Embedded Syslog Manager（ESM）

Embedded Syslog Manager（ESM）機能は、システム メッセージ ロガーによって伝送される前にシステム ロギング メッセージをフィルタリング、拡大、相互関連付け、ルーティング、カスタマイズできるようにするプログラマブル フレームワークを提供します。

- 機能情報の確認，1 ページ
- Embedded Syslog Manager の制約事項，2 ページ
- Embedded Syslog Manager について，2 ページ
- Embedded Syslog Manager の使用方法，4 ページ
- Embedded Syslog Manager の設定例，12 ページ
- Embedded Syslog Manager に関する追加情報，20 ページ
- Embedded Syslog Manager の機能情報，22 ページ
- 用語集，23 ページ

機能情報の確認

ご使用のソフトウェア リリースでは、このモジュールで説明されるすべての機能がサポートされているとは限りません。 最新の機能情報および警告については、Bug Search Tool およびご使用のプラットフォームおよびソフトウェア リリースのリリース ノートを参照してください。 このモジュールに記載されている機能の詳細を検索し、各機能がサポートされているリリースのリストを確認する場合は、このモジュールの最後にある機能情報の表を参照してください。

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

Embedded Syslog Manager の制約事項

Embedded Syslog Manager（ESM）は Tcl 8.3.4 サブシステムに依存します。これは、ESM フィルタが Tool Command Language（Tcl）で作成されるためです。 ESM は、Tcl バージョン 8.3.4 以降をサポートするイメージでのみ使用可能です。 Tcl 8.3.4 サポートが追加されるかどうかはリリースによって異なります。

ESM フィルタは Tcl で書かれています。

ESM フィルタリングは、SNMP「履歴」ロギングには適用できません。 したがって、**logginghistory** および **snmp-serverenabletrapssyslog** コマンドを使ってロギングされるメッセージには ESM フィルタリングが適用されません。

Embedded Syslog Manager について

System Message Logging

Embedded Syslog Manager を導入すると、システム メッセージを標準メッセージ、XML 形式のメッセージ、または ESM でフィルタリングされたメッセージとして、別個にロギングできるようになります。 これらの出力は、従来のあらゆる syslog ターゲットに送信できます。 たとえば、コンソール接続への標準ロギング、XML 形式のメッセージのバッファへのロギング、および ESM でフィルタリングされたメッセージのモニタへのロギングをイネーブルにできます。 同様に、各タイプの出力は異なるリモート ホストに送信できます。 別個のロギング プロセスの利点は、たとえば ESM フィルタ モジュールに問題がある場合に標準のロギングが影響を受けないことです。

システム ロギング メッセージの形式

システム ロギング メッセージは、次の形式で表示されます。

%<ファシリティ>-<重大度>-<ニーモニック>: <メッセージ テキスト>

以下に、システム ロギング メッセージの例を示します。

```
%LINK-5-CHANGED: Interface Serial3/3, changed state to administratively down
```

通常は、これらのメッセージの前にエラー シーケンス番号やタイムスタンプなどの追加のテキストが存在します。

<シーケンス番号>: <タイム スタンプ>:%<ファシリティ>-<重大度>-<ニーモニック>: <メッセージ テキスト>

エラー シーケンス番号とタイムスタンプの後に続くシステム ロギング メッセージの例を以下に示します。

```
000013: Mar 18 14:52:10.039:%LINK-5-CHANGED: Interface Serial3/3, changed state
```

```
to administratively down
```

（注） システム ロギング メッセージで使用されるタイム スタンプの形式は、**servicetimestamps** グローバル コンフィギュレーション モード コマンドによって決まります。
servicesequence-numbers グローバル コンフィギュレーション コマンドは、先頭のシーケンス番号をイネーブルまたはディセーブルにします。 時刻の前のアスタリスク（*）は、システム クロックが信頼できる時刻源と同期していないため、時刻が不正確である可能性があることを示しています。

Embedded Syslog Manager の利点

Cisco ソフトウェアの組み込み機能である Embedded Syslog Manager（ESM）を使用すると、送信元でのシステム メッセージ ロギングの完全な制御が可能になります。 ESM に備わっているプログラマティック インターフェイスを使用すると、システム ロギングに関連する特定のニーズを満たすカスタム フィルタを作成できます。 この機能の利点は次のとおりです。

- カスタマイズ：システム ロギング メッセージを扱う完全にカスタマイズ可能な処理。相互に連携する複数の syslog コレクタをサポートします。
- 主要なメッセージの重大度のエスカレーション：システム定義の重大度レベルを使用する代わりに、syslog メッセージに関する独自の重大度レベルを設定できます。
- 特定のメッセージに対象を絞る：ファシリティのタイプや重大度のタイプに基づいて、特定のメッセージまたはメッセージ タイプをさまざまな syslog コレクタにルーティングできます。
- SMTP ベースの E メール アラート：TCP ベースの syslog コレクタまたはシンプル メール転送プロトコル（SMTP）サーバなどの外部サーバに TCP を使用して通知する機能。
- メッセージの制限：デバイスレベルのイベントを関連付けることにより、syslog の「メッセージ ストーム」を制限し、管理することができます。

ESM は UDP ベースの syslog メカニズムに代わるものではなく、現在のシステム ロギング プロセスと並行して動作可能なオプションのサブシステムです。 たとえば、元の syslog メッセージ ストリームをサーバ A によって収集し続けるのと同時に、フィルタリング、相互関連、その他の方法でカスタマイズされた ESM ロギング ストリームをサーバ B に送信することができます。元の syslog ストリームまたは ESM ストリームのいずれかを受信するよう、syslog メッセージの現在のすべてのターゲット（コンソール、モニタ、バッファ、および syslog ホスト リスト）を設定することができます。 ESM ストリームは、それに応じて、ユーザ定義のストリームにさらに分割し、コレクタにルーティングできます。

syslog フィルタ モジュール

Embedded Syslog Manager（ESM）はsyslogフィルタモジュールを使用してシステム ロギング メッセージを処理します。 syslog フィルタ モジュールは、ローカル システム メモリまたはリモート ファイル サーバに保存される Tool Command Language（Tcl）で書かれたスクリプトです。 ESM は、独自のスクリプトを書いて参照できるため、カスタマイズ可能です。

syslog フィルタ モジュールは、プレーンテキスト ファイルまたはコンパイル済みファイルとして書いて、保存できます。 Tcl スクリプトの事前コンパイルは、TclPro などのツールを使用して実行できます。 コンパイル済みスクリプトは編集できないため、セキュリティおよび管理された一貫性の尺度になります。

