検出されたイベントに関する情報の EEM への問い合わせに使用される event_reqinfo イベント情報の Tcl コマンド拡張。

	コマンドまたはアクション	目的
		<ul style="list-style-type: none"> • EEM 特有のアクションの指定に使用される、action_syslog などのアクション Tcl コマンド拡張。 • 一般的なシステム情報の取得に使用される、sys_reqinfo_routername などのシステム情報の Tcl コマンド拡張。 • 他のポリシーによって使用される Tcl 変数の保存に使用される context_save および context_retrieve の Tcl コマンド拡張。 • ポリシーからの、SMTP ライブラリ（電子メール通知を送信）または CLI ライブラリ（CLI コマンドを実行）の使用。
ステップ 7	開始ステータスをチェックし、ポリシーがこのイベントに対して前に実行されたかどうかを判断します。	前のポリシーが正常終了した場合、現在のポリシーは、実行が必要な場合と、実行が不要な場合があります。開始ステータス指定には、0（前のポリシーが正常終了した）、Not=0（前のポリシーに障害が発生した）、および Undefined（実行された前のポリシーがない）の、3つの値のうちいずれか1つを使用できます。
ステップ 8	終了ステータスをチェックし、デフォルトアクションが存在する場合に、このイベントのデフォルトアクションが適用されたかどうかを判断します。	値 0 は、デフォルトのアクションを実行しないことを意味します。0 以外の値は、デフォルトのアクションを実行する必要があることを意味します。終了ステータスは、同じイベントで実行される後続ポリシーに渡されます。
ステップ 9	Cisco エラー番号（_cerno）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 _cerno が設定されます。_cerno が設定されるたびに、他の 4 つの Tcl グローバル変数が _cerno から分岐し、それとともに設定されます（_cerr_sub_num、_cerr_sub_err、および _cerr_str）。

	コマンドまたはアクション	目的
ステップ 10	新しいファイル名で Tcl スクリプトを保存し、Tcl スクリプトをルータにコピーします。	<p>Embedded Event Manager ポリシー ファイル名は、次の仕様に従っています。</p> <ul style="list-style-type: none"> • オプションのプレフィックス Mandatory. がある場合、これは、システムポリシーがまだ登録されていない場合に、自動的に登録される必要があるシステムポリシーであることを示します。たとえば、Mandatory.sl_text.tcl などです。 • 指定された 1 つめのイベントの 2 文字の省略形が含まれるファイル名の本体部（「表 12: 2 文字の省略形の指定 (57 ページ)」を参照）、下線文字部、および、ポリシーをさらに示す説明フィールド部。 • ファイル名拡張子部は .tcl と定義されます。 <p>詳細については、「Embedded Event Manager 用のシスコ ファイル命名規則 (56 ページ)」を参照してください。</p> <p>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。</p>
ステップ 11	configure	グローバル コンフィギュレーション モードを開始します。
ステップ 12	event manager directory user { library path policy path } 例 : <pre>RP/0/RP0/CPU0:Router(config)# event manager directory user library disk0:/user_library</pre>	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 : <pre>RP/0/RP0/CPU0:Router(config)# event</pre>	ポリシー内で定義された指定イベントが発生した場合に、EEM ポリシーを実行するよう、定義します。

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

EEM ユーザ Tcl ライブラリ索引の作成

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

手順

	コマンドまたはアクション	目的
ステップ 1	ワークステーション（UNIX、Linux、PC、またはMac）で、ライブラリディレクトリを作成し、Tcl ライブラリファイルをディレクトリにコピーします。	次の例ファイルを使用すると、Tcl シェルが実行されているワークステーション上で、 <code>tclIndex</code> を作成できます。 lib1.tcl <pre>proc test1 {} { puts "In procedure test1" } proc test2 {} { puts "In procedure test2" }</pre> lib2.tcl <pre>proc test3 {} { puts "In procedure test3" }</pre>
ステップ 2	tclsh 例： <code>workstation% tclsh</code>	Tcl シェルを開始します。

	コマンドまたはアクション	目的
ステップ 3	<p>auto_mkindex <i>directory_name</i> *.tcl</p> <p>例 :</p> <pre>workstation% auto_mkindex eem_library *.tcl</pre>	<p>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。</p> <p>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。</p> <p>tclIndex</p> <pre># Tcl autoload index file, version 2.0 # This file is generated by the "auto_mkindex" command # and sourced to set up indexing information for one or more commands. Typically each line is a command that # sets an element in the auto_index array, where the # element name is the name of a command and the value is # a script that loads the command. set auto_index(test1) [list source [file join \$dir lib1.tcl]] set auto_index(test2) [list source [file join \$dir lib1.tcl]] set auto_index(test3) [list source [file join \$dir lib2.tcl]]</pre>
ステップ 4	<p>ターゲットルータ上のユーザライブラリ ファイルの保存に使用されるディレクトリに、ステップ 1 から Tcl ライブラリ ファイルをコピーし、ステップ 3 から tclIndex ファイルをコピーします。</p>	—

	コマンドまたはアクション	目的
ステップ 5	Tcl で記述されたユーザ定義 EEM ポリシーファイルを、ターゲットルータ上でユーザ定義 EEM ポリシーの保存に使用されるディレクトリにコピーします。	ディレクトリは、ステップ 4 で使用されるディレクトリと同じディレクトリを使用できます。 次に、EEM でサポートされる Tcl ライブラリのテストに、ユーザ定義 EEM ポリシーを使用できる例を示します。 libtest.tcl <pre> ::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" } </pre>
ステップ 6	configure	
ステップ 7	event manager directory user library path 例： RP/0/RP0/CPU0:Router(config)# event manager directory user library disk0:/eem_library	EEM ユーザ ライブラリのディレクトリを指定します。これは、ステップ 4 のファイルがコピーされたディレクトリです。
ステップ 8	event manager directory user policy path 例： RP/0/RP0/CPU0:Router(config)# event manager directory user policy disk0:/eem_policies	EEM ユーザ ポリシーのディレクトリを指定します。これは、ステップ 5 のファイルがコピーされたディレクトリです。
ステップ 9	event manager policy policy-name username username [persist-time [seconds infinite] type [system user]] 例： RP/0/RP0/CPU0:Router(config)# event manager policy libtest.tcl username user_a	ユーザ定義の EEM ポリシーを登録します。

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

EEM ユーザ Tcl パッケージ索引の作成

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

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

この作業では、**pkg_mkIndex** コマンドを使用して、適切なライブラリ ディレクトリに Tcl パッケージディレクトリ **pkgIndex.tcl** ファイルが作成され、バージョン情報とともに、ディレクトリに含まれるすべての Tcl パッケージについての情報が含まれます。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時、Tcl スクリプトは、外部エンティティに対する XML-RPC 呼び出しを容易に行うことができます。



