Firepower システム 修復 API ガイド

Cisco Systems, Inc.
www.cisco.com

シスコは世界各国 200 箇所にオフィスを開設しています。
各オフィスの住所、電話番号、FAX 番号は当社の
Web サイト
(www.cisco.com/go/offices) をご覧ください。

バージョン 6.0
2016 年 5 月 30 日

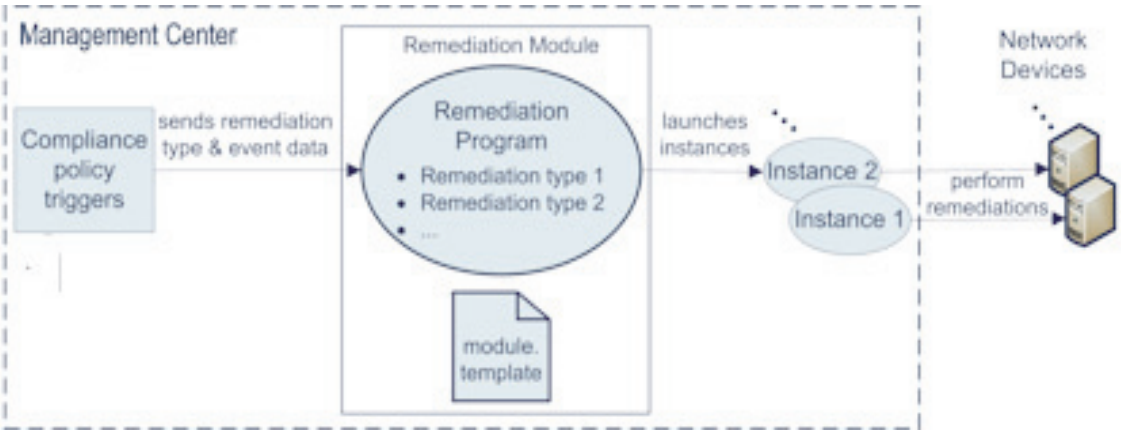
【注意】シスコ製品をご使用になる前に、安全上の注意（www.cisco.com/jp/go/safety_warning/）をご確認ください。

本書は、米国シスコシステムズ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。

修復サブシステムについて

Firepower System® 修復 API では、ネットワークの状況が関連する相関ポリシーに違反したときに、Firepower Management Center によって自動的に起動される修復を作成できます。*修復*は、検出された状況を軽減するためにソフトウェア プログラムが実行する応答です。たとえば、送信元または宛先 IP アドレスのルータでトラフィックをブロックしたり、ホストの状況を評価するホスト Nmap スキャンを開始したりできます。ポリシー内の複数のルールがトリガーとして使用された場合、Firepower Management Center はそれぞれのルールを起動できます。*修復モジュール*は、応答を実行するために Firepower Management Center にインストールするファイルのパッケージです。修復モジュールには、次の図に示すように、複数の*修復タイプ*を組み込むことができます。



ここに示す例では、システムに付属する修復モジュールの 1 つ（Cisco PIX ルータ モジュール）は、2 つの修復タイプを実行します。一方は送信元 IP アドレスでパケットをブロックし、もう一方は宛先 IP アドレスでブロックします。

修復モジュールがネットワーク上の複数のデバイス（ルータやホストなど）を対象とする場合、相関ポリシーがトリガーされたときに、複数の*インスタンス*（デバイスごとに 1 つずつ）を実行するように修復モジュールを設定します。インスタンスは、修復モジュール コードの機能に対応する 1 つ以上の修復タイプとターゲット デバイスで実行するのに必要な一連の変数で修復モジュールをインスタンス化したものです。インスタンスごとに、実行する修復タイプ、およびネットワークのターゲット デバイスにアクセスするためのインスタンス固有の情報（修復に関するデバイスの IP アドレスとパスワードなど）を指定します。

Firepower System のドメイン機能では、管理対象デバイス、設定、イベントへのユーザ アクセスをセグメント化することによって、展開内にマルチテナンシーを実装できます。複数のドメインを使用するシステムの場合、ユーザは異なるドメイン レベルに修復を作成できます。修復は、修復が作成されたドメインの親ドメインや同等のドメインには表示されません。子ドメインのユーザは、その親ドメインに作成された修復を表示できますが、変更したり、削除したりはできません。ルート ドメインで作成される修復は、すべての子ドメインで表示および使用できます。

前提条件

カスタム修復の修復 API を使用する前に、次のカテゴリの情報について十分に理解している必要があります。

- Firepower System（1-2 ページ）
- プログラミング要件とサポート（1-2 ページ）
- シスコが提供する修復モジュール（1-3 ページ）

Firepower System

このガイドの情報を理解するには、Firepower System の機能と名称、および特定のコンポーネントの機能について理解している必要があります。

- Firepower Management Center アーキテクチャの Firepower System ロール
- Firepower Management Center の相関ポリシー管理モジュール
- Firepower Management Center の修復管理モジュール

詳細については、『*Firepower System User Guide*』を参照してください。

プログラミング要件とサポート

カスタム修復を Perl またはシェル スクリプトで、あるいはプリコンパイルされた静的リンク済み C プログラム（glibc のルーチンへのリンクを除く）としてコーディングできなければなりません。

また、各修復モジュールのコンフィギュレーション ファイルを XML で作成できなければなりません。このファイルは、`module.template` と呼ばれます。システムに付属する修復モジュール（このファイルのサンプル）を参照してください。Firepower Management Center におけるモジュールの場所については、修復サブシステム ファイルの構造について（4-4 ページ）を参照してください。

追加するインスタンスごとに、Firepower Management Center は `instance.conf` と呼ばれるインスタンス固有の XML コンフィギュレーション ファイルを生成します。修復インスタンスが実行されるたびに、このファイルを解析するようにコーディングする必要があります。

次の表に、修復プログラムを作成および実行するために、リソースとして Firepower Management Center で使用可能なパッケージを示します。

表 1-1　　追加パッケージ

追加パッケージ	場所
GNU bash バージョン 3.2.33(1) リリース	`/bin/bash`
tcsh 6.17.00	`/bin/tcsh`
glibc 2.7	`/lib/libc-2.7.so`
perl v5.10.1	`/usr/bin/perl`
Net::Telnet	`N/A`
Net::SSH::Perl	`N/A`
XML::Smart	`N/A`

シスコが提供する修復モジュール

次の表で、Firepower Management Center に含まれている事前定義された修復モジュールについて説明します。修復プログラムを設計するときに、参照用にこれらのモジュールを使用してください。

システム付属のモジュールは、すでに Firepower Management Center にインストールされており、各モジュールの実行可能ファイル（Perl、C）とコンパイル済み `module.template` コンフィギュレーション ファイルの両方が含まれます。システム付属の修復モジュールを展開する簡単な手順については、『*Firepower System User Guide*』を参照してください。

表 1-2　　シスコが提供する修復モジュール

モジュール名	機能
Cisco IOS Null ルート	Cisco IOS® バージョン 12.0 以降を使用する Cisco ルータが実行中の場合、相関ポリシーに違反する IP アドレスまたはネットワークに送信されるトラフィックを動的にブロックできます。
Cisco PIX Shun	Cisco PIX® ファイアウォール バージョン 6.0 以降を実行中の場合、相関ポリシーに違反する IP アドレスから送信されたトラフィックを動的にブロックできます。
Nmap スキャン	特定のターゲットを能動的にスキャンし、そうしたホスト上で稼動中のオペレーティング システムおよびサーバを判別できます。
属性値の設定	相関イベントが発生するホストのホスト属性を設定できます。

修復サブシステム

修復サブシステムは次のコンポーネントから構成されています。

- Firepower Management Center の Web インターフェイス。このインターフェイスを使用して、相関ポリシーを設定したり、それらを修復に関連付けたりします。また、修復処理のステータスを追跡するためにも使用します。
- 修復 API。これにより、修復モジュールに提供されるデータを定義できます。
- 修復デーモン。これは、実行時に修復モジュールにデータを渡したり、実行ステータス情報を収集したりします。
- 修復モジュール。相関ポリシー違反に対して特定の応答を実行します。

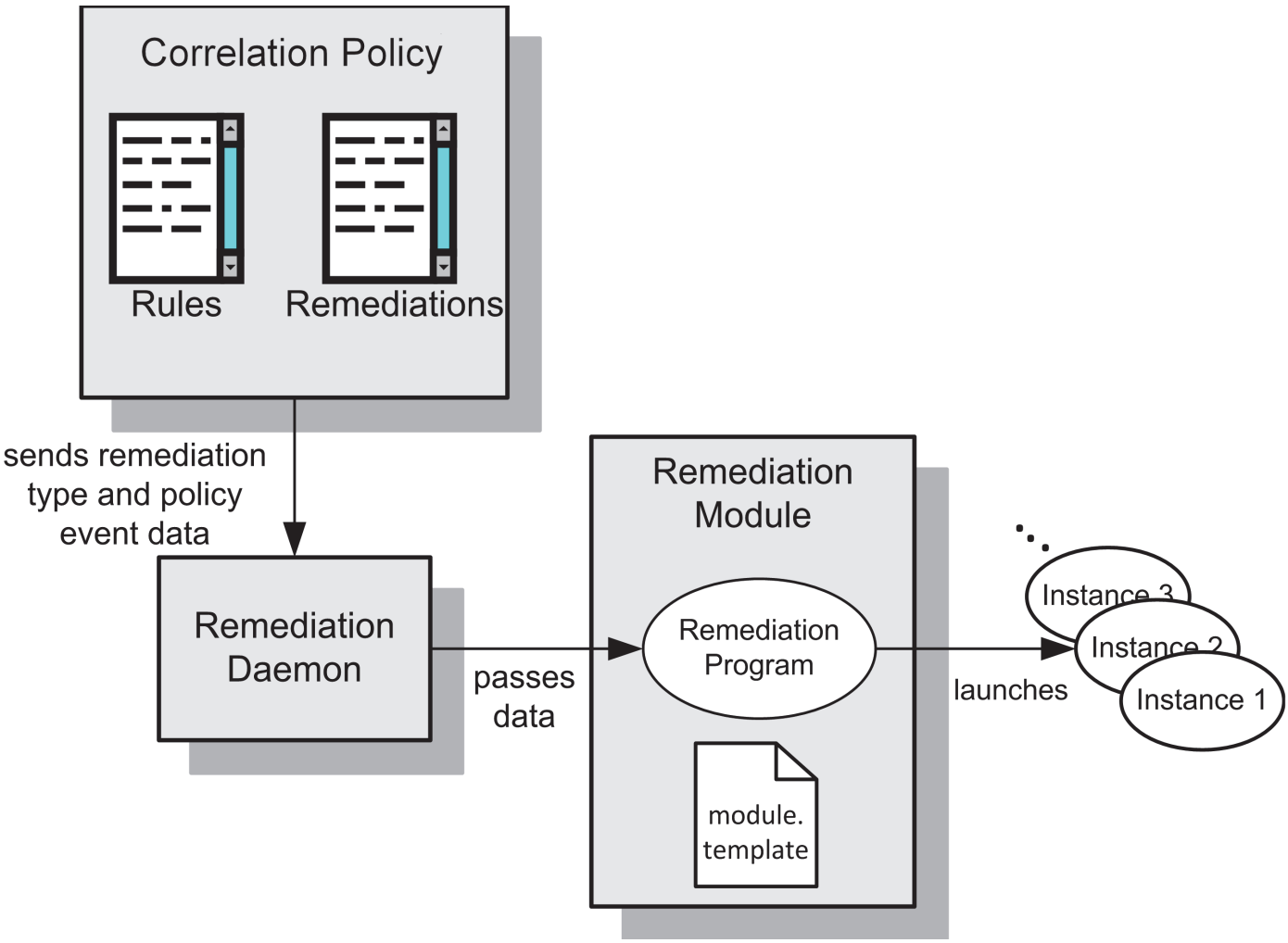
修復サブシステム アーキテクチャについて

修復サブシステムには、以下の図に示されている 2 つの部分からなるアーキテクチャがあります。アーキテクチャは、次のものから構成されます。

- すべての修復モジュールをサポートする Web インターフェイスや修復デーモンなどのインフラストラクチャ コンポーネント。インフラストラクチャ コンポーネントによって、Firepower Management Center 上のすべての修復モジュールを作成および管理できます。修復デーモンは修復の実行を管理します。詳細については、修復サブシステム コンポーネント（1-4 ページ）を参照してください。
- 特定の相関ポリシー違反に応答するようにユーザが作成する個々の修復モジュール。詳細については、修復モジュール アーキテクチャ（1-5 ページ）を参照してください。

修復サブシステム コンポーネント

次の図は、修復サブシステムのメイン機能とその対話を示しています。



自動モードでネットワークのルール違反に応答するように修復を作成します。Firepower Management Center Web インターフェイスでは、相関ポリシーを定義して有効化したり、それらを修復に関連付けたりすることができます。ポリシー違反が発生すると、修復サブシステムは修復の名前と `module.template` コンフィギュレーション ファイルで指定されたイベント データを修復デーモンに渡します。

修復デーモンは修復を起動し、相関イベント データとインスタンス固有のパラメータを修復プログラムに渡します。また、修復プログラムからの戻りコードを受け入れます。Firepower Management Center はステータスの表示に戻りコードを使用します。

関連するポリシー ルールがトリガーされると、修復プログラムは修復の一連の *インスタンス* を起動します。各インスタンスは、特定のネットワーク デバイスを対象とします。Firepower Management Center Web インターフェイスの [Instance Detail] ページでインスタンスを作成します。インスタンスごとに、ターゲット デバイスの IP アドレスとパスワードなど、必要なインスタンス固有の設定の詳細情報を指定します。

修復モジュール アーキテクチャ

Firepower Management Center にインストールする各修復モジュールには、1 つ以上の修復タイプが含まれます。各インスタンスに 1 つ以上の修復タイプを割り当てます。ポリシー違反に対する応答として修復を設定する方法については、『*Firepower System User Guide*』の「Configuring Remediations」の章を参照してください。