（注） Tcl スクリプト モジュールには実行可能コマンドが含まれているため、コンフィギュレーション ファイルを管理するのと同じ方法で、これらのファイルのセキュリティを管理する必要があります。

Embedded Syslog Manager の使用方法

ESM syslog フィルタ モジュールの書き込み

Embedded Syslog Manager（ESM）設定で syslog フィルタ モジュールを参照する前に、システム ロギング メッセージへの適用対象となるモジュールを書き込むか、取得する必要があります。 syslog フィルタ モジュールは、ローカル システム メモリまたはリモート ファイル サーバに保存できます。 syslog フィルタ モジュールを書き込む前に、次の概念を理解しておく必要があります。

ESM フィルタ プロセス

ESM がイネーブルの場合、すべてのシステム ロギング メッセージは、参照された syslog フィルタ モジュールを通して処理されます。 syslog フィルタ モジュールは、フィルタ チェーンで順番に処理されます。 フィルタ チェーン内の syslog フィルタ モジュールの位置は、**loggingfilter** グローバル コンフィギュレーション モード コマンドで適用された位置タグによって決定されます。 位置が指定されていない場合、モジュールは、設定に追加された順番で処理されます。

各フィルタ モジュールの出力は、チェーン内の次のフィルタ モジュールの入力として使用されます。 したがって、元の syslog メッセージ（::orig_msg）を含む Tcl グローバル変数は、チェーン内の次のフィルタを呼び出す前に、各フィルタの戻り値に設定されます。 したがって、フィルタが NULL を戻した場合、メッセージは ESM ストリームに送信されません。 すべてのフィルタがメッセージを処理した後、メッセージは、ロガーによる配信のためのキューに入れられます。

コンソール、バッファ、モニタ、および syslog ホストは、特定のメッセージ ストリーム（通常、XML、または ESM）を受信するように設定できます。 syslog ホストは、ユーザ定義の番号が付い

たストリームを受信するように、さらに限定できます。 各ターゲットは各メッセージを検査し、そのストリーム タグに基づいてメッセージを受け入れるか、または拒否します。 ESM フィルタは、Tcl グローバル変数「::stream」を変更することによってメッセージのストリーム タグを変更し、宛先ストリームを変更できます。

syslog フィルタ モジュールの入力

Embedded Syslog Manager（ESM）が有効になっている場合、システム ロギング メッセージがロギング プロセスに送信されます。 システム ロギング メッセージの中、およびフォーマットされた syslog メッセージ全体に含まれる各データ エレメントが、Tcl グローバル変数として記録されます。 syslog メッセージのデータ エレメントの形式は、次のとおりです。

<シーケンス番号>: <タイム スタンプ>:%<ファシリティ>-<重大度>-<ニーモニック>: <メッセージ テキスト>

メッセージ テキストには、しばしばメッセージ引数が含まれます。

また、メッセージが syslog ホストで受信されるときに「syslog カウント」番号が追加されます。

<syslog カウント>: <シーケンス番号>: <タイム スタンプ>:%<ファシリティ>-<重大度>-<ニーモニック>: <メッセージ テキスト>

次の例は、シーケンスの最初に含まれる syslog カウント番号を示しています。

以下の表に、syslog フィルタ モジュールで使用される Tcl スクリプト入力変数を示します。 フィルタで扱う必要のある syslog メッセージ データは、Tcl グローバル名前空間変数として渡されます。 したがって、スクリプトモジュール内で変数の前に二重のコロンを付ける必要があります。

標準的な ESM フィルタ処理

システム ロギング メッセージが生成されるたびに、syslog フィルタ モジュールがシリーズで呼び出されます。 このシリーズは ::module_position 変数によって決定されますが、これは一般的にシステム設定内でモジュールが参照される順番（モジュールが設定される順番）です。

あるフィルタ モジュールの出力が、次のフィルタ モジュールへの入力になります。 フィルタへの入力は Tcl グローバル名前空間変数であるため、それぞれのフィルタは、フィルタの目的に応じてこれらの変数のいずれかまたはすべてを変更できます。

後続の複数のフィルタ実行の間で Embedded Syslog Manager（ESM）フレームワークによって自動的に更新される Tcl グローバル変数は、::orig_msg 変数と ::cli_args 変数だけです。 フレームワークは、自動的に ::orig_msg の値をフィルタ モジュールの戻り値に設定します。 したがって、元のメッセージを変更するかフィルタリングするように設定されているフィルタでは、::orig_msg 変数の値を手動で設定しないでください。フィルタは目的の値を返すことだけが必要です。 たとえば、次の 1 行の ESM フィルタ

```
return “This is my new syslog message.”
```

は、受け取ったメッセージをすべて無視して出力を「This is my new syslog message」という定数文字列に常に変更します。 このモジュールがチェーン内の最後のフィルタである場合、すべての ESM ターゲットはこの文字列を最終的な syslog メッセージとして受け取ります。

1 行の ESM フィルタは、

```
return “”
```

ESM ストリームへのすべての syslog メッセージをブロックします。 たとえば、

```
return $::orig_msg
```

という行は、メッセージをチェーンの次のフィルタに渡すだけです。 したがって、不要なメッセージを抑制するように設定されている ESM フィルタは次のように見えます。

```
if { [my_procedure_to_check_this_message] == 1 } {
     return $::orig_msg
} else {
     return “”
}
```

その設計によって、一部のフィルタは ::orig_msg 変数をまったく使用せず、そのデータ エレメントから syslog メッセージを再構築する（::format_string、::msg_args、::timestamp などを使用）場合があります。 たとえば、XML タグ付きフィルタは個々のデータ エレメントにタグを付け、元のフォーマットされたメッセージを無視します。 このようなモジュールが Tcl スクリプトの最初で ::orig_msg 変数を検査することは重要です。以前のフィルタで「メッセージを送信してはならない」と指示されている場合（つまり ::orig_msg が NULL の場合）、メッセージは処理されず、NULL を返します。