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

手順

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

	コマンドまたはアクション	目的
ステップ 3	<p>pkg_mkindex <i>directory_name</i> *.tcl</p> <p>例 :</p> <pre>workstation% pkg_mkindex eem_library *.tcl</pre>	<p>pkg_mkindex コマンドを使用して、pkgIndex ファイルを作成します。すべてのパッケージのディレクトリが含まれる pkgIndex ファイルは、Tcl ライブラリファイルに含まれていました。どのディレクトリにも 1 つの pkgIndex ファイルのみを存在させることができ、他の Tcl ファイルはグループ化しておくことが可能であるため、ディレクトリ内で pkg_mkindex コマンドを実行することを推奨します。ディレクトリ内で pkg_mkindex コマンドを実行すると、特定の pkgIndex を使用してどの Tcl パッケージ ファイルを索引化できるかが判断されます。</p> <p>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。</p> <p>pkgIndex</p> <pre># 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]]</pre>
ステップ 4	<p>ターゲットルータ上のユーザライブラリファイルの保存に使用されるディレクトリに、ステップ 1 から Tcl パッケージ ファイルをコピーし、ステップ 3 から pkgIndex ファイルをコピーします。</p>	—

	コマンドまたはアクション	目的
ステップ 5	Tcl で記述されたユーザ定義 EEM ポリシーファイルを、ターゲットルータ上でユーザ定義 EEM ポリシーの保存に使用されるディレクトリにコピーします。	ディレクトリは、ステップ 4 で使用されるディレクトリと同じディレクトリを使用できます。 次に、EEM でサポートされる Tcl ライブラリのテストに、ユーザ定義 EEM ポリシーを使用できる例を示します。 packagetest.tcl <pre> ::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?" </pre>
ステップ 6	configure	
ステップ 7	event manager directory user library path 例： RP/0/RP0/CPU0:Router(config)# event manager directory user library disk0:/eem_library	EEM ユーザ ライブラリのディレクトリを指定します。これは、ステップ 4 のファイルがコピーされたディレクトリです。
ステップ 8	event manager directory user policy path 例： RP/0/RP0/CPU0:Router(config)# event manager directory user policy disk0:/eem_policies	EEM ユーザ ポリシーのディレクトリを指定します。これは、ステップ 5 のファイルがコピーされたディレクトリです。
ステップ 9	event manager policy policy-name username username [persist-time [seconds infinite] type [system user]] 例： RP/0/RP0/CPU0:Router(config)# event manager policy packagetest.tcl username user_a	ユーザ定義の EEM ポリシーを登録します。
ステップ 10	event manager run policy [argument] 例：	手動で EEM ポリシーを実行します。

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

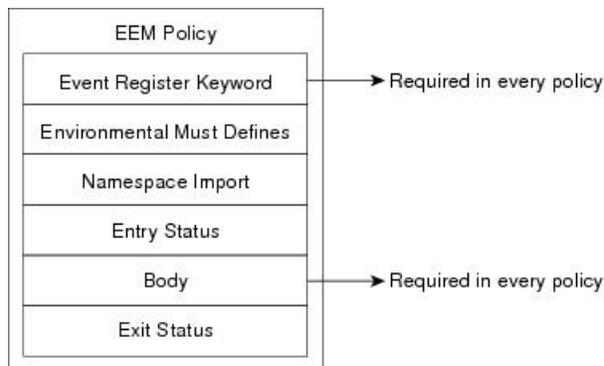
TCL を使用した EEM ポリシー : 詳細

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

Tcl ポリシーの構造と要件

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

図 3: Tcl ポリシーの構造と要件



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

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

次に、一部の環境変数をチェックし、定義する Tcl コードの例を示します。

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

```

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

```

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

```

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

```

ポリシーの本体は必須の構造で、次のものを含めることができます。

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

EEM 開始ステータス

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

開始ステータスの指定には、次の 3 種類の値のいずれかを使用できます。

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

EEM 終了ステータス

ポリシーでそのコードの実行を終了すると、終了値が設定されます。終了値は、EEM によって使用され、このイベントのデフォルトアクションがある場合に、それが適用されたかどうか判断されます。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 `_cerno` が設定されます。`_cerno` 変数が設定されると、他の Tcl グローバル変数が `_cerno` から分岐し、それとともに設定されます (`_cerr_sub_num`、`_cerr_sub_err`、および `_cerr_str`)。

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

```
XYSSSSSSSSSSSSSEEEEEEEEEPPPPPPPP
```

次の表に示されているように、この 32 ビットの整数は変数に分けられます。

表 17: `_cerno` : 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、 <code>CERR_CLASS_WARNING</code> を示します。 この変数特有の 4 つのエラー クラス エンコーディングについては、「 表 18: エラー クラス エンコーディング (81 ページ) 」を参照してください。
SSSSSSSSSSSSSS	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 <code>\$_cerr_sub_num</code> 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、 <code>\$_cerr_sub_err</code> に含まれています。

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

```
862439AE
```

この数字は、次の 32 ビット値として解釈されます。

```
10000110001001000011100110101110
```

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

表 18: エラー クラス エンコーディング

エラー戻り値	エラー クラス
00	<code>CERR_CLASS_SUCCESS</code>

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

ゼロのエラー戻り値は、SUCCESS を示します。



第 5 章

IP サービス レベル契約の実装

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

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

この章では、次のトピックについて取り上げます。

- [IP サービス レベル契約テクノロジーの概要 \(83 ページ\)](#)
- [IP サービス レベル契約を実装する前提条件 \(86 ページ\)](#)
- [IP サービス レベル契約の実装の制限 \(86 ページ\)](#)
- [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 \(87 ページ\)](#)
- [IP サービス レベル契約の動作タイプ \(89 ページ\)](#)
- [IP SLA の VRF サポート \(90 ページ\)](#)
- [IP SLA : 予防的しきい値モニタリング \(90 ページ\)](#)
- [Two-Way Active Measurement Protocol \(TWAMP\) \(92 ページ\)](#)
- [TWAMP Light \(95 ページ\)](#)
- [MPLS LSP モニタリング \(97 ページ\)](#)
- [LSP パス ディスカバリ \(100 ページ\)](#)
- [IP サービス レベル契約の実装方法 \(101 ページ\)](#)
- [IP サービス レベル契約を実装するための設定例 \(167 ページ\)](#)

IP サービス レベル契約テクノロジーの概要

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

- 応答時間

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

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

この項では、次のトピックについて取り上げます。

サービス レベル契約

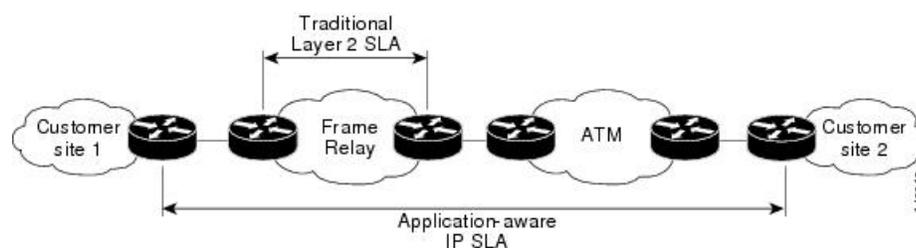
インターネットショッピングはこの数年で急激に成長し、テクノロジーの進化により高速で信頼性の高いインターネットアクセスが提供されるようになりました。多くの機能がオンラインアクセスを必要とし、その業務のほとんどをオンラインで行っており、サービスが失われると企業の収益に影響を与えることがあります。インターネットサービスプロバイダー（ISP）だけでなく、社内の IT 部門も、定義されたサービス レベル（サービス レベル契約）を提供し、顧客にある程度の予測性を提供するようになっていきます。