修復モジュールには、次のコンポーネントが含まれます。

- 修復プログラム。インストール時の修復モジュール パッケージに含まれます。修復モジュールの計画とパッケージ化（2-1 ページ）を参照してください。
- 必要な XML `module.template` ファイル。これもインストール時の修復モジュール パッケージに含まれます。このファイルでは、修復モジュールのインスタンスのいずれかを起動するたびに、修復サブシステムが参照するモジュールとそのデータ要件に関するモジュール レベルの情報が提供されます。修復サブシステムとの通信（3-1 ページ）を参照してください。
- インスタンスごとの XML `instance.conf` ファイル。修復モジュールの新しいインスタンスを設定するたびに、Firepower Management Center はこのファイルを自動生成します。

修復サブシステムの使用

Firepower Management Center 上の相関ポリシーの特定のルールに対する応答として追加することにより、修復を展開します。Firepower Management Center Web インターフェイスを使用して、相関ポリシーと修復の関連付けを定義します。

修復モジュール展開するには、次の手順を実行する必要があります。

1. 軽減したい状況、およびご使用の環境でその状況を適切に解決するアクションを識別します。これらのアクションは、カスタム修復プログラムが実装しなければならないメイン機能になります。

   シスコが提供する修復モジュールを使用できる場合、手順6.モジュールのインストール（2-12 ページ）で説明されているように、Web インターフェイスを使用して、Firepower Management Center にモジュールをインストールします。（1-5 ページ）までスキップします。

2. カスタム修復モジュールを作成する必要がある場合は、修復サブシステムから取得可能なデータ要素について理解してください。修復サブシステムで使用可能なデータ（2-1 ページ）を参照してください。
3. カスタム修復モジュールを作成した場合、モジュール パッケージに含めるモジュール テンプレート ファイルも作成する必要があります。ファイルの形式と構文については、修復サブシステムとの通信（3-1 ページ）を参照してください。
4. 目的の修復に必要なすべての機能に対応するように修復プログラムを記述します。bash、tsch、Perl、または C で修復モジュール プログラムを記述できます。修復プログラムの開発者への注意事項（4-3 ページ）の技術的なガイダンスを参考にして、プログラムを作成してください。
5. モジュールのパッケージ化（2-12 ページ）で説明されているように、修復モジュールをパッケージ化します。
6. モジュールのインストール（2-12 ページ）で説明されているように、Web インターフェイスを使用して、Firepower Management Center にモジュールをインストールします。
7. 修復モジュールの個々の修復タイプが、アクティブな相関ポリシーの適切な相関ルールに対する応答として割り当てられていることを確認します。手順の詳細については、『*Firepower System User Guide*』を参照してください。

修復リソース

このドキュメントに加えて、修復モジュールを作成するために使用できるその他のリソースには、以下が含まれます。

- C または Perl で記述されたサンプル プログラム コードを含む修復 SDK。この SDK は、syslog アラートを生成し、モジュールがネットワークと対話する方法について示します。詳細については、このドキュメントの修復 SDK の使用（4-1 ページ）の章を参照してください。SDK はサポート サイトからダウンロードできます。
- `module.template` スキーマ（`module.template.xsd`）。これは、Firepower Management Center の `/etc/sf/remediation/module.template.xsd` にあります。

次の表に、このドキュメントで説明されているトピックの一部と、詳細の参照先を示しています。

表 1-3　　修復リソース

内容	参照先
サンプル修復モジュールと、修復モジュールの作成、インストール、設定の一般的な手順	修復 SDK の使用（4-1 ページ）
修復プログラムの記述方法	修復モジュールの計画とパッケージ化（2-1 ページ）
`module.template` ファイルの作成方法	修復サブシステムとの通信（3-1 ページ）
Firepower Management Center にインストールできるようにする修復モジュールのパッケージ化	モジュールのパッケージ化（2-12 ページ）
修復モジュールのインストール方法	モジュールのインストール（2-12 ページ）
セキュリティ ポリシー違反に対する応答としての修復の設定	『*Firepower System User Guide*』の「Configuring Remediations」の章

修復モジュールの計画とパッケージ化

カスタム修復モジュールを開発する計画には、次の表に示すタスクが含まれます。この表では、各タスク領域の情報とガイダンスが得られる場所が示されています。

表 2-1　修復モジュールの計画タスク

ガイダンス内容	探す場所
機能分析の実行、および修復サブシステムの動作概念について理解することの重要性	展開とインストールの手順の概要（4-2 ページ）
修復サブシステムで使用可能なデータの確認	修復サブシステムで使用可能なデータ（2-1 ページ）
修復サブシステムの戻りコード機能の使用	モジュールによって返されるデータ（2-11 ページ）
ソフトウェア開発の調整、および `module.template` ファイルの生成	修復サブシステムとの通信（3-1 ページ）
修復モジュールのパッケージ化とそのインストール	モジュールのパッケージ化とインストール（2-11 ページ）

修復サブシステムで使用可能なデータ

カスタム修復モジュールは修復サブシステムから 2 種類のデータを受信できます。

- イベント データ。これには、違反した相関ポリシーに関するデータやポリシー違反の原因となった元のトリガー イベントに関するデータなどさまざまなデータが含まれます。
- インスタンス設定データ。これには、修復のインスタンスが設定されたときに Web インターフェイスで入力された値が含まれます。

これら 2 つのタイプのデータには、違反ポリシーのルールをトリガーしたネットワーク トラフィックまたは変更に関するデータとそのポリシー違反に応じて実行される修復の設定済みインスタンスの両方が含まれます。相関ポリシーと修復の作成、設定、および使用に関する詳細については、『*Firepower System User Guide*』の「Correlation Policies」および「Configuring Remediations」を参照してください。

詳細については、次の項を参照してください。

- イベント データ（2-2 ページ）では、イベント データが修復モジュールに提供される方法と、モジュールで使用可能な相関イベント データが示されます。
- インスタンス設定データ（2-8 ページ）では、`instance.config` ファイルを修復モジュールで使用できるようにする方法と、それらのファイルに含めることができるデータ タイプが示されます。

イベント データ

イベント データは、修復モジュールに使用できる情報の 1 つのタイプです。イベントは、相関ポリシーのルールがトリガーされるときに、Firepower Management Center が生成する侵入、相関、およびその他のイベント タイプに関する情報です。`module.template` ファイルの `pe_item` 要素を使用して、モジュールの各修復タイプに送信されるイベント データ フィールドを指定します。

修復デーモンが修復モジュールにイベント データを送信する際、最初に修復の名前を渡し、その後 `module.template` に表示される順序で `pe_item` フィールドの相関イベント データを渡します。

修復デーモンは、フィールドが module.template で optional または required とマークされているかに応じて、データベースの未定義 `pe_item` フィールドを異なる方法で処理します。未定義データ要素の処理（4-5 ページ）を参照してください。

修復のイベント データの指定に関する詳細については、修復タイプの定義（3-19 ページ）を参照してください。`pe_item` 要素を指定する際、次の表に示すフィールド名を使用します。

次の表に、相関ポリシー違反をトリガーした元のイベントに関する使用可能なデータを示します。この表の一部のフィールドはイベントに固有であることに注意してください。それらのフィールドは、トリガー イベントの特定のタイプに使用できない場合、0 に設定されます。

表 2-2　　イベント データのトリガー

名前	説明	フィールド	タイプ	バイト
トランスポート プロトコル	侵入をトリガーしたパケット、またはポリシー違反の原因となった検出イベントのトランスポート プロトコル（TCP、UDP、IP、ICMP）。	ip_protocol	uint8_t	1
ネットワーク プロトコル	侵入をトリガーしたパケット、またはポリシー違反の原因となった検出イベントのネットワーク プロトコル（たとえば、イーサネット）。	net_protocol	uint16_t	2
トリガー イベント タイプ	相関イベントをトリガーしたイベントのタイプの数値 ID。値は次のとおりです。 1 = 侵入 2 = ネットワーク検出、接続、または接続の概要 3 = ユーザ対応 4 = ホワイト リスト	event_type	uint8_t	1
トリガー イベント ID	相関イベントをトリガーしたイベントの内部 ID。侵入イベントの場合のみ設定します。その他のイベント タイプの場合、0 に設定します。	event_id	uint32_t	4
トリガー イベント時間	内容はイベント タイプによって異なります。 侵入、ネットワーク検出、接続、およびユーザ対応のイベントの場合：トリガー イベントの UNIX タイムスタンプ 接続の概要の場合：相関イベント時間（つまり、policy_tv_sec） ホワイト リスト イベントの場合：0 に設定	tv_sec	uint32_t	4
トリガー イベント時間（usec）	イベント時間のマイクロ秒単位の増分。精度を使用できない場合は 0 に設定します。	tv_usec	uint32_t	4
トリガー イベントの説明	相関イベントをトリガーした元のイベントのテキスト説明。内容はイベント タイプによって異なります。	description	char *	最大 1024

表 2-2　イベント データのトリガー(続き)

名前	説明	フィールド	タイプ	バイト
トリガー イベント センサー ID	トリガー イベントが発生したセンサーの内部 ID。 主にシスコ内部で使用するためのもので、通常修復には使用されません。	sensor_id	uint32_t	4
トリガー イベント ジェネレータ ID	内容はイベント タイプによって異なります。 侵入イベントの場合:イベントのジェネレータ ID(GID)。GID の完全なリストについては、『*Firepower System User Guide*』を参照してください。 ネットワーク検出と接続イベントの場合:ネットワーク検出イベント タイプ。 接続の概要の場合:すべて 4 に設定します。 ユーザ対応イベントの場合:ユーザ対応イベント タイプ。 ホワイト リスト イベントの場合:0 に設定します。 主にシスコ内部で使用するためのもので、通常修復には使用されません。	sig_gen	uint32_t	4
トリガー イベント シグニチャ ID	内容はイベント タイプによって異なります。 侵入イベントの場合:イベントのシグニチャ ID(SID)。ユーザ インターフェイスに表示される SID と一致しない場合があります。 ネットワーク検出と接続イベントの場合:ネットワーク検出イベント サブタイプ。 接続の概要の場合:すべて 17 に設定します。 ユーザ対応イベントの場合:ユーザ対応イベント サブタイプ。 ホワイト リスト イベントの場合:0 に設定します。 主にシスコ内部で使用するためのもので、通常修復には使用されません。	sig_id	uint32_t	4

表 2-2　イベント データのトリガー（続き）

名前	説明	フィールド	タイプ	バイト
影響フラグ	イベントの影響フラグ値。下位 8 ビットは影響レベルを示します。値は次のとおりです。 0x01（ビット 0）- 送信元または宛先ホストはシステムによってモニタされるネットワーク内にあります。 0x02（ビット 1）- 送信元または宛先ホストはネットワークマップ内に存在します。 0x04（ビット 2）- 送信元または宛先ホストはイベントのポート上のサーバを実行しているか（TCP または UDP の場合）、IP プロトコルを使用します。 0x08（ビット 3）- イベントの送信元または宛先ホストのオペレーティング システムにマップされた脆弱性があります。 0x10（ビット 4）- イベントで検出されたサーバにマップされた脆弱性があります。 0x20（ビット 5）- イベントが原因で、管理対象デバイスがセッションをドロップしました（デバイスがインライン、スイッチド、またはルーテッド展開で実行している場合にのみ使用されます）。Firepower System Web インターフェイスのブロックされた状態に対応します。 0x40（ビット 6）- このイベントを生成するルールに、影響フラグを赤色に設定するルールのメタデータが含まれます。送信元ホストまたは宛先ホストは、ウイルス、トロイの木馬、または他の悪意のあるソフトウェアによって侵害される可能性があります。 0x80（ビット 7）- イベントで検出されたクライアントにマップされた脆弱性があります。（バージョン 5.0 以上のみ） 次の影響レベル値は、Firepower Management Center の特定の優先順位にマップされます。x は、値が 0 または 1 になることを示しています。 グレー（0、不明）:00X00000 赤（1、脆弱）:XXXX1XXX, XXX1XXXX, X1XXXXXX, 1XXXXXXX（バージョン 5.0 以上のみ） オレンジ（2、潜在的に脆弱）:00X0011X 黄（3、現在は脆弱でない）:00X0001X 青（4、不明なターゲット）:00X00001	impact_flags	uint32_t	4

次の表に、各相関イベントに関する使用可能なデータを示します。一部のデータ要素は、特定のイベント タイプに対して入力されないことに注意してください。

表 2-3　　相関イベント データ

名前	説明	フィールド	タイプ	バイト
相関イベント時間	相関イベントが生成されたときの UNIX タイムスタンプ。	policy_tv_sec	uint32_t	4
相関イベント ID	センサーによって生成されるイベントの内部 ID 番号。侵入イベントの場合のみ設定します。 主にシスコ内部で使用するためのもので、通常修復には使用されません。	policy_event_id	uint32_t	4
相関アプライアンス ID	相関イベントを生成した Firepower Management Center の内部 ID 番号。 主にシスコ内部で使用するためのもので、通常修復には使用されません。	policy_sensor_id	uint32_t	4
相関ポリシー ID	トリガー イベントが違反した相関ポリシーの内部 ID 番号。 主にシスコ内部で使用するためのもので、通常修復には使用されません。	policy_id	uint32_t	4
相関ルール ID	相関イベントをトリガーした相関ルールの内部 ID 番号。 主にシスコ内部で使用するためのもので、通常修復には使用されません。	rule_id	uint32_t	4
相関ルールの優先順位	イベントを生成した相関ポリシーのルールに割り当てられた優先順位。ルールは、別のポリシーでは異なる優先順位になる可能性があります。 値：0 - 5（0 = 優先順位なし）	priority	uint32_t	4
イベント 定義済みマスク	マスクの後に続くどのフィールドが有効かを示す相関イベント メッセージのビット フィールド。値については、表 2-4イベントで定義された値（2-5 ページ）を参照してください。 主にシスコ内部で使用するためのもので、通常修復には使用されません。	defined_mask	uint32_t	4

次の表は、相関イベント メッセージ フィールドのマスク値を定義しています。これらの値は、マスクの後に続くどのフィールドが有効かを示す相関イベント メッセージで使用されます。