また、**exec** および **config** Tcl コマンドを使用して syslog フィルタ モジュールにコマンドを追加することもできます。 たとえば、syslog メッセージに送信元 IP アドレスを追加する必要があり、しかも（**logging source-interface**コマンドを使って）イーサネット 2/0 インターフェイスから送信されるよう syslog メッセージが設定される場合には、スクリプト内で **exec** Tcl コマンドを次のように使用することで、モジュールの初期化中に **show interface Ethernet 2/0** コマンドを発行できます。

```
set source_ip_string [exec show ip int E2/0 | inc Internet]
puts $source_ip_string
"  Internet address is 10.4.2.63/24"
```

バックグラウンド ESM フィルタの処理

Tcl では、後の時点で処理するためにコマンドをキューに入れることができます。それを行うには **after** Tcl コマンドを使用します。 このコマンドの最も一般的な使用方法は、一定の時間間隔（「相関期間」と呼ばれる）にわたって、複数のイベントを相互に関連付ける（収集して要約する）ことです。 該当する期間が満了すると、フィルタは「ウェイクアップ」し、その期間に発生したイベントを計算または要約する必要があります。また、多くの場合、イベント報告用の新しい syslog メッセージを送信する必要があります。 このバックグラウンド プロセスは、特定の時間の経過後、Tcl インタープリタがキューに入れられたコマンドを実行できるようにする ESM イベント ループ プロセスによって処理されます。

syslog フィルタ モジュールが相関期間を利用する必要がある場合、相関期間が満了した時点で **after** Tcl コマンドを使用して要約手順を呼び出す必要があります（「Embedded Syslog Manager の設定例」セクションにある例を参照）。 バックグラウンド プロセスの実行中は通常のフィルタ

チェーンの処理が行われないため、出力を生成するために、これらのフィルタで 2 つの ESM Tcl 拡張機能（**errmsg** または **esm_errmsg**）のいずれかを使用する必要があります。

バックグラウンド処理中に、**after** コマンドによってキューに入れられたコマンドは、（通常の処理のように）フィルタ チェーンのコンテキストでは実行されず、Tcl インタープリタによって連続して実行される自律的な手順となります。 したがって、これらのバックグラウンド手順では通常の Tcl グローバル名前空間変数を扱うべきではありません（ただし **esm_errmsg** の使用時に次のフィルタ用にグローバル名前空間変数を設定する場合を除く）。代わりに、独自の名前空間に保存される変数を扱う必要があります。 これらの変数が手順の定義の外側で宣言される場合、すべてのコールで永続的です。

errmsg Tcl コマンドの目的は、新しいメッセージを作成して配信用に送り出し、その際に他の syslog フィルタ モジュールをすべて回避することです。 **errmsg** コマンドの構文は次のとおりです。

```
errmsg <severity> <stream> <message_string>
```

esm_errmsg Tcl コマンドの目的は、新しいメッセージを作成し、フィルタ チェーン内のそれ以降にある syslog フィルタ モジュールでメッセージを処理して、配信用に送り出すことです。 **esm_errmsg** コマンドの構文は次のとおりです。

```
esm_errmsg <module_position>
```

errmsg() Tcl 関数と esm_errmsg() Tcl 関数の主な相違点として、**errmsg** はフィルタを無視してメッセージを配信用にキューに直接入れます。一方、**esm_errmsg** はフィルタ チェーンの後続部分に syslog メッセージを送ります。

次の例では、新しい syslog メッセージが作成され、アラート重大度 1 のタグを付けられて、設定された ESM ロギング ターゲット（ストリーム 2）に送信されます。 このフィルタの目的は、30 分の相関期間中に個々の SYS-5-CONFIG メッセージを抑制し、ウィンドウ終了時に要約メッセージを送信することです。

```
errmsg 1 2 "*Jan 24 09:34:02.539: %SYS-1-CONFIG_I: There have been 12
configuration changes to the router between Jan 24 09:04:02.539 and Jan 24
09:34:01.324"
```

（それ以降にある残りのフィルタを呼び出す）**esm_errmsg** を使用するために、このバックグラウンド プロセスは **esm_errmsg** を呼び出す前に、必要な Tool Command Language（Tcl）グローバル名前空間変数を設定する必要があります。 ::module_position を渡すと、開始するフィルタが ESM フレームワークに通知されます。 したがって、**esm_errmsg** コマンドを使用するフィルタは、バックグラウンド処理用に、（通常の処理中にグローバル名前空間変数で渡される）::module_position を独自の名前空間変数に保存する必要があります。 次に例を示します。

```
proc ::my_filter_namespace::my_summary_procedure{}
{
 set ::orig_msg "*Jan 24 09:34:02.539: %SYS-1-CONFIG_I: There have been 12
configuration changes to the router between Jan 24 09:04:02.539 and Jan 24
09:34:01.324"
 set ::timestamp "*Jan 24 09:34:02.539"
 set ::severity 1
 set ::stream 2
 set ::traceback ""
 set ::pid ""
set ::process ""
set ::format_string "There have been %d configuration changes to the router
between %s and %s"
set ::msg_args {12 "Jan 24 09:04:01.539" "Jan 24 09:34:01.324"}
```

```
esm_errmsg $::my_filter_namespace::my_module_position
}
```

esm_errmsg コマンド用にすべてのグローバル名前空間変数を設定することの利点は、フィルタがモジュール式であり、ESM フレームワークでのフィルタの使用順序が重要でないことです。 たとえば、ESM を宛先とするすべてのメッセージの後ろにメッセージ発信者のホスト名を付ける必要がある場合、次のように 1 行の「hostname」フィルタを作成し、フィルタ チェーンの最後に配置できます。

```
return "$::orig_msg -- $::hostname"
```

この例では、バックグラウンド処理中にいずれかのフィルタが新しいメッセージを生成し、**errmsg** ではなく **esm_errmsg** を使用する場合、これらのメッセージの後ろに明確にホスト名が付けられます。

次の作業

syslog フィルタ モジュールを作成した後、デバイスからアクセス可能な場所にファイルを保存する必要があります。 ファイルは、ローカル システム メモリにコピーするか、ネットワーク ファイル サーバに保存できます。

Embedded Syslog Manager の設定

Embedded Syslog Manager（ESM）を設定するには、生成された syslog メッセージに適用する 1 つまたは複数のフィルタを指定し、syslog メッセージのターゲットを指定します。