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



- (注)
- アプリケーションおよび IP SLA の処理速度が対応している場合は、IP SLA フローエントリのフローレートを最大で 1500 に指定できます。
 - IP SLA の高パフォーマンスの動作を実現するには、同じデバイスにおける複数の IP SLA 動作に同じ送信元ポートと宛先ポートを再利用しないようにします（特に大規模な場合）。

図 4: 従来のサービス レベル契約と IP SLA の範囲



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

表 19: 従来のサービス レベル契約に対する IP SLA の改良点

改良の種類	説明
エンドツーエンド測定	ネットワークの端からもう一方の端までパフォーマンスを測定できることにより、エンドユーザによるネットワーク利用状況をより広い到達範囲でより正確に表現できます。
詳細化	遅延、ジッター、パケットシーケンス、レイヤ3 接続、パスとダウンロード時間などの双方向のラウンドトリップの数値に詳細化される統計情報により、レイヤ2リンクの帯域幅だけよりも詳細なデータが得られます。
精度	ネットワーク パフォーマンスのわずかな変化にも影響を受けやすいアプリケーションは、ミリ秒以下の単位での IP SLA の測定精度を必要とします。
展開の簡易化	IP SLA は、大きいネットワーク内で既存のシスコ デバイスを活用することにより、従来のサービス レベル契約で必要になることが多い物理的な動作よりも、簡単に実装されます。
アプリケーション認識型の監視	IP SLA は、レイヤ3 からレイヤ7 の上で動作するアプリケーションによって生成されたパフォーマンス統計情報をシミュレートおよび測定できます。従来のサービス レベル契約では、レイヤ2 パフォーマンスしか測定できません。
普及	IP SLA は、ローエンドからハイエンドまでのルータとスイッチに及ぶ、シスコ ネットワーキング デバイスでサポートされています。この幅広い展開により、IP SLA は、従来のサービス レベル契約よりも高い柔軟性を備えています。

IP サービス レベル契約の利点

次の表に、IP SLA を実装することの利点の一覧を示します。

表 20: IP SLA の利点の一覧

利点	説明
IP SLA のモニタリング	サービス レベル契約モニタリング、評価、および検証の提供
ネットワーク パフォーマンス モニタリング。	ネットワーク内のジッター、遅延、パケット損失が測定できる。また、IP SLA は、予防的な通知に加えて、連続的で、信頼性が高く、予測可能な測定を提供します。

利点	説明
IP サービス ネットワーク稼働状態評価	既存の QoS が新しい IP サービスに対して十分であることの検証
ネットワーク動作のトラブルシューティング	問題をただちに特定し、トラブルシューティング時間を節約する、一貫し、信頼性が高い測定を提供します。

IP サービス レベル契約を実装する前提条件

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

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

IP サービス レベル契約の実装の制限

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



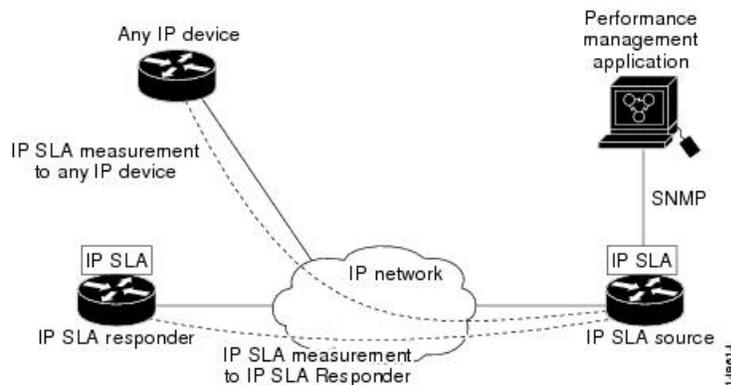
(注) 頻度を 60 秒未満に設定すると、送信されるパケット数が増加します。しかし、スケジュールされた動作の開始時刻が同じ場合、IP SLA 動作のパフォーマンスに悪影響を与える可能性があります。

- IP SLA は HA に対応していません。
- frequency、timeout、および threshold コマンドを設定する前に、次のガイドラインを検討してください。
- 制御無効モードでは、制御有効モードと比較すると、IP SLA の拡張性が向上します。

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

図 5: IP SLA の動作



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

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

ここでは、次の内容について説明します。

IP SLA レスポンダおよび IP SLA 制御プロトコル

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

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



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

UDP ジッター操作では IP SLA レスポンダを使用する必要があります。ターゲットルータ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動作パケットを送信できます。

IP SLA の応答時間の計算

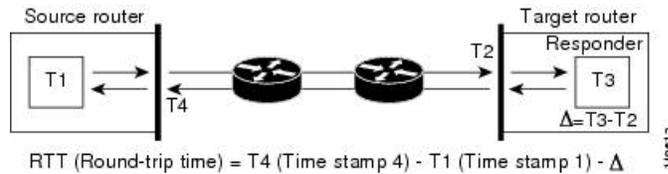
他の優先順位が高いプロセスにより、ルータは着信パケットを処理するために数十ミリ秒を要します。テストパケットへの応答が処理を待つ間キューに格納される可能性があるため、この遅延は応答時間に影響を与えます。この場合、応答時間は正しいネットワーク遅延を反映しません。IP SLA は、送信元ルータとターゲットルータ上でこれらの処理遅延を最小化し、真のラウンドトリップ遅延を判定します (IP SLA レスポンダが使用されている場合)。一部の IP SLA プロブパケットには、測定をより正確にするために、最終的な計算で使用される遅延情報が含まれています。

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

図 6: IP SLA レスポンダのタイムスタンプ処理 (89 ページ) に、レスポンダの動作を示します。T3 は、応答パケットが IP SLA レスポンダノードで送信された時刻であり、T1 は送信元ノードで要求が送信された時刻です。RTT を算出するためのタイムスタンプが 4 つ付けられます。ターゲットルータでレスポンダ機能がイネーブルの場合、タイムスタンプ 3 (TS3) からタイムスタンプ 2 (TS2) を引いてテストパケットの処理にかかった時間を求め、デルタ

(Δ) で表します。次に全体の RTT からこのデルタの値を引きます。精度を高めるため、優先度が高いパスで着信タイムスタンプ 4 (TS4) が取得される送信元ルータ上で、同じ原理が IP SLA によって適用されることに注意してください。

図 6: IP SLA レスポンダのタイムスタンプ処理



IP SLA 動作のスケジューリング

IP SLA 動作の設定が完了したら、その動作をスケジューリングして、統計情報の取得とエラー情報の収集を開始する必要があります。動作をスケジューリングすると、動作はただちに開始されるか、特定の月の特定の日に開始されます。また、動作を保留状態にすることができます。これは、その動作が、トリガーされるのを待つ反応（しきい値）動作である場合に使用されます。IP SLA 動作の通常のスケジューリングでは、一度に 1 つの動作をスケジューリングできます。