表 2-4　　イベントで定義された値

相関イベント フィールド	マスク値
イベント影響フラグ	0x00000001
IP プロトコル	0x00000002
ネットワーク プロトコル	0x00000004
送信元 IP	0x00000008
送信元ホスト タイプ	0x00000010
送信元 VLAN ID	0x00000020

表 2-4　イベントで定義された値（続き）

相関イベント フィールド	マスク値
送信元フィンガープリント ID	0x00000040
送信元重要度	0x00000080
送信元ポート	0x00000100
送信元サーバ	0x00000200
宛先 IP	0x00000400
宛先ホスト タイプ	0x00000800
宛先 VLAN ID	0x00001000
宛先フィンガープリント ID	0x00002000
宛先重要度	0x00004000
宛先ポート	0x00008000
宛先サーバ	0x00010000
送信元ユーザ	0x00020000
宛先ユーザ	0x00040000

次の表に、侵入イベントに関係する送信元ホストに関する使用可能なデータ、または相関ポリシー違反の原因となったその他の検出イベントに関係するホストのみに関する使用可能なデータを示します。確実に入力されるのは、送信元 IP アドレスのみであることに注意してください。

表 2-5　送信元ホスト データ

名前	説明	フィールド	タイプ	バイト
IP Address	ポリシー違反をトリガーしたイベントの送信元ホストの IP アドレス。検出イベントの場合、ホストまたは開始ホストの IP アドレス。	src_ip_addr	uint32_t	4
Host Type ID	ホストの認識されたタイプ（たとえば、ルータ、ブリッジ）。検出イベントのみ。	src_host_type	uint8_t	1
VLAN ID	ホストの VLAN ID。検出イベントのみ。	scr_vlan_id	uint16_t	2
OS Vendor	ホストの識別されたオペレーティング システムのベンダー。検出イベントのみ。	src_os_vendor	char*	最大 255
OS Product	ホストの識別されたオペレーティング システム。検出イベントのみ。	src_os_product	char*	最大 255
OS Version	ホストの識別されたオペレーティング システムのバージョン番号。検出イベントのみ。	src_os_version	char*	最大 255
Host Criticality	ホストおよび接続イベントのユーザ定義値。	src_criticality	uint16_t	2

次の表に、送信元ホストのサーバに関する使用可能なデータ、または相関イベントの原因となったイベントで識別されたサーバのみに関する使用可能なデータを示します。確実に入力されるのは、トランスポート プロトコルのみであることに注意してください。

表 2-6　送信元サーバ データ

名前	説明	フィールド	タイプ	バイト
Port	識別されたサーバが実行されるポート。侵入イベントの場合、プロトコルが TCP または UDP の場合にのみポートが入力されます。	src_port	uint16_t	2
Server	ポリシー違反の原因となったイベントで識別されたサーバ（たとえば、HTTP、SMTP）。	src_service	char	最大 255

次の表に、宛先ホストに関する使用可能なデータを示します。このデータを使用できるのは侵入イベントのみです。

表 2-7　宛先ホスト データ

名前	説明	フィールド	タイプ	バイト
IP Address	ポリシー違反をトリガーしたイベントの宛先ホストの IP アドレス。	dest_ip_addr	uint32_t	4
Host Type ID	宛先ホストの認識されたタイプ（たとえば、ルータ、ブリッジ）。	dest_host_type	uint8_t	1
VLAN ID	宛先ホストの VLAN ID。	dest_vlan_id	uint16_t	2
OS Vendor	ホストの識別されたオペレーティング システムのベンダー。検出イベントのみ。	dest_os_vendor	char*	最大 255
OS Product	ホストの識別されたオペレーティング システム。検出イベントのみ。	dest_os_product	char*	最大 255
OS Version	ホストの識別されたオペレーティング システムのバージョン番号。検出イベントのみ。	dest_os_version	char*	最大 255
Host Criticality	検出ホストおよび接続イベントのユーザ定義値。	dest_criticality	uint16_t	2

次の表に、宛先ホストのサーバに関する使用可能なデータ、または相関イベントの原因となったイベントで識別されたサーバのみに関する使用可能なデータを示します。確実に入力されるのは、トランスポート プロトコルのみであることに注意してください。

表 2-8　宛先サーバ データ

名前	説明	フィールド	タイプ	バイト
Destination Port	識別されたサーバが実行されるポート。侵入イベントの場合、プロトコルが TCP または UDP として識別された場合にのみポートが入力されます。	dest_port	uint16_t	2
Destination Server	ポリシー違反の原因となったイベントで識別されたサーバ（たとえば、HTTP、SMTP）。	dest_service	char	最大 255

インスタンス設定データ

ユーザがモジュールの新しいインスタンスを設定する際、module.template ドキュメントで要求されるデータを入力します。ユーザが入力した値は、修復プログラムによって、使用する instance.conf ドキュメントに書き込まれます。

修復の各設定済みインスタンスでは、修復サブシステムは instance.conf ドキュメントをインスタンスと同じ名前のディレクトリに配置します。このディレクトリは、モジュールがアップロードされ、インストールされたディレクトリに作成されます。たとえば、モジュールが「Firewall」と呼ばれる場合、firewall というディレクトリにアップロードされます。次に block_tokyo と呼ばれるインスタンスを設定する場合、修復サブシステムは firewall ディレクトリの block_tokyo と呼ばれるディレクトリを作成し、そこに instance.conf を配置します。次のようなディレクトリ パスが表示されます。

```
/var/sf/remediation/firewall/block_tokyo/instance.config
```

モジュール ファイルが存在するディレクトリの詳細については、モジュールのパッケージ化（2-12 ページ）を参照してください。

モジュールで、instance.conf ファイルのオープン、読み取り、解析、クローズを実行できる必要があります。

各 instance.conf ドキュメントには、instance という名前の最上位の要素が含まれます。instance 要素には、config と remediation の 2 つの子要素があります。次の表に、instance 要素で使用可能な属性と要素を示します。

表 2-9　instance 属性と子要素

名前	タイプ	説明
name	属性	ドキュメントのデータを、名前が付けられ、設定されたインスタンスと関連付け、設定するユーザが指定したインスタンスの名前が反映されるようにします。
config	要素	設定時に Web インターフェイスのインスタンス設定フィールドに入力されたデータが含まれます。
remediation	要素	インスタンスの修復を設定する際に、Web インターフェイスに入力されたデータが含まれます。

config および remediation 要素で指定されるデータに関する詳細については、次を参照してください。

- config 要素（2-8 ページ）
- remediation 要素（2-10 ページ）

config 要素

config 要素には、修復モジュールの module.template ドキュメントの config_template 要素に応じて Web インターフェイスに表示されるフィールドに入力したデータが含まれます。これらのフィールドは、子要素ではなく、その要素の属性として指定された名前を使用して指定された、module.template ドキュメントに指定するために使用される要素に再度変換されます。それらには、次の種類のフィールドが含まれます。

- boolean
- string
- integer
- password
- host

- netmask
- network
- ipaddress
- enumeration
- list

これらのフィールドの module.template ファイルでの指定方法の詳細については、設定テンプレートの定義（3-4 ページ）を参照してください。

たとえば、module.template ドキュメントに次の config_template 要素定義が含まれる場合：

```
<config_template>
<ipaddress>
    <name>host_ip</name>
    <display_name>Host IP</display_name>
</ipaddress>
<string>
    <name>user_name</name>
    <display_name>Username</display_name>
    <constraints>
           <pcre>\S+</pcre>
    </constraints>
</string>
<password>
    <name>login_password</name>
    <display_name>Login Password</display_name>
</password>
</config_template>
```

その要素の [Instance Configuration] 画面には、次の 3 つのフィールドが含まれます。

- [Host IP] IP アドレス値を取ります。
- [Username] スペース文字を含まない文字列値を取ります。
- [Login Password] パスワードとして識別される文字列値を取ります。

ユーザが修復モジュールの AdminInstance という名前のインスタンスを設定し、次の値を指定したとします。

表 2-10　サンプル値

フィールド	値
Host IP	192.1.1.1
Username	adminuser
Login Password	3admin3

instance.conf には、次が含まれます。

```
<instance name="AdminInstance">
<config>
    <ipaddress name="host_ip">192.1.1.1</ipaddress>
    <string name="user_name">adminuser</string>
    <password name="login_password">3admin3</password>
</config>
```

上記の例では、</instance> が含まれないことに注意してください。これは、この例のインスタンスの instance.conf ドキュメントに、このセクションの次で説明する remediation 要素を含めるためです。モジュールで追加の修復設定が必要ない場合、そのモジュールに返される instance.conf は修復要素を**含みません**。

remediation 要素

instance 要素には、そのインスタンスに対して設定された各修復の remediation 要素が含まれています。各 remediation 要素には、属性として修復インスタンスの名前（インスタンスの設定時に Web インターフェイスで入力した）と module.template ドキュメントの remediation_type 要素で最初に指定した修復タイプがあります。module.template ファイルの詳細については、修復サブシステムとの通信（3-1 ページ）を参照してください。

また、remediation 要素には、config 要素を含めることができます。これらは、instance の子要素である config 要素と同じ方法で機能しますが、module.template ドキュメントの remediation_type の子である config_template 要素で最初に指定されているデータを使用します。次に、これらの属性と要素を示します。

表 2-11　remediation 属性と子要素

名前	タイプ	説明
name	属性	ドキュメントのデータを、名前が付けられ、設定された修復と関連付け、設定するユーザが指定した名前が反映されるようにします。
type	属性	このインスタンスで設定された修復のタイプを提供します。
config	要素	設定時に Web インターフェイスの修復設定フィールドに入力したデータが含まれます。

たとえば、config 要素（2-8 ページ）で示された例の module.template ドキュメントは、次のように続きます。

```
<remediation_type name="acl_insert">
<display_name>ACL Insertion</display_name>
<policy_event_data>
    <pe_item>src_ip_addr</pe_item>
    <pe_item>src_port</pe_item>
    <pe_item>src_protocol</pe_item>
    <pe_item>dest_ip_addr</pe_item>
    <pe_item>dest_port</pe_item>
    <pe_item>dest_protocol</pe_item>
</policy_event_data>
<config_template>
    <integer>
        <name>acl_num</name>
        <display_name>ACL Number</display_name>
    </integer>
</config_template>
</remediation_type>
```

作成されたインスタンスに修復を追加できる [Instance Detail] ページには、修復タイプ [ACL Insertion] があります。インスタンスに [ACL Insertion] を追加すると、ユーザは名前フィールド（instance.conf のその修復要素の名前属性値を入力する）と [ACL Number] とラベル付けされたフィールド（整数値を受け入れる）が含まれるページに移動します。

ユーザがこの修復を AdminInstance インスタンスに追加し、次の値を指定したとします。

表 2-12　サンプル値

フィールド	値
Remediation Name	AdminRemediation
ACL Number	55

ユーザがサンプルの設定値を保存したときに書き込まれた `instance.conf` ドキュメントは（config 要素（2-8 ページ）の例で示されたセクションの後）、次のように続きます。

```
<remediation name="AdminRemediation" type="acl_insert">
<config>
    <integer="acl_num">55</integer>
</config>
</remediation>
```

これ以上修復がインスタンスに追加されない場合、`instance.conf` は、この時点で `</instance>` で終了されることに注意してください。

モジュールによって返されるデータ

修復モジュールは、戻りコードと呼ばれる終了ステータス コードを Firepower Management Center に返します。Firepower Management Center Web インターフェイスの修復のテーブル ビューには、起動した各修復の結果のメッセージが表示されます。修復プログラムからの戻りコードによって、表示される結果メッセージが決まります。

戻りコードは、次の表で定義されているように、0 ～ 255 の範囲の整数（両端の値を含む）である必要があります。

表 2-13　戻りコードの範囲

範囲	使用目的
0 ～ 128	シスコが事前に定義する戻りコード用に予約されています
129 ～ 255	カスタム修復に使用できます

事前定義されたコードのリストおよびカスタム コードを作成する方法については、終了ステータスの定義（3-21 ページ）を参照してください。

モジュールのパッケージ化とインストール

修復 API では、修復モジュールをパッケージ化する必要があります。モジュールを構成するファイルは、gzip された tar ファイルで提供する必要があります。

詳細については、次の項を参照してください。

- モジュールのパッケージ化（2-12 ページ）では、アップロードしたり、インストールしたりできるように、バイナリ、`module.template` ファイルをパッケージ化するうえで役立つヒントが提供されています。
- モジュールのインストール（2-12 ページ）では、修復モジュールを Firepower Management Center にインストールする方法が示されています。

モジュールのパッケージ化

インストールできるように修復ファイルをパッケージ化する際、次の点に注意してください。

- 修復モジュールは、インストールする前に、gzip された tarball（`.tar.gz` または `.tgz`）でパッケージ化する必要があります。
- モジュールをインストールする際、パッケージは `/var/sf/remediation/remediation_directory` に抽出されます。ここで、`remediation_directory` は、モジュールの `module` 要素の `name` 属性と `version` 要素のデータを組み合わせたものです。

  たとえば、Firepower Management Center に付属するデフォルトの修復モジュールの 1 つは、Cisco PIX Shun モジュールです。このモジュールは `/var/sf/remediation/cisco_pix_1.0` にあります。
- 抽出する際、修復モジュールの `module.template` ドキュメントは、そのモジュール パッケージを含めるために作成されたディレクトリの最上位に配置する必要があります。
- 修復のインスタンスが作成されると、そのインスタンスの名前が付けられた、モジュール ディレクトリに作成されたディレクトリに保存されます。

  たとえば、Cisco PIX Shun モジュールのインスタンスは、`/var/sf/remediation/cisco_pix_1.0/PIX_01` と `/var/sf/remediation/cisco_pix_1.0/PIX_02` に配置されます。

