APPENDIX A

# システム アーキテクチャについて

（注） 現在、Cisco IPS 7.1 をサポートしているプラットフォームは、IPS SSP を搭載した Cisco ASA 5585-X のみです。それ以外の Cisco IPS センサーは、IPS 7.1 を現在サポートしていません。

（注） IPS SSP を搭載した Cisco ASA 5585-X は、ASA 8.2(4.4) 以降および ASA 8.4(2) 以降でサポートされています。ASA 8.3(*x*) ではサポートされていません。

この付録では、Cisco IPS のシステム アーキテクチャについて説明します。次のような構成になっています。



## Cisco IPS の目的

Cisco IPS の目的は、悪意のあるネットワーク アクティビティを検出して防止することにあります。Cisco IPS ソフトウェアは、アプライアンスとモジュールの 2 つのプラットフォームにインストールできます。Cisco IPS には、管理アプリケーションとモニタリング アプリケーションが含まれています。IDM は、IPS の管理およびモニタリングに使用できるネットワーク管理 JAVA アプリケーションです。

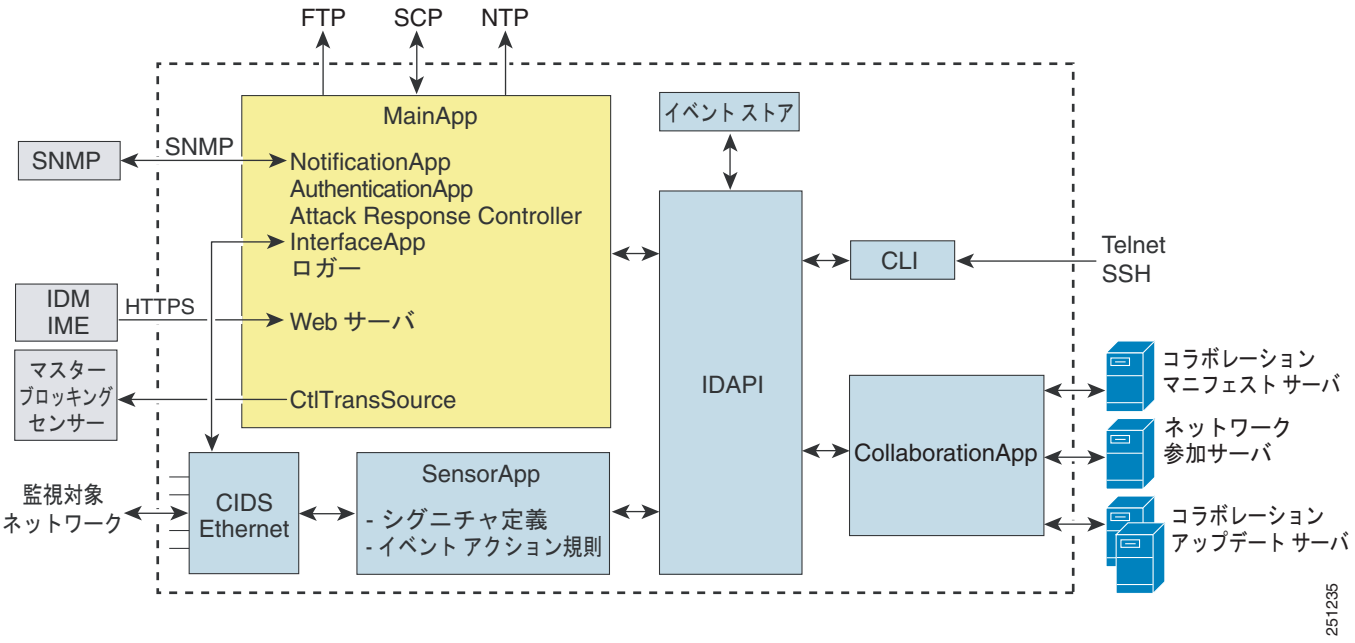
IME は、IPS イベントの表示に使用できる IPS ネットワーク モニタリング JAVA アプリケーションです。IME には、IDM 設定コンポーネントも含まれています。IDM および IME は、HTTP または HTTPS を使用して IPS と通信し、ユーザのコンピュータ上で実行されます。