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 21: 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。
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。

動作	説明
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。 <ul style="list-style-type: none"> • 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 <p>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ー (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。</p>
MPLS LSP トレース	LSP パスのホップバイホップのルートを追跡し、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。 <p>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 であるかどうかを確認されます。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。</p>

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 21: IP SLA のための動作タイプ (89 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 : 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

IPSLA 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。

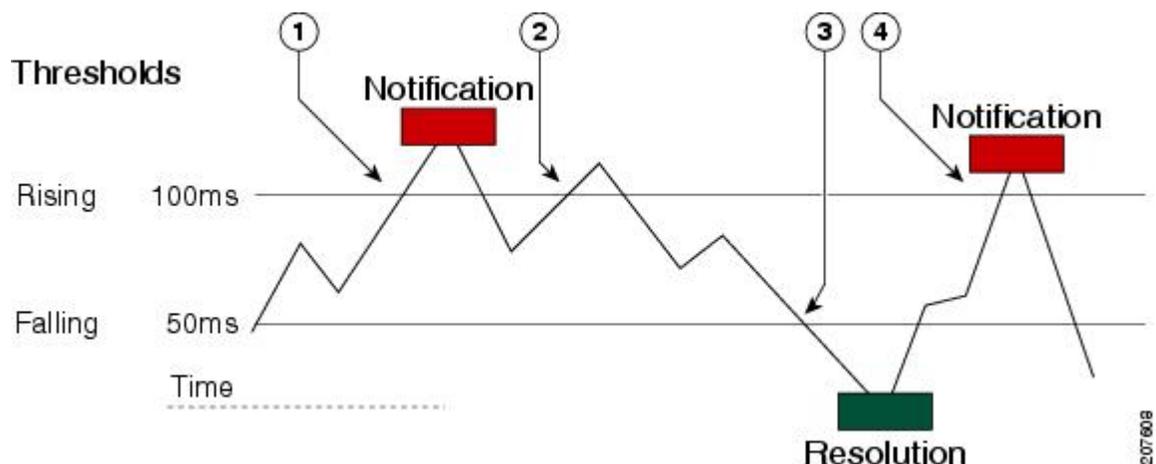
IP SLA しきい値モニタリングおよび通知

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。

通知は、しきい値違反が発生するたびに発行されるわけではありません。最初に上昇しきい値を超えたときに、イベントが送信され、通知が発行されます。後続のしきい値超過通知は、モニタリング対象の値が上昇しきい値を再び超える前に下限しきい値を下回った場合に限り発行されます。

次の図に、モニタリング対象要素が上限しきい値を超えたときに発生するトリガー反応の流れを示します。

図 7: IP SLA のトリガーされた反応条件およびしきい値超過通知



1	最初に上昇しきい値を超えたときに、イベントが送信され、しきい値超過通知が発行されます。
2	上昇しきい値の超過違反が連続して発生しても、追加の通知は発行されません。
3	モニタリング対象の値が下限しきい値を下回っています。
4	上昇しきい値を超えたときに別のしきい値超過通知が発行されているのは、モニタリング対象の値が最初に下限しきい値を下回った後だけです。

同様に、モニタリング対象の要素が下限しきい値を最初に下回った時点で、下限しきい値超過通知が発行されます。下限しきい値超過違反に対する後続の通知が発行されるのは、上昇しきい値を超えた後で、モニタリング対象の値が下限しきい値を再び下回った場合に限られます。

Two-Way Active Measurement Protocol (TWAMP)

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

TWAMP の利点

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



(注) TWAMP v4 および v6 がサポートされています。

ここでは、次の内容について説明します。

TWAMP エンティティ

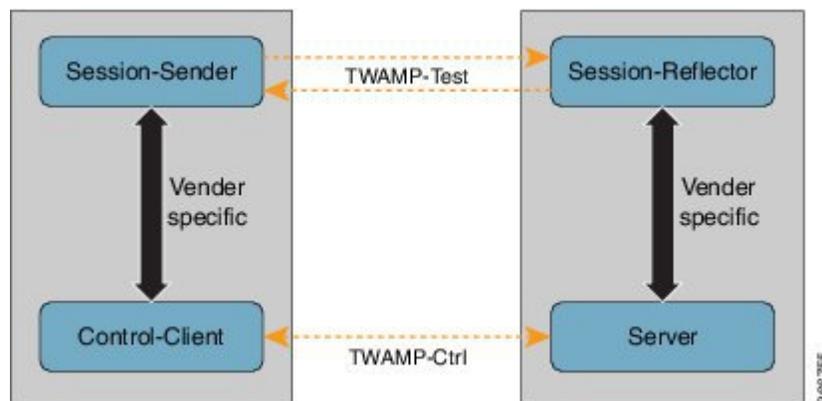
TWAMP システムは、4つの論理エンティティで構成されています。

- サーバ：1つ以上の TWAMP セッションを管理し、エンドポイント内のセッションごとのポートも構成します。

- セッションリフレクタ：TWAMP テスト パケットを受信するとすぐに測定パケットを反映します。
- 制御クライアント：TWAMP テスト セッションの開始と停止を開始します。
- セッション送信元：セッションリフレクタに送信された TWAMP テスト パケットをインスタンス化します。

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

図 8: TWAMP エンティティ



TWAMP プロトコル

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

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



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

- **TWAMP 制御交換**：TWAMP 制御プロトコルは TCP を介して実行され、測定セッションのインスタンス化および制御に使用されます。接続セットアップ交換とは異なり、TWAMP 制御コマンドは複数回送信できます。ただし、`session-start` コマンドの前に複数の

request-session コマンドを送信することはできますが、メッセージの順序が乱れることはありません。コマンドのシーケンスは、次のとおりです。

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

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

ルータでの TWAMP の制限事項

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

ルータでの TWAMP の設定

セッション サーバの設定

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

セッション リフレクタの設定

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

実行コンフィギュレーション

```
ipsla
 responder
 twamp
 !
 !
```

```
server twamp
  port 862
!
```

TWAMP の確認

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

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

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

```
Router# show ipsla twamp session
IP SLAs Responder TWAMP is: Enabled
Recv Addr: 10.5.139.11
Recv Port: 7222
Sender Addr: 172.27.111.233
Sender Port: 33243
Session Id: 10.5.139.11:70929508:88F7A620
Connection Id: 0
```

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

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

TWAMP Light

TWAMP Light は、制御セッションが不要な TWAMP の軽量モデルです。TWAMP 機能とは異なり、両方のエンドデバイスで TWAMP Light テストセッションのパラメータを設定する必要があります。これにより、制御セッションの確立と終了のオーバーヘッドが解消されます。さらに、リフレクタデバイスでサーバエンティティが不要であるため、サーバのメンテナンスのオーバーヘッドが軽減されます。