たとえば、firewall.tgz でパッケージ化され、`module.template` で `firewall` として名前が付けられた（バージョン値は `1.0`）モジュールをアップロードしたり、インストールしたりします。システムは、ディレクトリ `/var/sf/remediation/firewall_1.0` にモジュールをインストールします。このディレクトリには、module.template ファイルおよびプログラム バイナリが含まれます。インスタンスを修復モジュールに追加し、block_tokyo という名前を付けると、次のディレクトリが作成されます。

```
/var/sf/remediation/firewall_1.0/block_tokyo
```

`block_tokyo` の `instance.conf` ファイルをそのディレクトリに配置します。

モジュールのインストール

修復モジュールを正常にパッケージ化したら、[Modules] ページを使用してインストールします。

新しいモジュールを修復 API にインストールするには、次の手順を実行します。

1. [Policies] > [Actions] > [Modules] を選択します。

   [Installed Remediation Modules] ページが表示されます。
2. [Browse] をクリックして、カスタム修復モジュールを含む tar.gz ファイルを保存した場所に移動します。
3. [Install] をクリックします。

   カスタム修復モジュールがインストールされます。
4. [Policies] > [Actions] > [Modules] を選択します。

   [Installed Remediation Modules] テーブルに、インストールしたモジュールがリストされます。[Module Name]、[Version]、[Description] カラムは `module.template` ファイルに定義されている情報と一致します。
5. 『*Firepower System User Guide*』の説明に従って、新規モジュールのインスタンスを追加し、各インスタンスに修復を関連付けます。

[Modules] ページで、Firepower Management Center にインストールされた修復モジュールを確認できます。リストにカスタム修復モジュールとシスコが提供するモジュールが表示されます。また、カスタム モジュールを削除することもできます。

修復 API からモジュールを表示または削除するには、次の手順を実行します。

1. [Policies] > [Actions] > [Modules] を選択します。

   [Installed Remediation Modules] ページが表示されます。

2. 次のいずれかの操作を実行します。

- [View] アイコンをクリックして、モジュールを表示します。

  [Module Detail] ページが表示されます。

- 削除するモジュールの横にある [Delete] アイコンをクリックします。シスコで提供されるデフォルトのモジュールは削除**できません**。

  修復モジュールは修復 API から削除されます。

修復サブシステムとの通信

修復モジュールは、正常に機能を実行するために、Firepower Management Center 修復サブシステムから情報を受信する必要があります。モジュールが `module.template` と呼ばれる XML ファイルで受信する情報を設定します。これを設定しないと、修復サブシステムは修復モジュールと対話できません。

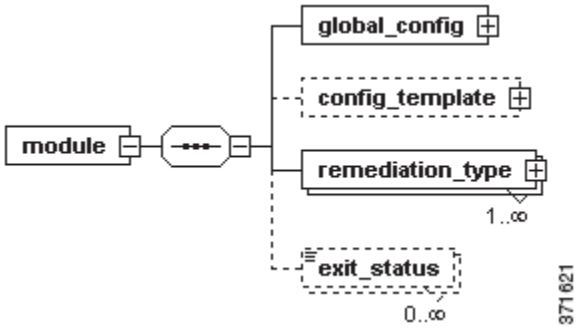
`module.template` XML ファイルには、以下を指定できます。

- 修復モジュールの名前とバージョン、短い説明テキスト、修復プログラムのバイナリ ファイル名などの一連のモジュール レベルの宣言
- ユーザが Firepower Management Center ユーザ インターフェイスで修復インスタンスを設定する際に、モジュールがユーザに要求する情報
- モジュールによって実行される特定の修復アクション（修復タイプとも呼ばれます）、および各修復タイプが必要とする相関イベント データ
- 修復プログラムが Firepower Management Center に返すカスタム戻りコードと終了ステータス メッセージ

修復モジュールの `module.template` を記述する前に、`module.template` スキーマ（`module.template.xsd`）について理解しておく必要があります。スキーマは要素（またはデータを格納するために使用するタグ）と属性（または要素に格納されたデータを変更するために使用するデータ）を定義します。これらを使用して、修復サブシステムに情報を提供できます。`module.template` スキーマは DC の `/etc/sf/remediation/module.template.vsd` にあります。

`module.template` における最上位の要素は、`module` です。この要素の `name` 属性を使用して修復モジュールの名前を指定します。`name` 属性は必須で、1 ～ 64 文字のアルファベット文字から成る文字列値を受け入れます。

注意：モジュールの `name` 属性値には、スペースは使用できません。また、句読点も使用できません（アンダースコア（_）とハイフン（-）を除く）。



一部の XML エディタでは、`module.template` スキーマを読み取り、名前空間、スキーマ宣言、最上位要素、子要素、属性を持つ `module.template` ファイルを自動的に生成することができます。このようなエディタを使用しない場合は、子要素を手動で追加する必要があります。

注意：名前空間とスキーマ ロケーションを自動生成するように XML エディタを設定した場合、インストール パッケージに module.template の最終バージョンを含める前に、その行を削除する必要があります。

次の例には、name 属性のみが定義された module 要素が示されています。

```
<module name="example_module">
<global_config>
    <display_name/>
    <version/>
    <binary/>
</global_config>
<remediation_type name="">
    <display_name/>
</remediation_type>
</module>
```

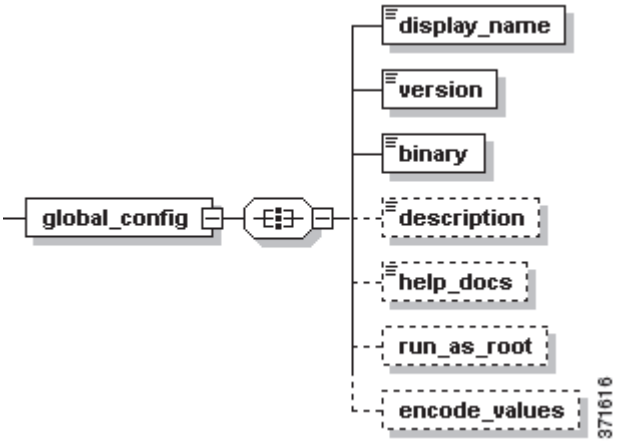
module.template の残りの記述方法の詳細については、次の各項を参照してください。

- グローバル設定の定義（3-2 ページ）には、[Modules] ページのモジュールに表示される名前、モジュールのバージョン、バイナリの場所、その説明を定義するための global_config 要素の使用方法が示されています。
- 設定テンプレートの定義（3-4 ページ）には、モジュールがユーザに Web インターフェイスから指定するよう要求する設定情報を定義するための config_template 要素の使用方法が示されています。
- グローバル設定の定義（3-2 ページ）には、モジュールが起動できる修復と各モジュールが必要とする相関イベント データを定義するための remediation_type 要素の使用方法が示されています。
- 終了ステータスの定義（3-21 ページ）には、モジュールが修復サブシステムに返すカスタム終了ステータスを定義するための exit_status 要素の使用方法が示されています。

グローバル設定の定義

module.template の最初の必須セクションでは、グローバル設定情報を定義する global_config 要素を使用します。これらの属性には、モジュール名と説明が含まれます。これらは、Firepower Management Center ユーザ インターフェイスの [Modules] ページに表示される修復モジュールのリストに表示されます。グローバル情報には、モジュールのバージョンと、修復がトリガーされたときに実行される実行可能プログラムの場所も含まれます。

module.template スキーマ図の次の部分は、global_config 要素の子要素を示しています。



次の表に、global_config 要素で使用可能な子要素を示します。

表 3-1　　global_config の子要素

名前	説明	必須かどうか
display_name	[Modules] ページのこの修復モジュールに対して表示される名前を指定します。表示名には、英数字とスペースのみを使用でき、1 ～ 127 文字にする必要があります。この表示名は、修復モジュール間で一意である必要があります。	yes
version	修復モジュールのバージョンを指定します。この値は [Modules] ページに表示されます。version 要素の値は数字で始まり、数字で終了する必要がありますが、ピリオド (.) 文字を含めることもできます。 注：module 要素の name 属性と version 要素のデータの組み合わせは、修復モジュール間で一意である必要があります。	yes
binary	修復モジュールを構成するバイナリの UNIX ファイル名を指定します。	yes
description	修復モジュールの説明とその使用可能な修復の説明を示します。description 要素は [Modules] ページに表示されます。255 文字を超える説明は切り捨てられます。	yes
run_as_root	修復モジュールがインストールされている シスコアプライアンスでそのモジュールをルートとして実行することを許可するフラグを設定します。 注意：シスコでは、どうしても必要な場合にのみ、この要素を使用することを推奨します。	no
encode_values	ユーザ入力を HTML エンコードするフラグを設定します。これにより、ユーザは、設定しなければ XML プロセッサに意図しない仕方で解釈されてしまう文字を入力できます。 注：この要素を使用する場合、修復モジュールは入力処理の一部として HTML デコードを処理**しなければなりません**。	no

module.template ファイルのグローバル設定の一部を説明する次の XML コードがあるとします。

```
<global_config>
<display_name>My Firewall</display_name>
<binary>firewall_block.pl</binary>
<description>Dynamically apply firewall rules to my firewall.</description>
<version>1.0</version>
<run_as_root/>
</global_config>
```

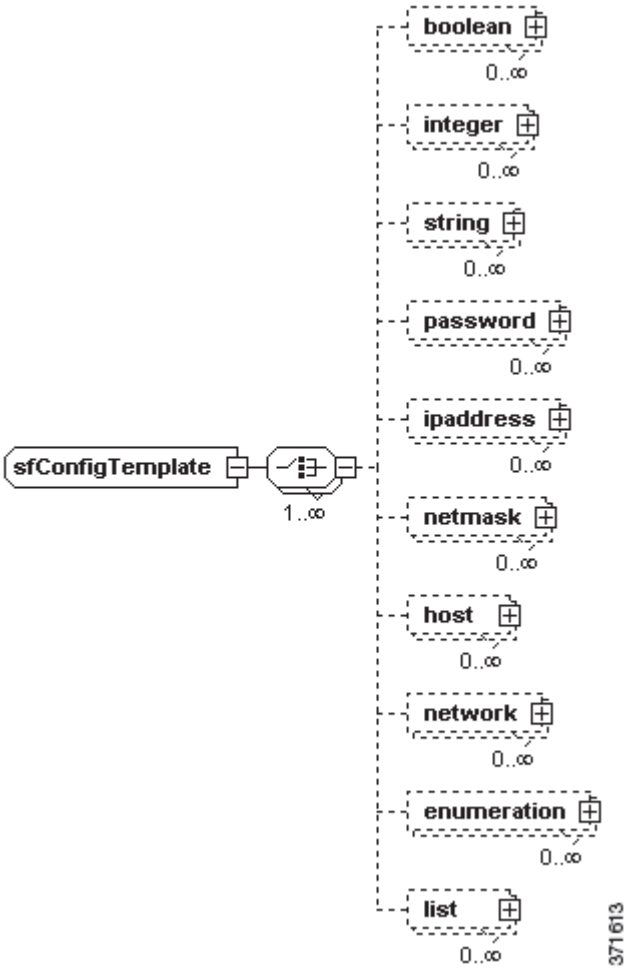
この例では、修復モジュールは、Web インターフェイスに「My Firewall」という名前で表示されます。Firepower Management Center を使用してインストールした firewall_block.pl という名前のプログラムのバージョン 1.0 を実行します（詳細については、モジュールのパッケージ化とインストール（2-11 ページ）を参照してください）。プログラムはファイアウォール ルールを特定のファイアウォールに動的に適用して、Firepower Management Center でルートとして実行します。

設定テンプレートの定義

module 要素の config_template 子要素は、この修復モジュールが実行するインスタンスを設定する際に、ユーザが入力しなければならない情報のタイプを指定します（インスタンス設定データ（2-8 ページ）を参照してください）。ユーザは、Firepower Management Center ユーザ インターフェイスからこの要素で指定した情報を指定します。各 module 要素には、直接の子 config_template 要素を 1 つだけ含めることができ、この要素は設定されているすべてのインスタンスに適用されます。

ただし、module.template 内の各 remediation_type 要素にも、config_template 子要素を含めることができます。remediation_type の下の config_template 子要素では、異なる修復タイプにそれぞれ指定しなければならない情報を定義することができます。ユーザは、module 部分の config_template 要素を使用して一般的なインスタンス レベル フィールドを設定し、そしてオプションで、インスタンスによって実行される修復タイプに固有の config_template フィールドの追加のセットを設定する必要があります。詳細については、修復タイプの定義（3-19 ページ）を参照してください。

次の図に、config_template 要素で使用可能な子要素を示します。



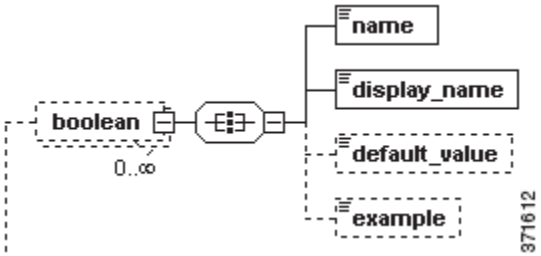
config_template 要素によって、Web インターフェイスに複数の基本的なフィールド タイプを表示することができます。修復モジュールのユーザから収集するために必要なデータに基づいて、使用する config_template 子要素を選択します。config_template のすべての子要素はオプションで、config_template 要素内で必要に応じて何度でも使用できます。フィールドは、config_template 要素に含めた順序で Web インターフェイスに表示されます。

Web インターフェイスのインスタンス設定や修復設定ページの設定情報を収集するために使用できる、フィールドを表す子要素の詳細については、次の各項を参照してください。

boolean 要素

`config_template` で使用するそれぞれの `boolean` 要素は、ユーザが Web インターフェイスで選択できる true/false の選択肢を表します（[On] と [Off] のラベル付けされたオプション ボタンのセットとして表示されます）。要素の `required` 属性を False に設定すると、[Not Selected] とラベル付けされた追加のオプション ボタンが使用できるようになります。

`module.template` スキーマ図の次の部分は、`boolean` 要素の子要素を示しています。



`boolean` 要素の発生の子要素を設定する際、使用可能な子要素をそれぞれ一度だけ使用できます。次の表に、`boolean` 要素で使用可能な子要素を示します。