はじめる前に

デバイスで 1 つまたは複数の syslog フィルタ モジュールを使用できなければなりません。

手順の概要

1. イネーブル化
2. **configureterminal**
3. **loggingfilter***filter-url* [*position*] [**args***filter-arguments*]
4. システム ロギング出力に適用する必要がある各 syslog フィルタ モジュールに対して、ステップ 3 を繰り返します。
5. 次のいずれか 1 つを入力します。
   - **logging** [**console** | **buffered** | **monitor**] **filtered** [*security-level*]
   - または
   - **logginghost** {*ip-address* | *hostname*} **filtered** [**stream***stream-id*]
6. 希望する各システム ロギング宛先に対して、ステップ 5 を繰り返します。
7. **loggingsource-interface***typenumber*
8. **loggingorigin-id** {**hostname** | **ip** | **ipv6** |**string***user-defined-id*}
9. **end**
10. **showlogging**

手順の詳細

	コマンドまたはアクション	目的
ステップ 1	イネーブル化 例： `Device> enable`	特権 EXEC モードをイネーブルにします。 • パスワードを入力します（要求された場合）。
ステップ 2	**configureterminal** 例： `Device# configure terminal`	グローバル コンフィギュレーション モードを開始します。
ステップ 3	**loggingfilter***filter-url* [*position*] [**args***filter-arguments*] 例： `Device(config)# logging filter slot0:/escalate.tcl 1 args CONFIG_I 1`	生成されたシステム ロギング メッセージに適用する 1 つまたは複数の syslog フィルタ モジュールを指定します。 • 使用する必要がある各 syslog フィルタ モジュールに対して、このコマンドを繰り返します。 • *filter-url* 引数は、syslog フィルタ モジュール（スクリプト）を示す Cisco IOS ファイル システムの場所です。 場所としてローカル メモリまたはリモート サーバが可能です（**tftp:**、**ftp:**、または **rcp:**を使用）。

	コマンドまたはアクション	目的
		• オプションの *position* 引数は、syslog フィルタ モジュールを実行する順序を指定します。 この引数を省略した場合、指定されたモジュールは、チェーンの最後のモジュールとして配置されます。 • 再び **loggingfilter** コマンドを入力して異なる位置（position）を指定することにより、フィルタの順序をすばやく再設定できます。 • オプションの **args***filter-arguments* 構文を追加すると、指定されたフィルタに引数を渡すことができます。 複数の引数を指定できます。 引数の数とタイプは、syslog フィルタ モジュールで定義する必要があります。 たとえば、特定の E メール アドレスを引数として受け入れるように syslog フィルタ モジュールが設計されている場合、**argsuser@host.com** 構文を使用して E メール アドレスを渡すことができます。 複数の引数は、通常、スペースで区切ります。 • 実行するモジュールのリストからモジュールを削除するには、このコマンドの **no** 形式を使用します。
ステップ 4	システム ロギング出力に適用する必要がある各 syslog フィルタ モジュールに対して、ステップ 3 を繰り返します。	--

ステップ 5	次のいずれか1つを入力します。
	• **logging** [**console** | **buffered** | **monitor**] **filtered** [*security-level*]
	• または
	• **logginghost** {*ip-address* | *hostname*} **filtered** [**stream***stream-id*]
	
	例：
	`Device(config)# logging console filtered informational`
	
	例：
	`Device(config)# logging host 209.165.200.225 filtered stream 20`
		ESM のフィルタリングされた syslog 出力のターゲットを指定します。
		• ESM フィルタリングされた syslog メッセージの送信先として、コンソール、モニタ（TTY および Telnet 接続）、システム バッファ、またはリモート ホストが可能です。
		• オプションの *level* 引数は、メッセージの送信を、指定された値またはそれより低い値のメッセージに限定します。 たとえば、レベル **1** が指定されている場合、レベル 1（アラート）またはレベル 0（緊急事態）のメッセージだけが、指定されたターゲットに送信されます。 レベルはキーワードまたは数字として指定できます。
		• コンソール、モニタ接続、またはシステム バッファにロギングする場合、*level* 引数によって指定された重大度のしきい値が ESM フィルタリングよりも優先されます。 ESM ターゲットに送信されるべきメッセージが ESM フィルタから返された場合でも、重大度が設定済みしきい値を満たさない（つまりレベル値より高い）場合には、メッセージが送信されません。
		• リモート ホストにロギングする場合、ストリーム タグを使用すると、メッセージ タイプに基づいて宛先を指定できます。 **stream***stream-id* 構文を使用すると、指定されたストリーム値を持つ

	コマンドまたはアクション	目的
		メッセージだけを特定のホストに送信するよう ESM を設定できます。 • ストリーム値は、設定された syslog フィルタ モジュールによって、メッセージに適用されます。 たとえば、重大度 5 のすべてのメッセージでストリーム タグ「20」を設定できます。 さらに、ストリーム タグ「20」を持つすべてのメッセージが 209.165.200.225 のホストに送信されるように指定できます。
ステップ 6	希望する各システム ロギング宛先に対して、ステップ 5 を繰り返します。	• ロギング ホスト コマンドを複数回発行することによって、異なるシステム ロギング ストリームに異なるターゲットを指定できます。 • コンソール、モニタ接続、またはシステム バッファに向けて、異なる重大度のメッセージを送信するように設定できます。 たとえば、ネットワーク オペレーション センター（NOC）の画面に重要なメッセージだけを表示できます（モニタまたはコンソール接続を使用）。
ステップ 7	**loggingsource-interface***typenumber* 例： `Device(config)# logging source-interface ethernet 0`	（任意）リモート syslog ホストに送信する syslog メッセージの送信元インターフェイスを指定します。 • 通常では、リモート ホストに送信される syslog メッセージは、メッセージ生成の時点で使用できるあらゆるインターフェイスを使用します。 このコマンドを使用すると、デバイスは、指定されたインターフェイスだけから syslog メッセージをリモート ホストに送信します。
ステップ 8	**loggingorigin-id** {**hostname** | **ip** | **ipv6** |**string***user-defined-id*} 例： `Device(config)# logging origin-id string “Domain 2, Router 5”`	（任意）リモート ホストに送信される syslog メッセージに発信元 ID を追加できます。 • 発信元 ID は、リモート ホストに送信されるすべての syslog メッセージの最初に追加されます。 ID はホスト名、IP アドレス、または指定する任意のテキストです。 • 発信元 ID は、syslog 出力を複数のデバイスから 1 つの syslog ホストに送信する場合、システム ロギング メッセージの送信元を識別するのに役立ちます。
ステップ 9	**end** 例： `Device(config)# end`	現在のコンフィギュレーション セッションを終了し、CLI を特権 EXEC モードに返します。