(注) TWAMP Light v4 および v6 がサポートされています。

Cisco NCS 540 シリーズ ルータ上の TWAMP Light の制限事項

- TWAMP Light テストセッションが Virtual Routing and Forwarding (VRF) インスタンスで実行されている場合、そのセッションは、インターフェイスで同じ VRF が設定されている場合にのみ機能します。
- デバイスで TWAMP Light テストセッションを設定すると、永続的なポートが開きます。このポートは、TWAMP Light の設定を削除するまで開いたままになります。この動作を希望しない場合は、タイムアウト時間後にセッションが非アクティブになるように TWAMP Light テストセッションのタイムアウトを設定する必要があります。
- 同じローカル IP アドレスとローカルポートを持つ 2 つの異なるテストセッションを持つ 2 つのクライアントが同じ VRF に存在する場合、レスポンスには基盤となるソケットが 1 つだけ存在します。このようなシナリオでは、UDP の制限により、これら 2 つのクライアントの最大パケット数をサポートすることはできません。これにより、パフォーマンスが影響を受けます。したがって、同じ VRF で 2 つのテストセッションが同じローカル IP アドレスとローカルポートを持つことはできません。

NCS 540 シリーズ ルータでの TWAMP Light の設定

```
Router# configure
Router(config)# ipsla
Router(config-ipsla)# responder
Router(config-ipsla-resp)# twamp-light test-session 1
Router(config-ipsla-resp)# local-ip 192.0.2.10 local-port 13001 remote-ip 192.0.2.186
remote-port 13002 vrf default
Router(config-ipsla-resp)# timeout 60
Router(config-ipsla-resp)# commit
```

実行コンフィギュレーション

```
ipsla
 responder
  twamp-light test-session 1
  local-ip 192.0.2.10 local-port 13001 remote-ip 192.0.2.186 remote-port 13002 vrf default

  timeout 60
  !
  !
  !
```

NCS 540 シリーズ ルータでの TWAMP Light の検証

TWAMP Light セッションは、`show ipsla twamp session` コマンドを使用して検証できます。コマンドの出力では、次に示すように、[Session status] フィールドを使用してセッションの状態が表示されます。

```
Router# show ipsla twamp session
***** TWAMP Sessions *****
No records matching query found
***** TWAMP-LIGHT Sessions *****
Session status: Active
```

```
Recv Addr: 192.0.2.10
Recv Port: 13001
Sender Addr: 192.0.2.186
Sender Port: 13002
Sender VRF Name: default
Session ID: 1
Mode: Unauthenticated
Number of Packets Received: 0
Session timeout: 60
```

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 \(98 ページ\)](#)」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

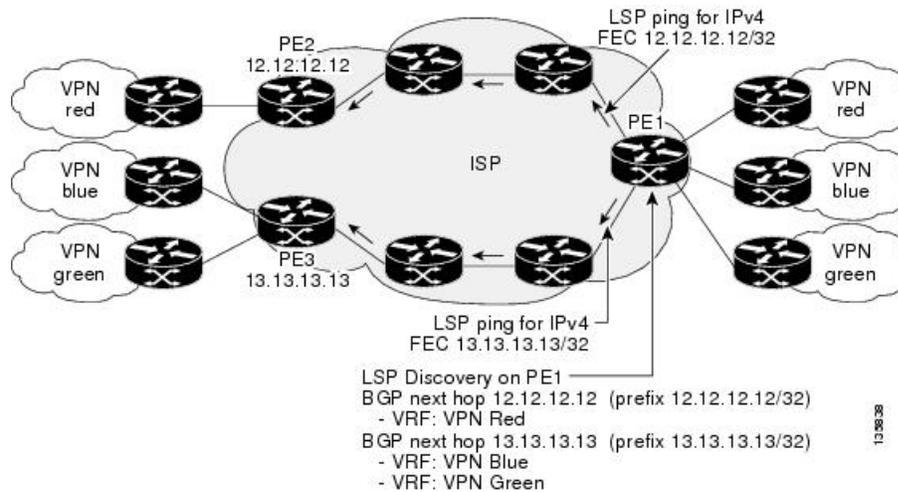
BGP ネクストホップネイバー探索は、送信元プロバイダーエッジ (PE) ルータに関連付けられているすべての VRF によって使用中の BGP ネクストホップネイバーを見つけるために使用されます。ほとんどの場合、これらのネイバーは PE ルータです。

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 9: 単純な VPN の BGP ネクストホップネイバー探索 (99 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 9: 単なる 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のネットワークの可用性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 \(154 ページ\)](#)」および「[MPLS LSP モニタリング トレース インスタンスの設定 \(158 ページ\)](#)」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 사용되는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等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。

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

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 10ms 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、10ms 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。

この項では、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

同じ、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、レスポンド側でオープンされます。

例は、レスポンド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、重複したパケットまたはアウトオブシーケンスパケットに対してチェックは実行されません。これは、プローブシーケンスの開始または終了を示す制御パケットがないためです。したがって、永続的なポートを使用する場合、シーケンス番号の検証はスキップされます。

手順

ステップ 1 configure

ステップ 2 ipsla responder

例：

```
RP/0/RP0/cpu 0: router(config)# ipsla responder
RP/0/RP0/cpu 0: 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。

ステップ 3 type udp ipv4 address ip-address port port

例：

```
RP/0/RP0/cpu 0: 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なアドレスとポートをイネーブルにします。

ステップ 4 commit

次のタスク

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 \(103 ページ\)](#)」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

IPSLA 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 で、送信元ルータからターゲットルータに、サイズ S の N 個の UDP パケットを T ミリ秒間隔で送信します。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 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 22 : UDP ジッター動作パラメータ \(103 ページ\)](#)」に示すように、ユーザが設定可能です。

表 22: 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	<ul style="list-style-type: none"> • <i>operation-number</i> 引数を指定した ipsla operation コマンド • type udp jittercommand • <i>count</i> 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	<ul style="list-style-type: none"> • <i>operation-number</i> 引数を指定した ipsla operation コマンド • type udp jittercommand • <i>size</i> 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	<ul style="list-style-type: none"> • <i>operation-number</i> 引数を指定した ipsla operation コマンド • type udp jittercommand • <i>interval</i> 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	<ul style="list-style-type: none"> • <i>operation-number</i> 引数を指定した ipsla operation コマンド • type udp jittercommand • <i>seconds</i> 引数を指定した frequency コマンド



- (注) IP SLA の設定中に **control disable** コマンドを使用して制御パケットを無効にした場合、送信元から送信されたパケットにはシーケンス番号がありません。ジッターを計算するには、シーケンス番号とタイムスタンプの値が必要です。したがって、**control disable** コマンドを使用する場合、ジッターは計算されません。

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 \(101 ページ\)](#)」のタスクを実行します。

送信元デバイスでの基本 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

ステップ 1 **configure**

ステップ 2 **ipsla operation *operation-number***

例：

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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 9 **exit**