表 3-2　boolean 属性と子要素

名前	タイプ	説明	必須かどうか
`required`	属性	フィールドに値を指定することが任意であるかどうかを示します。 この属性のデフォルト値は `true` です。この属性は、使用しなくてもかまいません。したがって、この属性を使用しない場合（明示的にその値を `true` に設定した場合）、ユーザは [On] または [Off] のいずれかを選択する必要があります。属性の値を `false` に設定した場合、Web インターフェイスに、選択はオプションであることが示されます。	no

表 3-2　boolean 属性と子要素（続き）

名前	タイプ	説明	必須かどうか
name	要素	フィールドに入力した値の修復モジュールへのコンテキストを指定します。名前には、スペースを含めることはできません。英数字およびアンダースコア（_）とハイフン（-）の文字を含めることができます。名前は、モジュール内で一意である必要があります。	yes
display_name	要素	このフィールドの Web インターフェイスのラベルを指定します。	yes
default_value	要素	このフィールドのデフォルト値を指定します。Web インターフェイスのユーザが値を指定しない場合、修復プログラムはデフォルトでこの値を使用します。	no

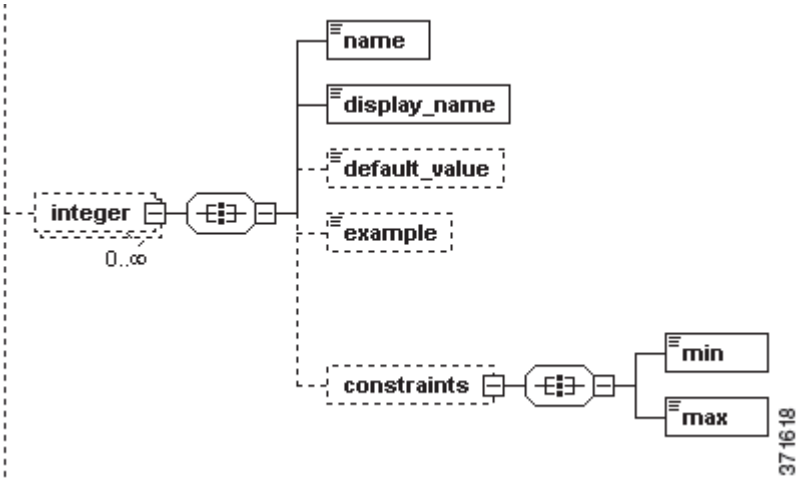
config_template 要素定義の次の部分は、Web インターフェイスに [Enabled?] とラベル付けされたフィールドが表示されることを示します。これは、[On] または [Off] という 2 つの選択肢を提供します。この選択は、デフォルトで true に設定されます。つまり、[On] とラベル付けされたオプション ボタンが事前に選択されています。

```
<boolean>
<name>process_enabled</name>
<display_name>Enabled?</display_name>
<default_value>true</default_value>
</boolean>
```

integer 要素

config_template で使用する各 integer 要素は、整数値を受け入れる Web インターフェイスのフィールドを表します。

次の図に、integer 要素の子要素と孫要素を示します。



次の表に、integer 要素で使用可能な子要素を示します。

表 3-3　integer 属性、子要素、および孫要素

名前	タイプ	説明	必須かどうか
required	属性	ユーザがフィールドの値を指定する必要があるかどうかを示します。 この属性のデフォルト値は true です。この属性は、使用しなくてもかまいません。したがって、この属性を使用しない場合（明示的にその値を true に設定した場合）、ユーザは値を指定する必要があります。属性の値を false に設定した場合、Web インターフェイスに、値の指定はオプションであることが示されます。	no
name	要素	フィールドに入力した値の修復モジュールへのコンテキストを指定します。名前には、スペースを含めることはできません。英数字およびアンダースコア（_）とハイフン（-）の文字を含めることができます。名前は、モジュール内で一意である必要があります。	yes
display_name	要素	このフィールドの Web インターフェイスのラベルを指定します。	yes
default_value	要素	このフィールドのデフォルト値を指定します。Web インターフェイスのユーザが値を指定しない場合、修復プログラムはデフォルトでこの値を使用します。	no
example	要素	修復モジュールが受信すると予期する入力例を提供します。 注：この値は、Web インターフェイスには表示されません。	no
constraints	要素	ユーザがこのフィールドに入力できる値を、指定した最小値と最大値の間（両端の値を含む）になるように抑制します。 constraints 要素には、min と max の 2 つの子要素があります。それぞれはオプションで、整数値を受け入れる 1 回のみ発生する子要素です。	no

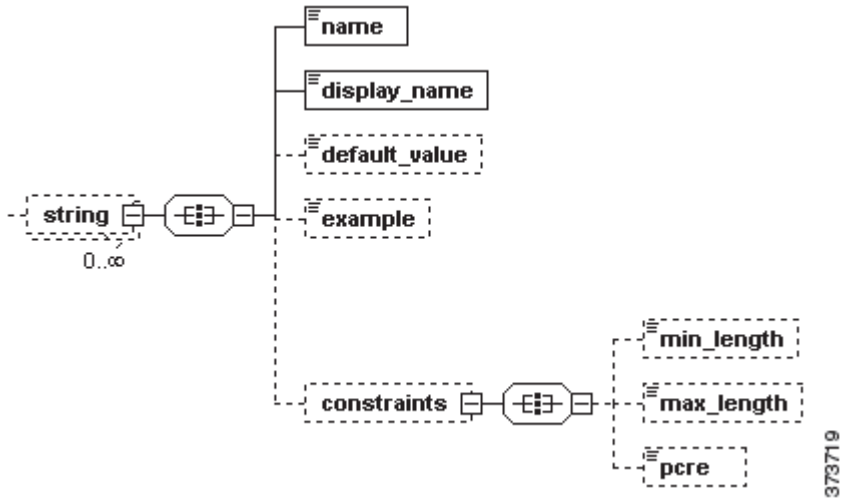
config_template 要素定義の次の部分は、Web インターフェイスに [Rate] とラベル付けされたフィールドが表示されることを示します。このフィールドは、0 ～ 500 までの整数値を受け入れ、デフォルト値は 430 です。

```
<integer>
<name>rate</name>
<display_name>Rate</display_name>
<default_value>430</default_value>
<constraints>
    <min>0</min>
    <max>500</max>
</constraints>
</integer>
```

string 要素

config_template で使用する各 string 要素は、文字列値を受け入れる Web インターフェイスのフィールドを表します。

次の図に、string 要素インスタンスの子要素を示します。



次の表に、string 要素で使用可能な子要素を示します。

表 3-4　string 属性、子要素、および孫要素

名前	タイプ	説明	必須かどうか
required	属性	ユーザがフィールドの値を指定する必要があるかどうかを示します。 この属性のデフォルト値は true です。この属性は、使用しなくてもかまいません。したがって、この属性を使用しない場合（明示的にその値を true に設定した場合）、ユーザは値を指定する必要があります。属性の値を false に設定した場合、Web インターフェイスに、値の指定はオプションであることが示されます。	no
name	要素	フィールドに入力した値の修復モジュールへのコンテキストを指定します。名前には、スペースを含めることはできません。英数字およびアンダースコア（_）とハイフン（-）の文字を含めることができます。名前は、モジュール内で一意である必要があります。	yes
display_name	要素	このフィールドの Web インターフェイスのラベルを指定します。	yes
default_value	要素	このフィールドのデフォルト値を指定します。Web インターフェイスのユーザが値を指定しない場合、修復プログラムはデフォルトでこの値を使用します。	no
example	要素	修復モジュールが受信すると予期する入力例を提供します。 注：この値は、Web インターフェイスには表示されません。	no
constraints	要素	ユーザがこのフィールドに入力できる値を抑制します。 constraints 要素には、min_length、max_length、pcre という 3 つの子要素があります。min_length と max_length 要素はオプションで、1 回のみ発生する子要素です。整数値を受け入れ、文字列値の長さの許容範囲を指定します。pcre 要素はオプションで、追加の制約を指定する Perl と互換性のある正規表現を指定するために使用します。	no

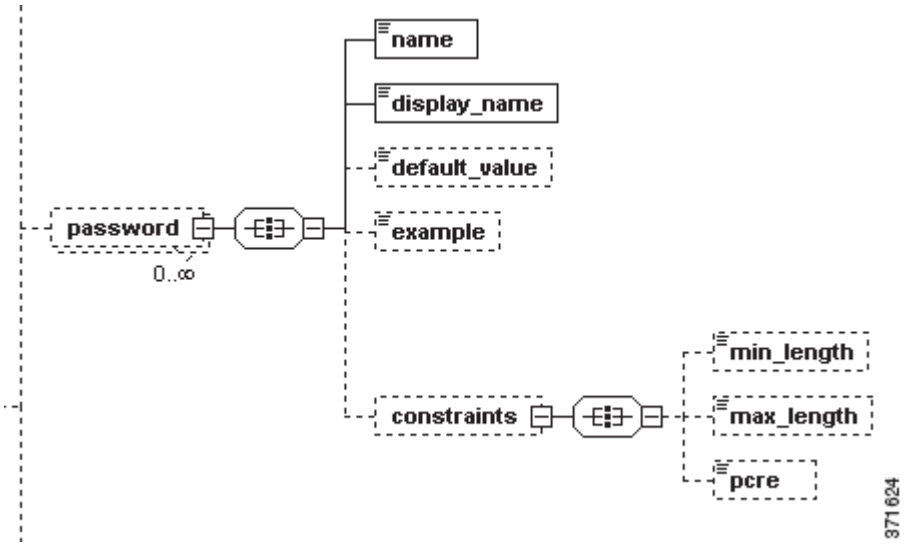
config_template 要素定義の次の部分は、Web インターフェイスに [Username] とラベル付けされたフィールドが表示されることを示します。このフィールドは、8 文字以上のスペースを含まない文字列値を受け入れます。

```
<string>
    <name>user_name</name>
    <display_name>Username</display_name>
    <constraints>
        <min_length>8</min_length>
        <pcre>\S+</pcre>
    </constraints>
</string>
```

password 要素

config_template で使用する各 password 要素は、英数字からなる文字列を受け入れる Web インターフェイスのフィールドを表します。

次の図に、password 要素インスタンスの子要素と孫要素を示します。



次の表に、password 要素で使用可能な子要素を示します。

表 3-5　password 属性、子要素、および孫要素

名前	タイプ	説明	必須かどうか
required	属性	ユーザがフィールドの値を指定する必要があるかどうかを示します。 この属性のデフォルト値は true です。この属性は、使用しなくてもかまいません。したがって、この属性を使用しない場合（明示的にその値を true に設定した場合）、ユーザは値を指定する必要があります。属性の値を false に設定した場合、Web インターフェイスに、値の指定はオプションであることが示されます。	no
name	要素	フィールドに入力した値の修復モジュールへのコンテキストを指定します。名前には、スペースを含めることはできません。英数字およびアンダースコア（_）とハイフン（-）の文字を含めることができます。名前は、モジュール内で一意である必要があります。	yes
display_name	要素	このフィールドの Web インターフェイスのラベルを指定します。	yes

表 3-5　password 属性、子要素、および孫要素（続き）

名前	タイプ	説明	必須かどうか
default_value	要素	このフィールドのデフォルト値を指定します。Web インターフェイスのユーザが値を指定しない場合、修復プログラムはデフォルトでこの値を使用します。	no
example	要素	修復モジュールが受信すると予期する入力例を提供します。 注：この値は、Web インターフェイスには表示されません。	no
constraints	要素	ユーザがこのフィールドに入力できる値を抑制します。 constraints 要素には、min_length、max_length、pcre という 3 つの子要素があります。min_length と max_length 要素はオプションで、1 回のみ発生する子要素です。整数値を受け入れ、パスワード値の長さの許容範囲を指定します。pcre 要素はオプションで、追加の制約を指定する Perl と互換性のある正規表現を指定するために使用します。	no

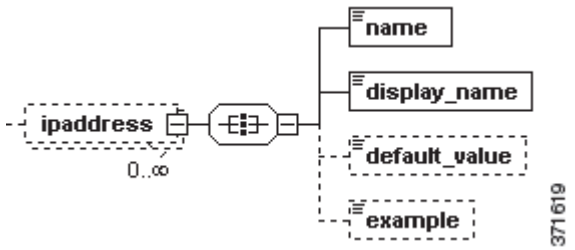
config_template 要素定義の次の部分は、Web インターフェイスに [Login Password] とラベル付けされたフィールドが表示されることを示します。このフィールドは、6 ～ 12 文字の英数字文字列を受け入れます。

```
<password>
    <name>login_password</name>
    <display_name>Login Password</display_name>
    <constraints>
        <min_length>6</min_length>
        <max_length>12</max_length>
    </constraints>
</password>
```

ipaddress 要素

config_template で使用する各 ipaddress 要素は、単一の IP アドレスを受け入れる Web インターフェイスのフィールドを表します。IP アドレスは、ドット区切りの 4 つの数字列（例：1.1.1.1）の完全な形式で入力されます。

次の図に、ipaddress 要素の子要素を示します。



ipaddress 要素の発生の子要素を設定する際、使用可能な子要素をそれぞれ一度だけ使用できます。次の表に、ipaddress 要素で使用可能な子要素を示します。

表 3-6　ipaddress 属性と子要素

名前	タイプ	説明	必須かどうか
required	属性	ユーザがフィールドの値を指定する必要があるかどうかを示します。 この属性のデフォルト値は true です。この属性は、使用しなくてもかまいません。したがって、この属性を使用しない場合（明示的にその値を true に設定した場合）、ユーザは値を指定する必要があります。属性の値を false に設定した場合、Web インターフェイスに、値の指定はオプションであることが示されます。	no
name	要素	フィールドに入力した値の修復モジュールへのコンテキストを指定します。名前には、スペースを含めることはできません。英数字およびアンダースコア（_）とハイフン（-）の文字を含めることができます。名前は、モジュール内で一意である必要があります。	yes
display_name	要素	このフィールドの Web インターフェイスのラベルを指定します。	yes
default_value	要素	このフィールドのデフォルト値を指定します。	no
example	要素	修復モジュールが受信すると予期する入力例を提供します。 注：この値は、Web インターフェイスには表示されません。	no