	コマンドまたはアクション	目的
ステップ 10	**showlogging** 例： `Device# show logging`	（任意）ESMのフィルタリングされたロギングのステータスを含む、システム ロギングのステータスを表示します。 • バッファへのフィルタリングされたロギングがイネーブルの場合、このコマンドは、バッファに保存されたデータも表示します。 • syslog フィルタ モジュールをこのコマンドの出力に示す順番は、フィルタ モジュールが実行される順番です。

Embedded Syslog Manager の設定例

例：Embedded Syslog Manager の設定例

次の例では、コンソール接続の Embedded Syslog Manager（ESM）フィルタ ロギングがイネーブル、モニタ接続とバッファでの標準ロギングがイネーブル、ホスト 209.165.200.225 での XML 形式のロギングがイネーブルです。

```
Device(config)# logging filter tftp://209.165.200.225/ESM/escalate.tcl
Device(config)# logging filter slot0:/email.tcl user@example.com
Device(config)# logging filter slot0:/email_guts.tcl
Device(config)# logging console filtered
Device(config)# logging monitor 4
Device(config)# logging buffered debugging
Device(config)# logging host 209.165.200.225 xml
Device(config)# end

Device# show logging
Syslog logging: enabled (0 messages dropped, 8 messages rate-limited,
               0 flushes, 0 overruns, xml disabled, filtering enabled)
    Console logging: level debugging, 21 messages logged, xml disabled,
                     filtering enabled
    Monitor logging: level warnings , 0 messages logged, xml disabled,
                     filtering disabled
    Buffer logging: level debugging, 30 messages logged, xml disabled,
                     filtering disabled
    Logging Exception size (8192 bytes)
    Count and timestamp logging messages: disabled

Filter modules:
    tftp://209.165.200.225/ESM/escalate.tcl
    slot0:/email.tcl user@example.com

    Trap logging: level informational, 0 message lines logged
        Logging to 209.165.200.225, 0 message lines logged, xml enabled,
              filtering disabled

Log Buffer (8192 bytes):

*Jan 24 09:34:28.431: %SYS-5-CONFIG_I: Configured from console by console
*Jan 24 09:34:51.555: %SYS-5-CONFIG_I: Configured from console by console
*Jan 24 09:49:44.295: %SYS-5-CONFIG_I: Configured from console by console
```

```
Device#
```

例：syslog フィルタ モジュール

syslog スクリプト モジュールは Tcl スクリプトです。 独自の syslog スクリプト モジュールの開発に役立つよう、次の例を示します。

（注） これらのスクリプト モジュールは単に例として提供されており、シスコによるサポートの対象外です。 これらのスクリプトの機能または影響に関して、明示的または黙示的に保証することはありません。

例：重大度のエスカレーション

この ESM syslog フィルタ モジュール例では、1 つのニーモニック（最初の CLI 引数経由で供給される）を監視し、メッセージの重大度を 2 つめの CLI 引数によって指定される重大度に拡大します。

```
#  ====================================================================
#  Embedded Syslog Manager                            ||        ||
#                                                     ||        ||
#  Severity Escalation Filter                        ||||      ||||
#                                              ..:||||||:..:||||||:..
#                                             ------------------------
#                                             C i s c o  S y s t e m s
#  ====================================================================
#
# Usage: Set CLI Args to "mnemonic new_severity"
#
# Namespace: global
# Check for null message
if { [string length $::orig_msg] == 0} {
   return ""
}

if { [info exists ::cli_args] } {
    set args [split $::cli_args]
    if { [ string compare -nocase [lindex $args 0] $::mnemonic ] == 0 } {
        set ::severity [lindex $args 1]
        set sev_index [ string first [lindex $args 0] $::orig_msg ]
        if {  $sev_index >= 2 } {
           incr sev_index -2
           return [string replace $::orig_msg $sev_index $sev_index \
              [lindex $args 1]]
        }
    }
}
return $::orig_msg
```

例：メッセージのカウント

この ESM syslog フィルタ モジュール例は、読みやすくするために 2 つのファイルに分割されています。 最初のファイルで、ユーザは msg_to_watch アレイを設定することにより、カウント対象のメッセージおよび要約する頻度（相関期間）を設定できます。 実際の手順は counting_guts.tcl ファ

イルに含まれています。 バックグラウンド処理を行う可能性がある他の ESM フィルタとの競合を避けるために、別個の名前空間「counting」を使用していることに注意してください。

```
#  ==================================================================
#  Embedded Syslog Manager                          ||        ||
#                                                   ||        ||
#  Message Counting Filter                         ||||      ||||
#                                             ..:||||||:..:||||||:..
#                                            ------------------------
#                                            C i s c o  S y s t e m s
# ==========================================================================


#
# Usage:
# 1) Define the location for the counting_guts.tcl script
#
# 2) Define message categories to count and how often to dump them (sec)
#    by populating the "msg_to_watch" array below.
#    Here we define category as facility-severity-mnemonic
#    Change dump time to 0 to disable counting for that category
#
# Namespace: counting
namespace eval ::counting {
    set sub_script_url tftp://172.16.0.0/12/ESM/counting_guts.tcl
    array set msg_to_watch {
        SYS-5-CONFIG_I            5
    }
#  ======================= End User Setup ==============================
# Initialize processes for counting
    if { [info exists init] == 0 } {
       source $sub_script_url
       set position $module_position
    }
# Process the message
process_category
} ;# end namespace counting
```