例 :

```
RP/0/RP0/cpu 0: router(config-ipsla-udp-jitter)# exit
RP/0/RP0/cpu 0: router(config-ipsla-op)# exit
RP/0/RP0/cpu 0: router(config-ipsla)# exit
RP/0/RP0/cpu 0: router(config)#
```

IPSLA 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/RP0/cpu 0: router(config)# ipsla schedule operation 432
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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。
- （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 **commit**

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

ステップ 1 **configure**

ステップ 2 `ipsla operation operation-number`

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-udp-jitter)# statistics hourly
RP/0/RP0/cpu 0: 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。

ステップ 9 **buckets** *hours*

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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。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/RP0/cpu 0: 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 **exit**

例 :

```
RP/0/RP0/cpu 0: router(config-ipsla-op-stats)# exit
```

IP SLA 統計情報コンフィギュレーションモードを終了します。

ステップ 13 **datasize request** *size*

例 :

```
RP/0/RP0/cpu 0: 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 14 **timeout** *milliseconds*

例 :

```
RP/0/RP0/cpu 0: 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 15 **tos number**

例 :

```
RP/0/RP0/cpu 0: 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 16 **exit**

例 :

```
RP/0/RP0/cpu 0: router(config-ipsla-udp-jitter)# exit
RP/0/RP0/cpu 0: router(config-ipsla-op)# exit
RP/0/RP0/cpu 0: router(config-ipsla)# exit
RP/0/RP0/cpu 0: 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 17 **ipsla schedule operation op-num**

例 :

```
RP/0/RP0/cpu 0: router(config)# ipsla schedule operation 432
RP/0/RP0/cpu 0: 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 18 **life {forever | seconds}**

例 :

```
RP/0/RP0/cpu 0: 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 19 **ageout seconds**

例 :

```
RP/0/RP0/cpu 0: 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 20 **recurring**

例 :

```
RP/0/RP0/cpu 0: 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 21 **start-time** [*hh:mm:ss {day | month day}*] | **now** | **pending** | **after hh:mm:ss**]

例 :

```
RP/0/RP0/cpu 0: 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 22 **commit**

ステップ 23 **show ipsla statistics** [*operation-number*]

例 :

```
RP/0/RP0/cpu 0: router # show ipsla statistics 432
```

現在の統計情報を表示します。

ステップ 24 **show ipsla statistics aggregated** [*operation-number*]

例 :

```
RP/0/RP0/cpu 0: 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。
- 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 (101 ページ)」の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

ステップ 1 **configure**

ステップ 2 **ipsla operation operation-number**

例：

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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 **destination address ipv4address**

例：

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-udp-echo)# exit
RP/0/RP0/cpu 0: router(config-ipsla-op)# exit
RP/0/RP0/cpu 0: router(config-ipsla)# exit
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config)# ipsla schedule operation 432
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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 11 recurring

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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。これ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 commit

ステップ 14 show ipsla statistics [operation-number]

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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

ステップ 1 **configure**

ステップ 2 **ipsla operation operation-number**

例：

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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。

ステップ 10 **timeout milliseconds**

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-udp-echo)# exit
```

```
RP/0/RP0/cpu 0: router(config-ipsla-op)# exit
RP/0/RP0/cpu 0: router(config-ipsla)# exit
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config)# ipsla schedule operation 432
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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 17 `start-time [hh:mm:ss {day | month day} | now | pending | after hh:mm:ss]`

例：

```
RP/0/RP0/cpu 0: 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 **commit**

ステップ 19 **show ipsla statistics enhanced aggregated [operation-number] interval seconds**

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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と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 レスポンド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

ステップ 1 **configure**

ステップ 2 **ipsla operation operation-number**

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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 5 **frequency seconds**

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-icmp-echo)# exit
```

```
RP/0/RP0/cpu 0: router(config-ipsla-op)# exit
```

```
RP/0/RP0/cpu 0: router(config-ipsla)# exit
```

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config)# ipsla schedule operation 432
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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 **commit**

ステップ 13 **show ipsla statistics** [*operation-number*]

例：

```
RP/0/RP0/cpu 0: 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

ステップ 1 **configure**

ステップ 2 **ipsla operation operation-number**

例：

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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。

- (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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-icmp-echo)# exit
```

```
RP/0/RP0/cpu 0: router(config-ipsla-op)# exit
RP/0/RP0/cpu 0: router(config-ipsla)# exit
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config)# ipsla schedule operation 432
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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 15 **recurring**

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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指定します。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 **commit**

ステップ 18 **show ipsla statistics** [*operation-number*]

例 :

```
RP/0/RP0/cpu 0: 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

ステップ 1 **configure**

ステップ 2 **ipsla operation** *operation-number*

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-op)# type icmp path-echo
RP/0/RP0/cpu 0: 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 destination address ipv4address

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-icmp-path-echo)# exit
RP/0/RP0/cpu 0: router(config-ipsla-op)# exit
RP/0/RP0/cpu 0: router(config-ipsla)# exit
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config)# ipsla schedule operation 432
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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 **commit**

ステップ 13 **show ipsla statistics [operation-number]**

例：

```
RP/0/RP0/cpu 0: 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

ステップ 1 **configure**

ステップ 2 **ipsla operation operation-number**

例：

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-op)# type icmp path-echo  
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-icmp-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/RP0/cpu 0: router(config-ipsla-icmp-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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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。

ステップ 9 ***tos number***

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-icmp-path-echo)# exit
RP/0/RP0/cpu 0: router(config-ipsla-op)# exit
RP/0/RP0/cpu 0: router(config-ipsla)# exit
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config)# ipsla schedule operation 432
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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 16 **ageout *seconds***

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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 **commit**

ステップ 20 **show ipsla statistics [operation-number]**

例 :

```
RP/0/RP0/cpu 0: 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 23: 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	<ul style="list-style-type: none"> • <code>operation-number</code> 引数を指定した ipsla operation コマンド • <code>count</code> 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 ミリ秒	<ul style="list-style-type: none"> • <code>operation-number</code> 引数を指定した ipsla operation コマンド • <code>interval</code> 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	<ul style="list-style-type: none"> • <code>operation-number</code> 引数を指定した ipsla operation コマンド • <code>seconds</code> 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

ステップ 1 **configure**

ステップ 2 **ipsla operation operation-number**

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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 `destination address ipv4address`

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-icmp-path-jitter)# exit
RP/0/RP0/cpu 0: router(config-ipsla-op)# exit
RP/0/RP0/cpu 0: router(config-ipsla)# exit
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config)# ipsla schedule operation 432
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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 11 **ageout** *seconds*

例：

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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 **commit**

ステップ 15 **show ipsla statistics** [*operation-number*]

例 :

```
RP/0/RP0/cpu 0: 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

ステップ 1 **configure****ステップ 2** **ipsla operation** *operation-number*

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-icmp-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/RP0/cpu 0: router(config-ipsla-icmp-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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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10 datasize request *size*

例：

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-icmp-path-jitter)# exit
RP/0/RP0/cpu 0: router(config-ipsla-op)# exit
RP/0/RP0/cpu 0: router(config-ipsla)# exit
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config)# ipsla schedule operation 432
RP/0/RP0/cpu 0: 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 16 **life** {**forever** | *seconds*}

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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 commit

ステップ 21 show ipsla statistics [operation-number]

例 :