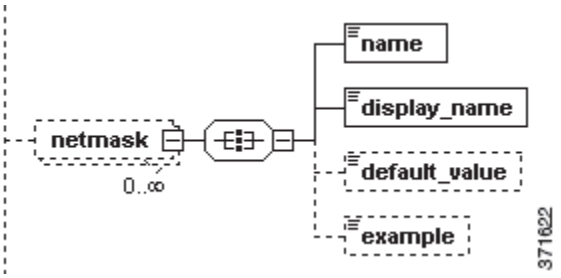
config_template 要素定義の次の部分は、Web インターフェイスに [Mail Server] とラベル付けされたフィールドが表示されることを示します。このフィールドは、単一の IP アドレスを受け入れます。

```
<ipaddress>
<name>mail_server</name>
<display_name>Mail Server</display_name>
</ipaddress>
```

netmask 要素

config_template で使用する各 netmask 要素は、ネットマスク値を受け入れる Web インターフェイスのフィールドを表します。ネットマスク値は、ドット区切りの 4 つの数字列（255.255.255.255）または CIDR マスク（/8）で表すことができます。

次の図に、netmask 要素の子要素を示します。



netmask 要素の発生の子要素を設定する際、使用可能な子要素をそれぞれ一度だけ使用できます。次の表に、netmask 要素で使用可能な子要素を示します。

表 3-7　netmask 属性と子要素

名前	タイプ	説明	必須かどうか
required	属性	ユーザがフィールドの値を指定する必要があるかどうかを示します。 この属性のデフォルト値は true です。この属性は、使用しなくてもかまいません。したがって、この属性を使用しない場合（明示的にその値を true に設定した場合）、ユーザは値を指定する必要があります。属性の値を false に設定した場合、Web インターフェイスに、値の指定はオプションであることが示されます。	no
name	要素	フィールドに入力した値の修復モジュールへのコンテキストを指定します。名前には、スペースを含めることはできません。英数字およびアンダースコア（_）とハイフン（-）の文字を含めることができます。名前は、モジュール内で一意である必要があります。	yes
display_name	要素	このフィールドの Web インターフェイスのラベルを指定します。	yes
default_value	要素	このフィールドのデフォルト値を指定します。Web インターフェイスのユーザが値を指定しない場合、修復プログラムはデフォルトでこの値を使用します。	no
example	要素	修復モジュールが受信すると予期する入力例を提供します。 注：この値は、Web インターフェイスには表示されません。	no

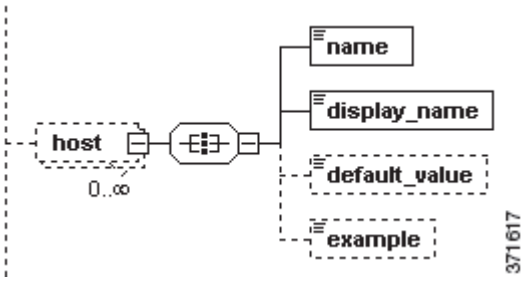
config_template 要素定義の次の部分は、Web インターフェイスに [Netmask] とラベル付けされたフィールドが表示されることを示します。このフィールドは、ドット区切りの 4 つの数字列または CIDR マスクで表されるネットマスク値を受け入れ、デフォルト値は 255.255.255.255 です。

```
<netmask>
<name>netmask</name>
<display_name>Netmask</display_name>
<default_value>255.255.255.0</default_value>
</netmask>
```

host 要素

config_template で使用する各 host 要素は、単一の IP アドレスまたは文字列を受け入れる Web インターフェイスのフィールドを表します。

次の図に、host 要素の子要素を示します。



host 要素の発生の子要素を設定する際、使用可能な子要素をそれぞれ一度だけ使用できます。次の表に、host 要素で使用可能な子要素と属性を示します。

表 3-8　host 属性と子要素

名前	タイプ	説明	必須かどうか
required	属性	ユーザがフィールドの値を指定する必要があるかどうかを示します。 この属性のデフォルト値は true です。この属性は、使用しなくてもかまいません。したがって、この属性を使用しない場合（明示的にその値を true に設定した場合）、ユーザは値を指定する必要があります。属性の値を false に設定した場合、Web インターフェイスに、値の指定はオプションであることが示されます。	no
name	要素	フィールドに入力した値の修復モジュールへのコンテキストを指定します。名前には、スペースを含めることはできません。英数字およびアンダースコア（_）とハイフン（-）の文字を含めることができます。名前は、モジュール内で一意である必要があります。	yes
display_name	要素	このフィールドの Web インターフェイスのラベルを指定します。	yes
default_value	要素	このフィールドのデフォルト値を指定します。Web インターフェイスのユーザが値を指定しない場合、修復プログラムはデフォルトでこの値を使用します。	no
example	要素	修復モジュールが受信すると予期する入力例を提供します。 注：この値は、Web インターフェイスには表示されません。	no

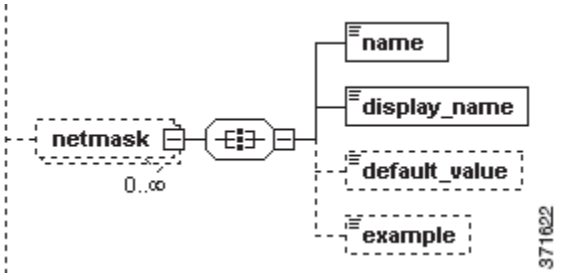
config_template 要素定義の次の部分は、Web インターフェイスに [Host Name] とラベル付けされたフィールドが表示されることを示します。このフィールドは、IP アドレスまたは文字列を受け入れます。Web インターフェイスには、「192.10.1.3」というサンプル テキストも表示されます。

```
<host>
<name>hostname</name>
<display_name>Host Name</display_name>
<example>192.10.1.3</example>
</host>
```

network 要素

config_template 内で使用する各 network 要素は、Web インターフェイスのフィールドを表します。ネットワーク フィールドは、IP アドレス（単一の IP アドレス、つまり /32 ネットマスクを持つ IP アドレスを想定します）または CIDR ブロックを受け入れます。

次の図に、network 要素の子要素を示します。



network 要素の発生の子要素を設定する際、使用可能な子要素をそれぞれ一度だけ使用できます。次の表に、network 要素で使用可能な子要素と属性を示します。

表 3-9　network 属性と子要素

名前	タイプ	説明	必須かどうか
required	属性	ユーザがフィールドの値を指定する必要があるかどうかを示します。 この属性のデフォルト値は true です。この属性は、使用しなくてもかまいません。したがって、この属性を使用しない場合（明示的にその値を true に設定した場合）、ユーザは値を指定する必要があります。属性の値を false に設定した場合、Web インターフェイスに、値の指定はオプションであることが示されます。	no
name	要素	フィールドに入力した値の修復モジュールへのコンテキストを指定します。名前には、スペースを含めることはできません。英数字およびアンダースコア（_）とハイフン（-）の文字を含めることができます。名前は、モジュール内で一意である必要があります。	yes
display_name	要素	このフィールドの Web インターフェイスのラベルを指定します。	yes
default_value	要素	このフィールドのデフォルト値を指定します。Web インターフェイスのユーザが値を指定しない場合、修復プログラムはデフォルトでこの値を使用します。	no
example	要素	修復モジュールが受信すると予期する入力例を提供します。 注：この値は、Web インターフェイスには表示されません。	no

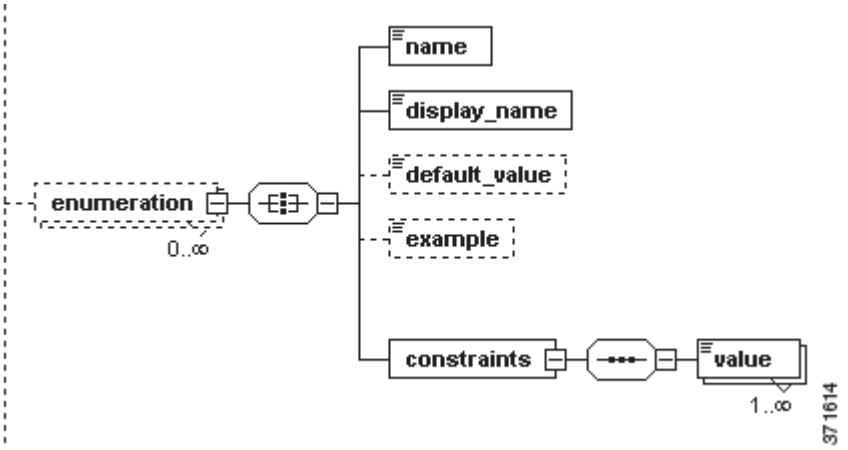
config_template 要素定義の次の部分は、Web インターフェイスに [Monitored Network] とラベル付けされたフィールドが表示されることを示します。このフィールドは、/32 IP アドレスまたは IP アドレスとネットマスク値を受け入れ、デフォルト値は 192.168.1.0/24 になります。

```
<network>
<name>monitored_network</name>
<display_name>Monitored Network</display_name>
<default_value>192.168.1.0/24</default_value>
</network>
```

enumeration 要素

config_template で使用する各 enumeration 要素は、Web インターフェイスに表示される文字列のドロップダウン リストを表します。ユーザはこのリストから 1 つの値を選択できます。

次の図に、enumeration 要素の子要素と孫要素を示します。



次の表に、enumeration 要素で使用可能な子要素と属性を示します。

表 3-10　enumeration 属性、子要素、および孫要素

名前	タイプ	説明	必須かどうか
required	属性	ユーザがフィールドの値を指定する必要があるかどうかを示します。 この属性のデフォルト値は true です。この属性は、使用しなくてもかまいません。したがって、この属性を使用しない場合（明示的にその値を true に設定した場合）、ユーザは値を指定する必要があります。属性の値を false に設定した場合、Web インターフェイスに、値の指定はオプションであることが示されます。	no
name	要素	フィールドに入力した値の修復モジュールへのコンテキストを指定します。名前には、スペースを含めることはできません。英数字およびアンダースコア（_）とハイフン（-）の文字を含めることができます。名前は、モジュール内で一意である必要があります。	yes
display_name	要素	このフィールドの Web インターフェイスのラベルを指定します。	yes
default_value	要素	このフィールドのデフォルト値を指定します。Web インターフェイスのユーザが値を指定しない場合、修復プログラムはデフォルトでこの値を使用します。	no
example	要素	修復モジュールが受信すると予期する入力例を提供します。 注：この値は、Web インターフェイスには表示されません。	no
constraints	要素	ユーザがこのフィールドに入力できる値を指定します。 constraints 要素には、value という必須の子要素が 1 つあります。この要素は、ユーザの選択を表す文字列を受け入れます。複数の value 要素を使用して、複数の選択肢をユーザに提示します。	yes

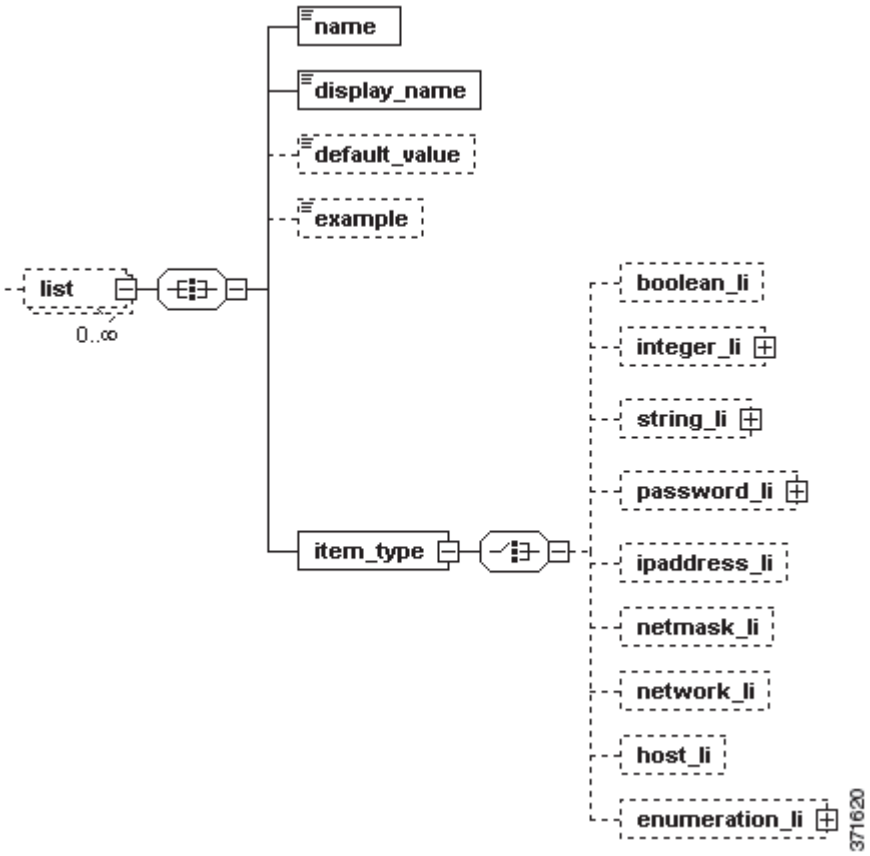
config_template 要素定義の次の部分は、Web インターフェイスに [Day] とラベル付けされたフィールドが表示されることを示します。このフィールドで、ユーザは提示される値（Monday、Tuesday、Wednesday、Thursday、Friday）の 1 つを選択できます。

```
<enumeration>
<name>day</name>
<display_name>Day</display_name>
<constraints>
    <value>Monday</value>
    <value>Tuesday</value>
    <value>Wednesday</value>
    <value>Thursday</value>
    <value>Friday/value>
</constraints>
</enumeration>
```

list 要素

config_template で使用する各 list 要素は、ユーザが 1 行ごとに値のリストを入力できる Web インターフェイスのフィールドを表します。そのタイプは、必須の item_type 子要素で指定します。