メッセージ カウント サポート モジュール（counting_guts.tcl）

```
#  ==================================================================
#  Embedded Syslog Manager                          ||        ||
#                                                   ||        ||
#  Message Counting Support Module                 ||||      ||||
#                                             ..:||||||:..:||||||:..
#  (No User Modification)                     ------------------------
#                                             C i s c o  S y s t e m s
# ==========================================================================



namespace eval ::counting {

# namespace variables

array set cat_msg_sev {}
array set cat_msg_traceback {}
array set cat_msg_pid {}
array set cat_msg_proc {}
array set cat_msg_ts {}
array set cat_msg_buginfseq {}
array set cat_msg_name {}
array set cat_msg_fac {}
array set cat_msg_format {}
array set cat_msg_args {}
array set cat_msg_count {}
array set cat_msg_dump_ts {}
```

```
# Should I count this message ?
   proc query_category {cat} {
        variable msg_to_watch
        if { [info exists msg_to_watch($cat)] } {
            return $msg_to_watch($cat)
        } else {
            return 0
        }
    }
   proc clear_category {index} {
        variable cat_msg_sev
        variable cat_msg_traceback
        variable cat_msg_pid
        variable cat_msg_proc
        variable cat_msg_ts
        variable cat_msg_buginfseq
        variable cat_msg_name
        variable cat_msg_fac
        variable cat_msg_format
        variable cat_msg_args
        variable cat_msg_count
        variable cat_msg_dump_ts
        unset cat_msg_sev($index) cat_msg_traceback($index) cat_msg_pid($index)\
              cat_msg_proc($index) cat_msg_ts($index) \
              cat_msg_buginfseq($index)  cat_msg_name($index) \
              cat_msg_fac($index) cat_msg_format($index) cat_msg_args($index)\
              cat_msg_count($index) cat_msg_dump_ts($index)
    }
# send out the counted messages
   proc dump_category {category} {
        variable cat_msg_sev
        variable cat_msg_traceback
        variable cat_msg_pid
        variable cat_msg_proc
        variable cat_msg_ts
        variable cat_msg_buginfseq
        variable cat_msg_name
        variable cat_msg_fac
        variable cat_msg_format
        variable cat_msg_args
        variable cat_msg_count
        variable cat_msg_dump_ts
        variable poll_interval
        set dump_timestamp [cisco_service_timestamp]
foreach index [array names cat_msg_count $category] {
      set fsm "$cat_msg_fac($index)-$cat_msg_sev($index)-$cat_msg_name($index)"
      set ::orig_msg \
        [format "%s%s: %%%s: %s %s %s %s - (%d occurence(s) between %s and %s)"\
        $cat_msg_buginfseq($index)\
          $dump_timestamp\
          $fsm \
          [uplevel 1 [linsert $cat_msg_args($index) 0 ::format
$cat_msg_format($index) ]] \
          $cat_msg_pid($index) \
          $cat_msg_proc($index) \
          $cat_msg_traceback($index) \
          $cat_msg_count($index) \
          $cat_msg_ts($index) \
          $dump_timestamp]
# Prepare for remaining ESM filters
                set ::severity $cat_msg_sev($index)
                set ::traceback $cat_msg_traceback($index)
                set ::pid $cat_msg_pid($index)
                set ::process $cat_msg_proc($index)
                set ::timestamp $cat_msg_ts($index)
                set ::buginfseq $cat_msg_buginfseq($index)
                set ::mnemonic $cat_msg_name($index)
                set ::facility $cat_msg_fac($index)
                set ::format_string $cat_msg_format($index)
                set ::msg_args [split $cat_msg_args($index)]
                esm_errmsg $counting::position
                clear_category $index
        }
```

```
  }
# See if this message already has come through since the last dump.
# If so, increment the count, otherwise store it.
   proc process_category {} {
        variable cat_msg_sev
        variable cat_msg_traceback
        variable cat_msg_pid
        variable cat_msg_proc
        variable cat_msg_ts
        variable cat_msg_buginfseq
        variable cat_msg_name
        variable cat_msg_fac
        variable cat_msg_format
        variable cat_msg_args
        variable cat_msg_count
        variable cat_msg_dump_ts
        if { [string length $::orig_msg] == 0} {
           return ""
        }
        set category "$::facility-$::severity-$::mnemonic"
        set correlation_window [expr [ query_category $category ] * 1000]
        if { $correlation_window == 0 } {
            return $::orig_msg
        }
        set message_args [join $::msg_args]
        set index "$category,[lindex $::msg_args 0]"
        if { [info exists cat_msg_count($index)] } {
                 incr cat_msg_count($index)
         } else {
              set cat_msg_sev($index) $::severity
              set cat_msg_traceback($index) $::traceback
              set cat_msg_pid($index) $::pid
              set cat_msg_proc($index) $::process
              set cat_msg_ts($index) $::timestamp
              set cat_msg_buginfseq($index) $::buginfseq
              set cat_msg_name($index) $::mnemonic
              set cat_msg_fac($index) $::facility
              set cat_msg_format($index) $::format_string
              set cat_msg_args($index) $message_args
              set cat_msg_count($index) 1
              set cat_msg_dump_ts($index) [clock seconds]
              catch [after $correlation_window counting::dump_category $index]
         }
         return ""
   }
# Initialized
set init 1
} ;#end namespace counting
```