```
RP/0/RP0/cpu 0: 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

ステップ 1 **configure**

ステップ 2 **ipsla operation operation-number**

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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。

ステップ 4 **output interface** *type interface-path-id*

例：

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-mpls-lsp-ping)# target ipv4 10.25.26.10 255.255.255.255
```

または

```
RP/0/RP0/cpu 0: router(config-ipsla-mpls-lsp-ping)# target ipv4 10.25.26.10/32
```

または

```
RP/0/RP0/cpu 0: router(config-ipsla-mpls-lsp-ping)# target traffic-eng tunnel 12
```

または

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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。

ステップ 9 `reply mode {control-channel | router-alert}`

例 :

```
RP/0/RP0/cpu 0: router(config-ipsla-mpls-lsp-ping)# reply mode router-alert
```

または

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-mpls-lsp-ping)# exit
RP/0/RP0/cpu 0: router(config-ipsla-op)# exit
```

```
RP/0/RP0/cpu 0: router(config-ipsla)# exit
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config)# ipsla schedule operation 432
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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 15 **commit**

ステップ 16 **show ipsla statistics [operation-number]**

例：

```
RP/0/RP0/cpu 0: 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

ステップ 1 **configure**

ステップ 2 **ipsla operation operation-number**

例 :

```
RP/0/RP0/cpu 0: 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 trace**

例 :

```
RP/0/RP0/cpu 0: 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/cpu 0: 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/RP0/cpu 0: router(config-ipsla-mpls-lsp-trace)# target ipv4 10.25.26.10
255.255.255.255
```

```
RP/0/RP0/cpu 0: router(config-ipsla-mpls-lsp-trace)# target ipv4 10.25.26.10/32
```

または

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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。

ステップ 9 **reply mode router-alert**

例：

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-mpls-lsp-trace)# exit
RP/0/RP0/cpu 0: router(config-ipsla-op)# exit
RP/0/RP0/cpu 0: router(config-ipsla)# exit
RP/0/RP0/cpu 0: 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//CPU0:router(config)# ipsla schedule operation 432
RP/0//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/RP0/cpu 0: 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 15 **commit**

ステップ 16 **show ipsla statistics** [*operation-number*]

例 :

```
RP/0/RP0/cpu 0: 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

ステップ 1 **configure**

ステップ 2 **ipsla reaction operation** *operation-number*

例 :

```
RP/0/RP0/cpu 0: 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**]

例 :

```
RP/0/RP0/cpu 0: router(config-ipsla-react)# react connection-loss
```

```
RP/0/RP0/cpu 0: 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 commit

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

ステップ 1 configure

ステップ 2 ipsla reaction operation operation-number

例：

```
RP/0/RP0/cpu 0: 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 [jitter-average {dest-to-source | source-to-dest}]

例：

```
RP/0/RP0/cpu 0: router(config-ipsla-react)# react jitter-average  
RP/0/RP0/cpu 0: 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 commit

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

ラウンドトリップ違反のトリガーの設定

ができます。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

ステップ 1 configure

ステップ 2 `ipsla reaction operation operation-number`

例：

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-react)# react packet-loss dest-to-source
RP/0/RP0/cpu 0: 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 commit

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

ステップ 1 configure

ステップ 2 `ipsla reaction operation operation-number`

例：

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-react)# react rtt
RP/0/RP0/cpu 0: router(config-ipsla-react-cond)#
```

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。

ステップ 4 commit

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

ステップ 1 configure

ステップ 2 ipsla reaction operation *operation-number*

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-react)# react timeout
RP/0/RP0/cpu 0: 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 commit

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

ステップ1 configure

ステップ2 `ipsla reaction operation operation-number`

例：

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-react)# react verify-error
```

```
RP/0/RP0/cpu 0: 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 commit

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 24: 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
連続	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 (150 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 (149 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 (151 ページ) 」を参照してください。
平均	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 (152 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

ステップ 1 configure

ステップ 2 ipsla reaction operation operation-number

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-react)# react timeout
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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 **commit**

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

ステップ 1 **configure**

ステップ 2 **ipsla reaction operation operation-number**

例：

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-react)# react connection-loss
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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 commit

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** キーワードを指定して使用します。

手順

ステップ 1 configure

ステップ 2 ipsla reaction operation operation-number

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-react)# react rtt  
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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 commit

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

ステップ 1 configure

ステップ 2 `ipsla reaction operation operation-number`

例：

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-react)# react packet-loss dest-to-source
RP/0/RP0/cpu 0: 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 `threshold type average number-of-probes`

例：

```
RP/0/RP0/cpu 0: 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 commit

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

ステップ 1 **configure**

ステップ 2 **ipsla reaction operation operation-number**

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla-react)# react connection-loss  
RP/0/RP0/cpu 0: 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。

ステップ 4 **action [logging | trigger]**

例 :

```
RP/0/RP0/cpu 0: 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めます。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されます。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 commit

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

始める前に



(注) MPLS LSP モニタリングは PE ルータで設定されます。

手順

ステップ 1 **configure**

ステップ 2 **ipsla**

例 :

```
RP/0/RP0/cpu 0: 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 discovery vpn**

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla)# mpls lsp-monitor
RP/0/RP0/cpu 0: router(config-ipsla-mplsmlm)#
```

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/RP0/cpu 0: router(config-ipsla-mplsmlm)# monitor 1
RP/0/RP0/cpu 0: router(config-ipsla-mplsmlm-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/RP0/cpu 0: router(config-ipsla-mplsmlm-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/RP0/cpu 0: router(config-ipsla-mplsmlm-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。

ステップ 10 **scan interval scan-interval**

例 :

```
RP/0/RP0/cpu 0: router(config-ipsla-mplsmlm-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/RP0/cpu 0: router(config-ipsla-mplsml-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/RP0/cpu 0: router(config-ipsla-mplsml-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/RP0/cpu 0: router(config-ipsla-mplsml-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/RP0/cpu 0: router(config-ipsla-mplsml-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。

ステップ 15 **force explicit-null**

例 :

```
RP/0/RP0/cpu 0: router(config-ipsla-mpls-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。デフォルトでは、無効になっています。

ステップ 16 **reply dscp** *dscp-bits*

例 :

```
RP/0/RP0/cpu 0: router(config-ipsla-mpls-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/RP0/cpu 0: router(config-ipsla-mpls-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。デフォルトでは、無効になっています。

ステップ 18 **ttl** *time-to-live*

例 :

```
RP/0/RP0/cpu 0: router(config-ipsla-mpls-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/RP0/cpu 0: router(config-ipsla-mpls-lsp-ping)# tag mpls-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/RP0/cpu 0: router(config-ipsla-mpls-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/RP0/cpu 0: router(config-ipsla-mpls-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 commit

次のタスク

- 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

始める前に



(注) MPLS LSP モニタリングは PE ルータで設定されます。

手順

ステップ 1 configure

ステップ 2 ipsla

例：