次の図に、list 要素の子要素と孫要素を示します。



次の表に、list 要素で使用可能な子要素を示します。

表 3-11　list 属性と子要素

名前	タイプ	説明	必須かどうか
required	属性	ユーザがフィールドの値を指定する必要があるかどうかを示します。 この属性のデフォルト値は true です。この属性は、使用しなくてもかまいません。したがって、この属性を使用しない場合（明示的にその値を true に設定した場合）、ユーザは値を指定する必要があります。属性の値を false に設定した場合、Web インターフェイスに、値の指定はオプションであることが示されます。	no
name	要素	フィールドに入力した値の修復モジュールへのコンテキストを指定します。名前には、スペースを含めることはできません。英数字およびアンダースコア（_）とハイフン（-）の文字を含めることができます。名前は、モジュール内で一意である必要があります。	yes
display_name	要素	このフィールドの Web インターフェイスのラベルを指定します。	yes
default_value	要素	このフィールドのデフォルト値を指定します。Web インターフェイスのユーザが値を指定しない場合、修復プログラムはデフォルトでこの値を使用します。	no

表 3-11　list 属性と子要素（続き）

名前	タイプ	説明	必須かどうか
example	要素	修復モジュールが受信すると予期する入力例を提供します。 注：この値は、Web インターフェイスには表示されません。	no
item_type	要素	このフィールドに表示できる値のタイプを指定します。値のタイプは子要素で指定されます。有効な子要素を次にリストします。	no

次に、item_type 要素で使用できる子要素を示します。これは、config_template 要素の子要素と同様ですが、唯一異なるのは item_type 子要素が required 属性を使用しない点です。item_type 要素の各インスタンスは、1 つの子要素だけを使用できます。

- boolean_li は、リストが複数のブール値を受け入れることを示します（boolean 要素（3-5 ページ）を参照してください）。
- integer_li は、リストが複数の整数値を受け入れることを示します（integer 要素（3-6 ページ）を参照してください）。
- string_li は、リストが複数の文字列値を受け入れることを示します（string 要素（3-7 ページ）を参照してください）。
- password_li は、リストが複数のパスワード値を受け入れることを示します（password 要素（3-9 ページ）を参照してください）。
- ipaddress_li は、リストが複数の IP アドレス値を受け入れることを示します（ipaddress 要素（3-10 ページ）を参照してください）。
- network_li は、リストが複数のネットワーク値を受け入れることを示します（network 要素（3-13 ページ）を参照してください）。
- netmask_li は、リストが複数のネットマスク値を受け入れることを示します（netmask 要素（3-11 ページ）を参照してください）。
- host_li は、リストが複数のホスト値を受け入れることを示します（host 要素（3-12 ページ）を参照してください）。
- enumeration_li は、リストが enumeration_li 要素の constraints 子要素の value 子要素で定義される複数の値を受け入れることを示します（enumeration 要素（3-14 ページ）を参照してください）。

config_template 要素定義の次の部分は、ユーザが Web インターフェイスの [Integer List] とラベル付けされたフィールドに、0 ～ 500（両端の値を含む）の整数を 1 行に 1 つずつ指定したリストを入力できることを示します。

```
<list>
<name>list_integer</name>
<display_name>Integer List</display_name>
<example>Constrained value [0-500]</example>
<item_type>
    <integer_li>
        <constraints>
            <min>0</min>
            <max>500</max>
        </constraints>
    </integer_li>
</item_type>
</list>
```

サンプル設定テンプレート

このセクションでは、サンプルの config_template 要素定義を提供します。これは、Web インターフェイスの外観と修復モジュールがユーザから受け取らなければならない情報のタイプを制御します。

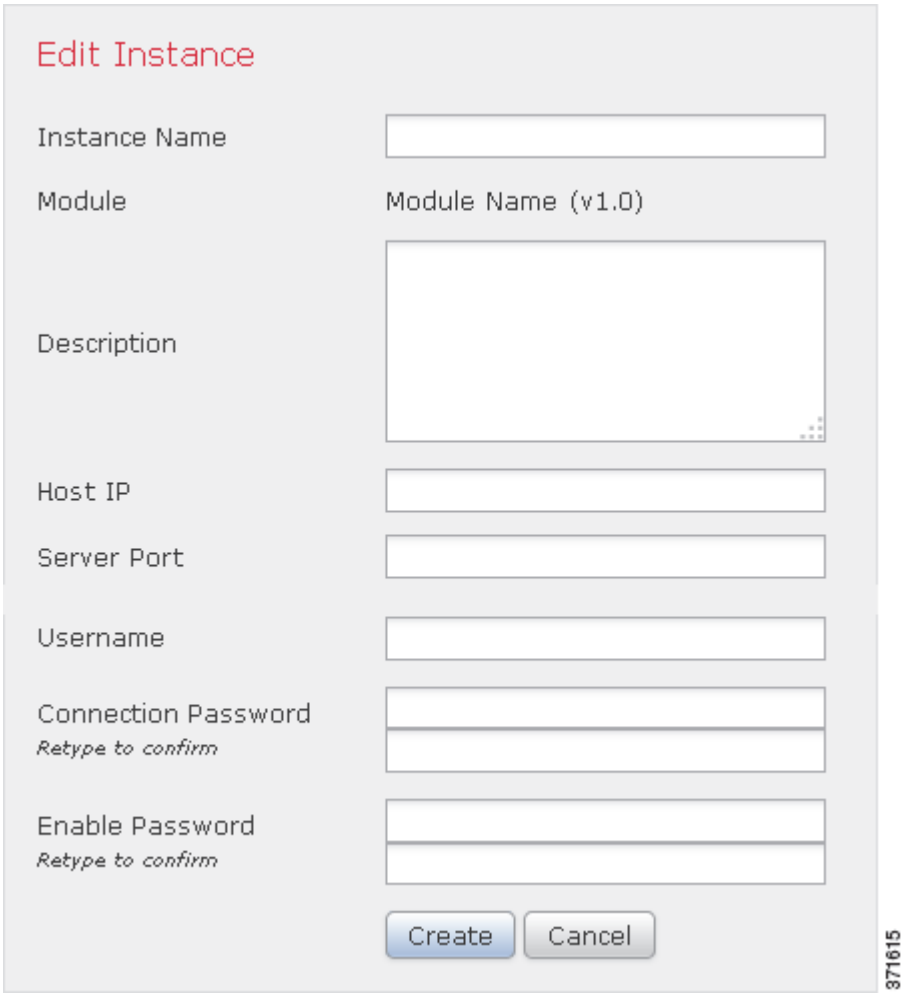
```
<config_template>
    <ipaddress>
        <name>host_ip</name>
        <display_name>Host IP</display_name>
    </ipaddress>
    <string>
        <name>user_name</name>
        <display_name>Username</display_name>
    </string>
    <password>
        <name>login_password</name>
        <display_name>Connection Password</display_name>
    </password>
    <password>
        <name>root_password</name>
        <display_name>Enable Password</display_name>
    </password>
</config_template>
```

上記のテンプレートでは、Web インターフェイスの 4 つのフィールドが表示されます。次の表で、各フィールドについて説明します。

表 3-12　サンプル設定テンプレートで作成されるフィールド

フィールド	説明
Host IP	修復モジュールが host_ip として識別する IP アドレスを受け入れます。
Username	修復モジュールが user_name として識別する文字列を受け入れます。
Connection Password	修復モジュールが login_password として識別する英数字パスワード文字列を受け入れます。
Enable Password	修復モジュールが root_password として識別する英数字パスワード文字列を受け入れます。

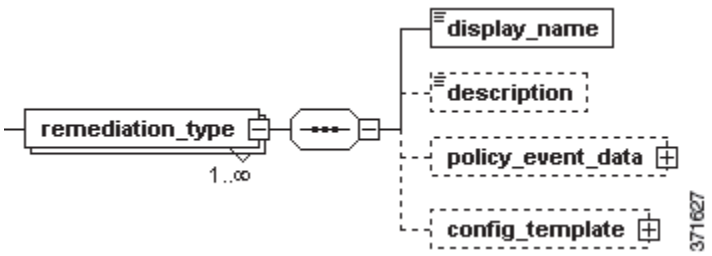
次の画面は、これらのフィールドが Web インターフェイスにどのように表示されるかを示しています。Web インターフェイスからの修復モジュールを設定するには、これらのフィールドで要求されるデータを指定する必要があります。



修復タイプの定義

修復タイプは、修復モジュールによって制御されるデバイスによって実行されるアクション（*修復*）について説明します。`module.template` で使用する各 `remediation_type` 要素は、これらの修復のいずれかを表します。修復は修復サブシステムの相関イベント データによってトリガーされます。詳細については、イベント データ（2-2 ページ）を参照してください。

次の図に、`remediation_type` 要素の子要素を示します。



次の表に、remediation_type 要素で使用可能な属性と子要素を示します。

表 3-13　remediation_type 属性と子要素

名前	タイプ	説明	必須かどうか
name	属性	修復タイプの修復モジュールへのコンテキストを指定します。 この属性は必須で、1 ～ 64（両端の値を含む）文字の文字列を受け入れます。名前には、スペースを含めることはできません。英数字およびアンダースコア（_）とハイフン（-）の文字を含めることができます。remediation_type 名は、各モジュール内で一意である必要があります。	yes
display_name	要素	Web インターフェイスの修復タイプをラベル付けします。	yes
policy_event_data	要素	修復モジュールが修復サブシステムから受け取る必要のある相関イベント データを指定します。 policy_event_data には、特定の相関イベント データ項目を表す 1 つの子要素 pe_item があります。複数の pe_item 要素を使用して、複数の相関イベント データ項目を提供します。適切な相関イベント データ値の詳細については、イベント データ（2-2 ページ）を参照してください。	no
config_template	要素	この修復モジュールのインスタンスを設定する際に、ユーザが入力しなければならない情報を指定します。詳細については、設定テンプレートの定義（3-4 ページ）を参照してください。	no

module.template ファイルの次の部分は、複数の remediation_type 要素定義を示しています。

```
<remediation_type name="block_src">
<display_name>Block Source</display_name>
<policy_event_data>
    <pe_item>src_ip_addr</pe_item>
    <pe_item>src_port</pe_item>
    <pe_item>src_protocol</pe_item>
</policy_event_data>
</remediation_type>
<remediation_type name="block_dest">
<display_name>Block Destination</display_name>
<policy_event_data>
    <pe_item>dest_ip_addr</pe_item>
    <pe_item>dest_port</pe_item>
    <pe_item>dest_protocol</pe_item>
</policy_event_data>
</remediation_type>
<remediation_type name="acl_insert">
<display_name>ACL Insertion</display_name>
<policy_event_data>
    <pe_item>src_ip_addr</pe_item>
    <pe_item>src_port</pe_item>
    <pe_item>src_protocol</pe_item>
    <pe_item>dest_ip_addr</pe_item>
    <pe_item>dest_port</pe_item>
    <pe_item>dest_protocol</pe_item>
</policy_event_data>
    <config_template>
        <integer>
```

```
            <name>acl_num</name>
            <display_name>ACL Number</display_name>
        </integer>
    </config_template>
</remediation_type>
```

上記の例では、block_src、block_dest、acl_insert の 3 つの修復タイプが指定されています。これらにはそれぞれ、特定の相関イベント（pe_item）データが必要です。acl_insert 修復タイプも、config_template 子要素で指定された設定データを必要とします。ユーザは、そのタイプのインスタンスを設定するときに、ACL 番号を指定する必要があります。

終了ステータスの定義

修復サブシステムでは、修復モジュールから整数形式で、終了ステータスまたは戻りコードを受け取ることを想定しています。

シスコでは、修復モジュールが返すことのできる一連の事前定義された終了ステータス メッセージを提供します。1 ～ 128（両端の値を含む）の整数値に対応する事前定義された終了ステータスを返すこともできます。次に、これらの事前定義された終了ステータス コードを示します。

表 3-14　事前定義された終了ステータス

終了ステータス	説明
0	修復は正常に完了しました。
1	修復モジュールに提供された入力でエラーが発生しました。
2	修復モジュールの設定でエラーが発生しました。
3	リモート デバイスまたはサーバへのログインでエラーが発生しました。
4	リモート デバイスまたはサーバで必要な権限が取得できませんでした。
5	リモート デバイスまたはサーバへのログインがタイムアウトしました。
6	リモート コマンドまたはサーバの実行がタイムアウトしました。
7	リモート デバイスまたはサーバに到達できませんでした。
8	修復が試行されましたが、失敗しました。
10	ホワイト リストへの一致が検出されました。
11	修復プログラムの実行に失敗しました。
20	原因不明または予期しないエラーが発生しました。

また、モジュールで、カスタム終了ステータスとして 129 ～ 254（両端の値を含む）の整数を返すこともできます。修復モジュールでカスタム終了ステータスを返す場合、モジュールが返す終了ステータスのセットを定義する必要があります。module.template で使用する各 exit_status 要素は、修復モジュールが返すことのできるカスタム終了ステータスを表します。詳細については、モジュールによって返されるデータ（2-11 ページ）を参照してください。

exit_status 要素は戻りコードを説明する文字列を受け入れます。また、要素は、129 ～ 255 の一意の整数を受け入れる value 属性を必要とします。この属性は、修復モジュールの戻りコードを、修復ステータス イベント ビューに表示されるその説明に関連付けます。

次の例は、有効なカスタム exit_status 要素を示しています。

```
<exit_status value="138">syslog error</exit_status>
<exit_status value="139">unknown error</exit_status>
```

修復 SDK の使用

修復 SDK について

シスコが提供する修復モジュールを展開することに加え、独自のカスタム修復をインストールして実行することによって、関連する相関ポリシーの違反に対する応答を自動化することができます。シスコでは、使用を開始する際に役立つソフトウェア開発キット（SDK）を提供しており、サポート サイトからダウンロードできます。

SDK の目的

SDK と『シスコ修復 API ガイド』のこの章の情報を使用することによって、以下を実行できます。

- 簡単な修復モジュールを実際に展開し、処理に精通することができます。インストール、設定、削除は簡単です。
- 修復プログラムのソース コードを調べ、修復サブシステムと対話したり、複数の修復機能を実行したりするために API を使用する 1 つの方法について理解することができます。