例：XML タギング

この ESM syslog フィルタ モジュールは、ユーザ定義の XML タグを syslog メッセージに適用します。

```
#  ==================================================================
#  Embedded Syslog Manager                          ||        ||
#                                                   ||        ||
#  XML Tagging Filter                              ||||      ||||
#                                             ..:||||||:..:||||||:..
#                                            ------------------------
#                                            C i s c o  S y s t e m s
#  ==================================================================
#
# Usage: Define desired tags below.
#
# Namespace: xml
# Check for null message
    if { [string length $::orig_msg] == 0} {
       return ""
```

```
    }
namespace eval xml {
####  define tags ####
set MSG_OPEN "<ios-log-msg>"
set MSG_CLOSE "</ios-log-msg>"
set FAC_OPEN   "<facility>"
set FAC_CLOSE  "</facility>"
set SEV_OPEN   "<severity>"
set SEV_CLOSE  "</severity>"
set MNE_OPEN   "<msg-id>"
set MNE_CLOSE  "</msg-id>"
set SEQ_OPEN   "<seq>"
set SEQ_CLOSE  "</seq>"
set TIME_OPEN  "<time>"
set TIME_CLOSE "</time>"
set ARGS_OPEN  "<args>"
set ARGS_CLOSE "</args>"
set ARG_ID_OPEN "<arg id="
set ARG_ID_CLOSE "</arg>"
set PROC_OPEN "<proc>"
set PROC_CLOSE "</proc>"
set PID_OPEN "<pid>"
set PID_CLOSE "</pid>"
set TRACE_OPEN "<trace>"
set TRACE_CLOSE "</trace>"
#  ======================= End User Setup ==============================
#### clear result ####
set result ""
#### message opening, facility, severity, and name ####
append result $MSG_OPEN $FAC_OPEN $::facility $FAC_CLOSE $SEV_OPEN $::severity
$SEV_CLOSE $MNE_OPEN $::mnemonic $MNE_CLOSE
#### buginf sequence numbers ####
if { [string length $::buginfseq ] > 0 } {
    append result $SEQ_OPEN $::buginfseq $SEQ_CLOSE
}
#### timestamps ####
if { [string length $::timestamp ] > 0 } {
    append result $TIME_OPEN $::timestamp $TIME_CLOSE
}
#### message args ####
if { [info exists ::msg_args] } {
    if { [llength ::msg_args] > 0 } {
        set i 0
        append result $ARGS_OPEN
        foreach arg $::msg_args {
            append result $ARG_ID_OPEN $i ">" $arg $ARG_ID_CLOSE
                incr i
        }
        append result $ARGS_CLOSE
    }
}
#### traceback ####
if { [string length $::traceback ] > 0 } {
    append result $TRACE_OPEN $::traceback $TRACE_CLOSE
}
#### process ####
if { [string length $::process ] > 0 } {
    append result $PROC_OPEN $::process $PROC_CLOSE
}
#### pid ####
if { [string length $::pid ] > 0 } {
    append result $PID_OPEN $::pid $PID_CLOSE
}
#### message close ####
append result $MSG_CLOSE
return "$result"
} ;# end namespace xml
```

例：SMTP ベースの電子メール アラート

この ESM syslog フィルタ モジュール例では、コンフィギュレーション メッセージを監視し、CLI 引数として供給される E メール アドレスに送信します。 このフィルタは 2 つのファイルに分割されています。 最初のファイルはフィルタを実装し、2 番目のファイルは Simple Mail Transfer Protocol（SMTP）クライアントを実装します。

```
#  ==================================================================
#  Embedded Syslog Manager                          ||        ||
#                                                   ||        ||
#  Email Filter                                    ||||      ||||
#  (Configuration Change Warning)               ..:||||||:..:||||||:..
#                                             ------------------------
#                                              C i s c o  S y s t e m s
#   ==================================================================
# Usage:  Provide email address as CLI argument.  Set email server IP in
#         email_guts.tcl
#
# Namespace: email
if { [info exists email::init] == 0 } {
   source tftp://123.123.123.123/ESM/email_guts.tcl
}
# Check for null message
if { [string length $::orig_msg] == 0} {
      return ""
   }
if { [info exists ::msg_args] } {
    if { [string compare -nocase CONFIG_I $::mnemonic ] == 0 } {
                email::sendmessage $::cli_args $::mnemonic \
                [string trim $::orig_msg]
    }
}
return $::orig_msg
```

E メール サポート モジュール（email_guts.tcl）

```
#  ==================================================================
#  Embedded Syslog Manager                          ||        ||
#                                                   ||        ||
#  Email Support Module                            ||||      ||||
#                                               ..:||||||:..:||||||:..
#                                             ------------------------
#                                              C i s c o  S y s t e m s
#   ==================================================================
#
# Usage: Set email host IP, from, and friendly strings below.
#
namespace eval email {
    set sendmail(smtphost)172.16.0.1
    set sendmail(from) $::hostname
    set sendmail(friendly) $::hostname
    proc sendmessage {toList subject body} {
        variable sendmail
        set smtphost $sendmail(smtphost)
        set from $sendmail(from)
        set friendly $sendmail(friendly)
        set sockid [socket $smtphost 25]
## DEBUG
set status [catch {
        puts $sockid "HELO $smtphost"
        flush $sockid
        set result [gets $sockid]
        puts $sockid "MAIL From:<$from>"
        flush $sockid
        set result [gets $sockid]
        foreach to $toList {
```

```
            puts $sockid "RCPT To:<$to>"
            flush $sockid
        }
        set result [gets $sockid]
        puts  $sockid "DATA "
        flush $sockid
        set result [gets  $sockid]
        puts  $sockid "From: $friendly <$from>"
        foreach to $toList {
            puts $sockid "To:<$to>"
        }
        puts  $sockid "Subject: $subject"
        puts  $sockid "\n"
        foreach line [split $body "\n"] {
            puts  $sockid " $line"
        }
        puts  $sockid "."
        puts  $sockid "QUIT"
        flush $sockid
        set result [gets  $sockid]
} result]
        catch {close $sockid }
    if {$status} then {
        return -code error $result
    }
}
} ;# end namespace email
set email::init 1
```

例：ストリーム

この ESM syslog フィルタ モジュールの例では、特定のファシリティを監視し（最初の CLI 引数）、これらのメッセージを特定のストリームにルーティングします（2 番目の CLI 引数）。

```
#  ==================================================================
#  Embedded Syslog Manager                          ||        ||
#                                                   ||        ||
#  Stream Filter (Facility)                        ||||      ||||
#                                             ..:||||||:..:||||||:..
#                                            ------------------------
#                                             C i s c o  S y s t e m s
#   ==================================================================
# Usage:  Provide facility and stream as CLI arguments.
#
# Namespace: global
# Check for null message
#  ======================= End User Setup ==============================
set args [split $::cli_args]
if { [info exists ::msg_args] } {
    if { $::facility == [lindex $args 0] } {
                set ::stream [lindex $args 1]
    }
}
return $::orig_msg}
```

例：送信元 IP タギング

loggingsource-interface CLI コマンドを使用すると、デバイスから送信されるすべての syslog パケットで送信元 IP アドレスを指定できます。 次の syslog フィルタ モジュールの例では、フィルタ モジュール内で **show** CLI コマンド（この場合は **showrunning-config** および **showipinterface**）を使用して、syslog メッセージに送信元 IP アドレスを追加する方法を示します。 スクリプトは、最初に「source_ip::init」というローカル名前空間変数を検索します。 処理される最初の syslog メッセー