```
RP/0/RP0/cpu 0: 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 discovery vpn

例：

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla)# mpls lsp-monitor  
RP/0/RP0/cpu 0: router(config-ipsla-mplsmlm)#
```

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/RP0/cpu 0: router(config-ipsla-mplsmlm)# monitor 1  
RP/0/RP0/cpu 0: router(config-ipsla-mplsmlm-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/RP0/cpu 0: router(config-ipsla-mplsmlm-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/RP0/cpu 0: router(config-ipsla-mplsmlm-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。

ステップ 10 **scan interval scan-interval**

例：

```
RP/0/RP0/cpu 0: router(config-ipsla-mplsmlm-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/RP0/cpu 0: router(config-ipsla-mpls-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/RP0/cpu 0: router(config-ipsla-mpls-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/RP0/cpu 0: router(config-ipsla-mpls-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。

ステップ 14 **force explicit-null**

例：

```
RP/0/RP0/cpu 0: router(config-ipsla-mpls-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。デフォルトでは、無効になっています。

ステップ 15 **reply dscp** *dscp-bits*

例：

```
RP/0/RP0/cpu 0: router(config-ipsla-mpls-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/RP0/cpu 0: router(config-ipsla-mplsml-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。デフォルトでは、無効になっています。

ステップ 17 **ttl time-to-live**

例 :

```
RP/0//CPU0:router(config-ipsla-mplsml-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/RP0/cpu 0: router(config-ipsla-mplsml-lsp-trace)# tag mplsml-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/RP0/cpu 0: router(config-ipsla-mplsml-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/RP0/cpu 0: router(config-ipsla-mplsml-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 **commit**

次のタスク

- 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。

始める前に

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

ステップ 1 **configure**

ステップ 2 **ipsla**

例：

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla)# mpls lsp-monitor  
RP/0/RP0/cpu 0: router(config-ipsla-mplsmlm)#
```

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 **reaction monitor *monitor-id***

例：

```
RP/0/RP0/cpu 0: router(config-ipsla-mplsmlm)# reaction monitor 2  
RP/0/RP0/cpu 0: router(config-ipsla-mplsmlm-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/RP0/cpu 0: router(config-ipsla-mplsmlm-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/RP0/cpu 0: router(config-ipsla-mpls-lm-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/RP0/cpu 0: router(config-ipsla-mpls-lm-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 commit

次のタスク

- 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

ステップ 1 configure**ステップ 2 ipsla**

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: router(config-ipsla)# mpls lsp-monitor
RP/0/RP0/cpu 0: router(config-ipsla-mpls-lm)#
```

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/RP0/cpu 0: router(config-ipsla-mplsmlm)# schedule monitor 2
RP/0/RP0/cpu 0: router(config-ipsla-mplsmlm-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/RP0/cpu 0: router(config-ipsla-mplsmlm-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/RP0/cpu 0: router(config-ipsla-mplsmlm-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。

ステップ 7 `start-time hh:mm:ss [day | month day]`

例：

```
RP/0/RP0/cpu 0: router(config-ipsla-mplsmlm-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 `commit`

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

ステップ 1 **configure**

ステップ 2 **ipsla**

例 :

```
RP/0/RP0/cpu 0: 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/RP0/cpu 0: 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。

ステップ 4 **monitor monitor-id**

例 :

```
RP/0/RP0/cpu 0: router(config-ipsla-mplsmlm)# 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/RP0/cpu 0: router(config-ipsla-mplsmlm-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/RP0/cpu 0: router(config-ipsla-mplsmlm-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/RP0/cpu 0: router(config-ipsla-mplsml-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/RP0/cpu 0: router(config-ipsla-mplsml-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/RP0/cpu 0: router(config-ipsla-mplsml-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/RP0/cpu 0: router(config-ipsla-mplsml-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/RP0/cpu 0: router(config-ipsla-mplsml-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。

ステップ 12 **path retry range**

例：

```
RP/0/RP0/cpu 0: router(config-ipsla-mplsml-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/RP0/cpu 0: router(config-ipsla-mplsml-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/RP0/cpu 0: router(config-ipsla-mpls-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 commit

IP サービス レベル契約を実装するための設定例

ここでは、次の設定例を示します。

IP サービス レベル契約の設定：例

次の例では、UDP ジッター動作を設定およびスケジューリングする方法を示します。

```
configure
ipsla
operation 101
type udp jitter
destination address 12.2.0.2
statistics hourly
buckets 5
distribution count 5
distribution interval 1
!
destination port 400
statistics interval 120
buckets 5
!
!
!
schedule operation 101
start-time now
life forever
!
!

show ipsla statistics
```

```

Fri Nov 28 16:48:48.286 GMT
Entry number: 101
Modification time: 16:39:36.608 GMT Fri Nov 28 2014
Start time       : 16:39:36.633 GMT Fri Nov 28 2014
Number of operations attempted: 10
Number of operations skipped  : 0
Current seconds left in Life  : Forever
Operational state of entry    : Active
Operational frequency(seconds): 60
Connection loss occurred      : FALSE
Timeout occurred              : FALSE
Latest RTT (milliseconds)     : 3
Latest operation start time    : 16:48:37.653 GMT Fri Nov 28 2014
Next operation start time     : 16:49:37.653 GMT Fri Nov 28 2014
Latest operation return code  : OK
RTT Values:
  RTTAvg  : 3          RTTMin: 3          RTTMax : 4
  NumOfRTT: 10        RTTSum: 33         RTTSum2: 111
Packet Loss Values:
  PacketLossSD      : 0          PacketLossDS : 0
  PacketOutOfSequence: 0        PacketMIA    : 0
  PacketLateArrival : 0          PacketSkipped: 0
  Errors            : 0          Busies       : 0
  InvalidTimestamp  : 0
Jitter Values :
  MinOfPositivesSD: 1          MaxOfPositivesSD: 1
  NumOfPositivesSD: 2          SumOfPositivesSD: 2
  Sum2PositivesSD : 2
  MinOfNegativesSD: 1          MaxOfNegativesSD: 1
  NumOfNegativesSD: 1          SumOfNegativesSD: 1
  Sum2NegativesSD : 1
  MinOfPositivesDS: 1          MaxOfPositivesDS: 1
  NumOfPositivesDS: 1          SumOfPositivesDS: 1
  Sum2PositivesDS : 1
  MinOfNegativesDS: 1          MaxOfNegativesDS: 1
  NumOfNegativesDS: 1          SumOfNegativesDS: 1
  Sum2NegativesDS : 1
  JitterAve: 1          JitterSDAve: 1          JitterDSAve: 1
  Interarrival jitterout: 0          Interarrival jitterin: 0
One Way Values :
  NumOfOW: 0
  OWMinSD : 0          OWMaxSD: 0          OWSumSD: 0
  OWSum2SD: 0          OWAveSD: 0
  OWMinDS : 0          OWMaxDS: 0          OWSumDS: 0
  OWSum2DS: 0          OWAveDS: 0

```

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。次の操作を実行できます。

- `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
```