注意：SDK の syslog モジュールは、実稼働での使用を想定していません。

開発中に、参照リソースとして Firepower Management Center にロード済みのシスコが提供するモジュールを使用できます。これらのモジュールはすべて、Firepower Management Center の `/var/sf/remediation_modules` でアクセスできます。インストールされている各モジュールのこのディレクトリに `.tgz` パッケージがあります。モジュールについては、シスコが提供する修復モジュール（1-3 ページ）を参照してください。

SDK の説明

修復 SDK には、2 つのバージョン（Perl と C）の syslog アラート修復モジュールがあります。このモジュールを使用するには、リモート トラフィックを実行および受信する syslog サーバが必要です。

このモジュールでは、以下の 2 つの修復タイプが提供されます。

- `Simple_Notification` - イベントをトリガーするために、送信元 IP アドレス、送信元ポート（使用可能な場合）、および IP プロトコル（使用可能な場合）で syslog アラートを生成します。
- `Complete_Notification` - イベントをトリガーするために、Simple Notification と同じフィールドを持つ syslog アラートを生成します。これには、宛先 IP アドレス、宛先ポート、および重大度インジケータも含まれます。

すべての修復モジュールと同様、モジュールのインスタンスを追加するために、Web インターフェイスでいくつかの設定を入力します。各インスタンスは、ネットワーク上の特定のデバイス（この場合は syslog サーバ）を対象とし、そのインスタンスに対して修復を実行します。`Complete_Notification` 修復タイプを実行する場合、`Simple_Notification` 修復タイプでは必要とされない syslog ファシリティ レベルを選択します。

Perl バージョンのファイルの一覧については、次の表を参照してください。

表 4-1　サンプル Perl モジュール

含まれるファイル	説明
syslog.pl	関連付けられた相関ポリシーに違反した場合に、syslog アラートを実行するプログラム。
module.template	モジュール設定ファイル。必要なイベント データ、ユーザがインスタンスを作成する際に Web インターフェイスで収集する必要がある情報、およびその他の必須設定パラメータを定義します。
Makefile	Firepower Management Center にインストールするために、修復モジュールのファイルをパッケージ化するサンプル Makefile。

C バージョンのファイルの一覧については、次の表を参照してください。

表 4-2　サンプル C モジュール

含まれるファイル	説明
syslogc.c	関連付けられた相関ポリシーに違反した場合に、syslog アラートを実行するプログラム。
module.template	モジュール設定ファイル。必要なイベント データ、ユーザがインスタンスを作成する際に Web インターフェイスで収集する必要がある情報、およびその他の必須設定パラメータを定義します。

SDK のダウンロード

修復 SDK をダウンロードするには、次の手順を実行します。

1. サポート Web サイト（https://support.sourcefire.com/downloads）にアクセスします。
2. [Product Category] の下でソフトウェア バージョンを選択し、[Software] を選択します。修復 SDK のダウンロード リンクがページの **api** 部分にあります。
3. クライアント マシンの任意のフォルダで .zip ファイルを解凍します。

展開とインストールの手順の概要

以下の手順は、カスタム修復モジュールの作成、インストール、設定を実行するために必要なタスクのチェックリストの形式を取っています。手順の一部には、『*修復 API ガイド*』または『*Firepower System User Guide*』の相互参照セクションで説明される手順に関する詳細と説明が含まれます。

カスタム修復モジュールを開発、インストール、設定するには、以下を実行する必要があります。

1. 軽減したい状況、および環境で検出されたその状況を適切に解決するアクションを識別します。
2. 修復サブシステムから取得できるデータ要素について理解します。Firepower Management Center で提供される修復に関する使用可能なすべてのフィールドの定義については、修復サブシステムで使用可能なデータ（2-1 ページ）を参照してください。

修復サブシステムに組み込まれている戻りコード機能についても理解する必要があります。詳細については、終了ステータスの定義（3-21 ページ）を参照してください。

3. プログラムが対応する必要のあるすべての修復アクション（修復タイプ）を識別する高いレベルでの設計を生成します。
4. 目的の修復に必要なすべての機能に対応するように修復プログラムを記述します。修復モジュール プログラムは bash、tsch、Perl、または C で記述できます。修復プログラムの開発者への注意事項（4-3 ページ）で提供される技術的なガイダンスを参考にして、プログラムを作成してください。
5. 修復モジュールに関するモジュール テンプレート ファイルを作成します。モジュール テンプレートのデータ要素および構文については、修復サブシステムとの通信（3-1 ページ）の章を参照してください。

既存の `module.template` ファイルを編集して作成を開始することにより、時間を節約できます。

6. モジュールのパッケージ化（2-12 ページ）で説明されているように、修復モジュールをパッケージ化します。
7. モジュールのインストール（2-12 ページ）で説明されているように、Policy & Response コンポーネントを使用して、Firepower Management Center にモジュールをインストールします。Firepower Management Center 上のパッケージをロードし、シスコが提供するモジュールの 1 つを設定するかのように処理を続けます。
8. 修復モジュールの個々の修復タイプが、定義済みの相関ポリシーの適切な相関ルールに対する応答として割り当てられていることを確認します。手順の詳細については、『*Firepower System User Guide*』を参照してください。

修復プログラムの開発者への注意事項

修復プログラムの必要な範囲および機能を定義し、修復アクションに使用できるデータ要素について理解したら、修復プログラムを記述できます。

修復モジュール プログラムは bash、tsch、Perl、または C で記述できます。

次の表に、必要なトピックの参照先を示します。

表 4-3　　プログラマの注記

内容	探す場所
修復サブシステムのファイル構造とワークフロー環境	修復サブシステム ファイルの構造について（4-4 ページ）
1 つの修復プログラムへの複数の修復タイプの実装	修復プログラムへの修復タイプの実装（4-4 ページ）
修復サブシステム ファイルの構造	修復サブシステム ファイルの構造について（4-4 ページ）
修復プログラムと Firepower Management Center 修復サブシステムとの対話	修復プログラムのワーク フローについて（4-5 ページ）
パラメータが Firepower Management Center から修復モジュールに渡される順序	コマンド ライン パラメータの順序（4-5 ページ）
修復デーモンが未定義のデータ要素を処理する方法	未定義データ要素の処理（4-5 ページ）
修復プログラムからの戻りコード	戻りコードの処理（4-6 ページ）
修復プログラムのランタイム モード	重要なグローバル コンフィギュレーション要素（4-6 ページ）
ユーザ入力の代替エンコーディング	重要なグローバル コンフィギュレーション要素（4-6 ページ）

修復プログラムへの修復タイプの実装

Firepower Management Center の修復デーモンは、修復プログラムを起動するときに、コマンド ラインの最初の引数として修復の名前を指定します。SDK Perl プログラム `syslog.pl`, の次のコード スニペットは、プログラムが適切な修復機能に分岐できる 1 つの方法を示しています。プログラムは、修復デーモンの最初のフィールドで設定される `$remediation_config` の内容に基づいて、`SimpleNotification()` または `CompleteNotification()` のいずれかを実行します。このサンプルには、戻りコードの処理（4-6 ページ）で説明される戻りコードの使用方法も示されています。

```
# Call the appropriate function for the remediation type
my $rval = 0;
if($remediation_config->{type} eq "Simple_Notification")
{
$rval = SimpleNotification($instance_config, $remediation_config,
\@pe_event_data);
}
elsif($remediation_config->{type} eq "Complete_Notification")
{
$rval= CompleteNotification($instance_config,$remediation_config,
\@pe_event_data);
}
else
{
warn "Invalid remediation type.Check your instance.conf\n";
exit(CONFIG_ERR);
}
exit($rval);
```

`module.template` ファイルですべての修復タイプの名前を宣言し、Web インターフェイスからインスタンスを追加する際に、修復タイプと各インスタンスを関連付けます。インスタンスによって実行される修復タイプは、`instance.config` ファイルに記録されます。このファイルは、修復サブシステム ファイルの構造について（4-4 ページ）で説明される `instance.config` サブディレクトリに保存されます。

修復サブシステム ファイルの構造について

各修復モジュールのルート ディレクトリは、修復モジュールの名前とバージョン番号に由来します。これらは両方とも `module.template` ファイルで宣言されます。`module.template` の要素の詳細については、config 要素（2-8 ページ）を参照してください。

`module.template` で名前 `syslog` とバージョン `1.0` を宣言し、syslog.tgz にパッケージ化したモジュールをインストールした場合、システムはディレクトリ `/var/sf/remediation/syslog_1.0` にモジュールを配置します。このディレクトリには、`module.template` ファイルとモジュールの修復プログラム バイナリが含まれます。

修復のインスタンスを追加し、そのインスタンスに `log_tokyo` という名前を付けると、次のディレクトリが作成されます。

```
/var/sf/remediation/syslog_1.0/log_tokyo
```

また、`instance.conf` という名前のファイルをそのディレクトリに配置します。XML 形式の `instance.conf` ファイルには、`log_tokyo` インスタンスの設定情報が含まれます。

次の Linux コマンド シーケンスは、上記のディレクトリ構造を示しています。

```
# cd /var/sf/remediations
# ls
NMap_perl_2.0  SetAttrib_1.0         cisco_pix_1.0
cisco_ios_router_1.0  syslog_perl_0.1
# cd syslog_perl_0.1
# ls
```

```
log_chicago log_tokyo module.template syslog.pl
# cd log_tokyo
# ls
# instance.conf
```

instance.conf ファイルには、log_tokyo インスタンスが実行される修復タイプの名前が含まれます。上記の例では、log_tokyo インスタンスを追加したユーザは、syslog 修復モジュールに対して定義される Simple_Notification または Complete_Notification のいずれの修復タイプを実行するかを設定できます。

instance.conf XML ファイルの要素の詳細については、インスタンス設定データ（2-8 ページ）を参照してください。

修復プログラムのワーク フローについて

Firepower Management Center が修復インスタンスを実行すると、修復デーモンはインスタンスのサブディレクトリから修復プログラムを起動し、instance.conf ファイルから修復プログラムにコマンド ラインの引数としてデータを提供します。

このプロセスを具体例で説明すると次のようになります。ポリシー違反が log_tokyo という名前の syslog インスタンス（これは、送信元 IP アドレス 1.1.1.1 と宛先 IP アドレス 2.2.2.2 で Simple_Notification という名前の修復を呼び出します）を起動すると、Firepower Management Center は作業ディレクトリを /var/sf/remediations/Syslog_1.0/log_tokyo（つまり、instance.conf サブディレクトリ）に設定し、修復バイナリ syslog.pl を実行します。デーモンのコマンド ライン構文は次のようになります。

```
../syslog.pl Simple_Notification 1.1.1.1 2.2.2.2
```

syslog.pl 実行可能ファイルは instance.conf サブ ディレクトリの親ディレクトリにあることに特に注目してください。

この方法でコマンドが実行されると、instance.conf ファイルが現在のディレクトリに存在するため、syslog.pl バイナリはそのファイルの情報をロードできます。バイナリで親ディレクトリ（/var/sf/remediations/Syslog_1.0）にあるモジュールやその他のファイルをロードする必要がある場合、親ディレクトリから明示的にロードするようにコーディングする必要がります（つまり、"../" から始めてパスを指定する必要があります）。そうしないと、バイナリは必要とするファイルを検出できません。

Perl では、次のように lib() 関数を使用してこの問題に対処することもできます。

```
use lib("../");
```

プログラムで、instance.conf ファイルのオープン、読み取り、解析、クローズを実行できる必要があります。

コマンド ライン パラメータの順序

修復デーモンが修復モジュールにイベント データを渡す際、修復の名前を渡し、その後 module.template で指定されたフィールドの順序で相関イベント データを渡します。module.template では、モジュールに渡す各フィールドは <pe_item> タグを使用して宣言されます。

pe_item が module.template で optional に設定されていて、未定義である（つまり、特定の pe_item に値が設定されていない）場合、修復デーモンは「undefined」または NULL をモジュールに渡します。pe_item が module.template で required に設定されているものの、未定義である場合、修復デーモンは、使用可能な値がなく、修復モジュール バイナリは実行されないことを示すメッセージを修復ログに記録します。修復のテーブル ビューと呼ばれる Web インターフェイスに修復ログを表示できます。このビューへのアクセス方法および使用方法の詳細については、『*Firepower System User Guide*』を参照してください。

未定義データ要素の処理

修復デーモンは、項目が module.template で optional または required とマークされているかに応じて、未定義のデータ項目を異なる方法で処理します。未定義とは、Firepower Management Center データベースで項目に値が指定されていないことを意味します。デーモンの処理は次のとおりです。

- `pe_item` が `module.template` で `optional` に設定されている場合、修復デーモンは「undefined」または NULL をモジュールに渡します。
- `pe_item` が `module.template` で `required` に設定されている場合、修復デーモンは修復を実行せず、使用可能な値がないことを示すメッセージを修復ログに記録します。

戻りコードの処理

Firepower Management Center は各インスタンスの戻りコードを待機し、修復ログにコードを記録します。事前定義されたカスタム戻りコードの詳細については、終了ステータスの定義（3-21 ページ）を参照してください。

Firepower Management Center の Web インターフェイスの修復のテーブル ビューに起動された修復のそれぞれの結果が表示されます。修復のテーブル ビューへのアクセス方法および使用方法については、『*Firepower System User Guide*』を参照してください。

重要なグローバル コンフィギュレーション要素

`module.template` ファイルの対応する要素を設定して、次の表で説明する修復 API 機能を有効にすることができます。設定の詳細については、グローバル設定の定義（3-2 ページ）を参照してください。

表 4-4　module.template のグローバル コンフィギュレーションで有効にされる機能

次の機能を有効にする	次の module.template パラメータを設定する
ルートとしての修復プログラムの実行	run_as_root 警告：シスコでは、どうしても必要な場合にのみ、この要素を使用することを推奨します。
ユーザ入力の HTML エンコーディング	encode_values 注：この要素を使用する場合、修復モジュールは入力処理の一部として HTML デコードを処理しなければなりません。