ジで変数が定義されていない場合、フィルタは**show**コマンドを実行し、正規表現を使用して、送信元インターフェイスとその IP アドレスを取得します。

このスクリプトでは、**show**コマンドが 1 回だけ実行されることに注意してください。送信元インターフェイスまたはその IP アドレスが変更される場合、新しい情報を取得するためにフィルタを再び初期化する必要があります（show コマンドをすべての syslog メッセージで実行できますが、これはうまく拡張できません）。

```
#  ==================================================================
#  Embedded Syslog Manager                              ||        ||
#                                                       ||        ||
#  Source IP Module                                    ||||      ||||
#                                                 ..:||||||:..:||||||:..
#                                               -------------------------
#                                                C i s c o  S y s t e m s
#   ==================================================================
# Usage: Adds Logging Source Interface IP address to all messages.
#
# Namespace:source_ip
#
#  ======================= End User Setup ==============================
namespace eval ::source_ip {
    if { [info exists init] == 0 } {
       if { [catch {regexp {^logging source-interface (.*$)} [exec show
run | inc logging source-interface] match source_int}]} {
           set suffix "No source interface specified"
       } elseif { [catch {regexp {Internet address is (.*)/.*$} [exec
show ip int $source_int | inc Internet] match ip_addr}]} {
           set suffix "No IP address configured for source interface"
       } else {
           set suffix $ip_addr
       }
       set init 1
    }

    if { [string length $::orig_msg] == 0} {
      return ""
    }
    return "$::orig_msg - $suffix"
} ;# end namespace source_ip
```

Embedded Syslog Manager に関する追加情報

関連資料

関連項目	マニュアル タイトル
Cisco IOS コマンド	『Cisco IOS Master Commands List, All Releases』
Cisco IOS XE コマンド	コマンド ルックアップ ツール
System Message Logging	「Troubleshooting and Fault Management」の章
XML 形式のシステム メッセージ ロギング	syslog メッセージ モジュールの XML インターフェイス
シスコ ソフトウェアでの Tcl 8.3.4 サポート	「Cisco IOS Scripting with Tcl」の章

関連項目	マニュアル タイトル
ネットワーク管理コマンド（logging コマンドを含む）：コマンド構文の詳細、デフォルト設定、コマンドモード、コマンド履歴、使用上のガイドライン、および例	『*Cisco IOS Network Management Command Reference*』

標準および RFC

標準/RFC	タイトル
新しい規格または変更された規格はサポートされていません。また、既存の規格に対するサポートに変更はありません。	--
RFC-3164	『*The BSD Syslog Protocol*』 この RFC は、情報提供のために記載しています。 シスコによる syslog の実装では、この RFC で言及されているプロトコルガイドラインとの完全な準拠性を要求していません。 サポートされている RFC がすべて記載されているわけではありません。

MIB

MIB	MIB のリンク
新しい規格または変更された規格はサポートされていません。また、既存の規格に対するサポートに変更はありません。	選択したプラットフォーム、Cisco IOS リリース、およびフィーチャ セットに関する MIB を探してダウンロードするには、次の URL にある Cisco MIB Locator を使用します。 http://www.cisco.com/go/mibs

シスコのテクニカル サポート

説明	リンク
★枠で囲まれた Technical Assistance の場合★右の URL にアクセスして、シスコのテクニカル サポートを最大限に活用してください。 これらのリソースは、ソフトウェアをインストールして設定したり、シスコの製品やテクノロジーに関する技術的問題を解決したりするために使用してください。 この Web サイト上のツールにアクセスする際は、Cisco.com のログイン ID およびパスワードが必要です。	http://www.cisco.com/cisco/web/support/index.html

Embedded Syslog Manager の機能情報

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

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

表 *1*：*Embedded Syslog Manager* の機能情報

機能名	リリース	機能情報
Embedded Syslog Manager	12.3(2)T 12.3(2)XE 12.2(25)S 12.2(33)SRC 12.2(33)SB 12.2(33)SXI Cisco IOS XE Release 2.1	Embedded Syslog Manager（ESM）機能は、Cisco IOS システム メッセージ ロガーによる伝送前に、システム ロギング メッセージのフィルタリング、拡大、相互の関連付け、ルーティング、およびカスタマイズを可能にするプログラマブル フレームワークを提供します。 導入または変更されたコマンドは、**loggingbufferedfiltered**、**loggingconsolefiltered**、**loggingfilter**、**logginghost**、**loggingmonitorfiltered**、**loggingorigin-id**、**showlogging** です。

用語集

（注） この用語集に記載されていない用語については、『Internetworking Terms and Acronyms』を参照してください。

コンソール：デバイスのコンソールポートへの接続（CTY またはコンソール回線）を示します。一般的にこれはコンソールポートに直接接続された端末または端末エミュレーションプログラムを備えた PC です。 **showterminal** コマンドに対応します。

モニタ：ライン ポートでの TTY（TeleTYpe 端末）ライン接続を示します。 つまり、「モニタ」キーワードは、端末回線接続または Telnet（端末エミュレーション）接続に対応します。 TTY ライン（ポートとも呼ばれます）は、端末、モデム、シリアル プリンタなどの周辺装置と通信します。 TTY 接続の例として、ダイヤルアップ モデムを使用してデバイスに接続する端末エミュレーション プログラムを備えた PC があります。

SEM：システム エラー メッセージの略。 システムロギング（syslog）プロセスによって生成されるメッセージを指す用語として、「システムエラーメッセージ」が以前に使用されていました。 syslog メッセージは標準化された形式を使用し、「緊急」（レベル 0）から「デバッグ」（レベル 7）までの 8 つの重大度があります。 これらのメッセージには、エラー以外のデバイス動作の通

知（情報の通知など）が含まれる場合もあるため、「システム エラー メッセージ」という用語は、実際に誤解を招くおそれがあります。

syslog：シスコ ソフトウェアでのシステム メッセージ ロギング プロセスの略。 また、「syslog メッセージ」のように、生成されたメッセージを指す場合もあります。 専門用語としての「syslog」はリモート ホストへのメッセージ ロギング プロセスのみを指しますが、一般的には、すべてのシスコ システム ロギング プロセスを示すために使用されます。

トラップ：エラー メッセージを送信するためのシステム ソフトウェア内のトリガー。 「トラップ ロギング」とは、リモート ホストへのメッセージのロギングを意味します。 リモート ホストはトラップ メッセージを送信するデバイスから見ると実際には syslog ホストですが、受信側デバイスは、収集された syslog データを他のデバイスに提供することが多いため、受信側デバイスもまた「syslog サーバ」と呼ばれます。