# システム設計

Cisco IPS ソフトウェアは、Linux オペレーティング システム上で動作します。Linux OS を強化するために、不要なパッケージの削除、使用しないサービスの無効化、ネットワーク アクセスの制限、およびシェルへのアクセスの停止を行いました。

図 A-1 に、IPS ソフトウェアのシステム設計を示します。

**図 A-1　システム設計**




### 詳細情報

- MainApp の詳細については、「MainApp」（P.A-5）を参照してください。
- SensorApp の詳細については、「SensorApp」（P.A-22）を参照してください。
- CollaborationApp の詳細については、「CollaborationApp」（P.A-27）を参照してください。
- CLI の詳細については、「CLI」（P.A-29）を参照してください。

# システム アプリケーション

（注）　各アプリケーションには、それぞれ独自の XML 形式の構成ファイルがあります。

Cisco IPS ソフトウェアには、次のアプリケーションが含まれています。

- MainApp：システムの初期化、他のアプリケーションの起動および停止、OS の設定、およびアップグレードの実行を行います。これには、次のコンポーネントが含まれます。
  - ctlTransSource（Control Transaction Server）：センサーによる制御トランザクションの送信を可能にします。これは、Attack Response Controller（旧称 Network Access Controller）のマスター ブロッキング センサー機能をイネーブルにするために使用します。
  - イベント ストア：IPS イベント（エラー、ステータス、アラートの各システム メッセージ）を格納するために使用され、CLI、IDM、IME、ASDM、または SDEE からアクセスできるインデックス付きストア。
  - InterfaceApp：バイパスおよび物理設定を処理し、ペアになっているインターフェイスを定義します。物理設定は、速度、デュプレックス、および管理状態です。
  - ロガー：アプリケーションのすべてのログ メッセージをログ ファイルに書き出し、アプリケーションのエラー メッセージをイベント ストアに書き出します。
  - Attack Response Controller（旧称 Network Access Controller）：リモート ネットワーク デバイス（ファイアウォール、ルータ、およびスイッチ）を管理し、アラート イベントの発生時にブロッキング機能を提供します。ARC は、制御対象のネットワーク デバイスで ACL を作成して適用するか、または **shun** コマンドを使用します（ファイアウォール）。
  - NotificationApp：アラート、ステータス、およびエラー イベントによってトリガーされたときに SNMP トラップを送信します。NotificationApp はパブリック ドメイン SNMP エージェントを使用します。SNMP GET は、センサーの全般的なヘルスに関する情報を提供します。
  - Web サーバ（HTTP SDEE サーバ）：いくつかのサーブレットを使用して IPS サービスを提供することにより、SDEE プロトコルを介した Web インターフェイスおよび他の IPS デバイスとの通信を提供します。
  - AuthenticationApp：ユーザが CLI、IDM、IME、ASDM、または SDEE アクションを実行できることを確認します。
- SensorApp（分析エンジン）：パケットのキャプチャと分析を行います。
- CollaborationApp：IDAPI 制御トランザクション、セマフォ、共有メモリ、ファイル交換などのさまざまなプロセス間通信テクノロジーを使用して、MainApp および SensorApp とのインターフェイスをとります。
- CLI：Telnet または SSH を通じてセンサーに正しくログインすると実行されるインターフェイス。CLI で作成されたすべてのアカウントは、CLI をアカウントのシェルとして使用します（サービス アカウントは例外。許可されるサービス アカウントは 1 つだけです）。使用できる CLI コマンドは、ユーザの権限に依存します。

すべての Cisco IPS アプリケーションは、IDAPI と呼ばれる共通 API を通じて相互に通信します。リモート アプリケーション（他のセンサー、管理アプリケーション、およびサードパーティ ソフトウェア）は、SDEE プロトコルを通じてセンサーと通信します。

センサーには、次のパーティションがあります。

- アプリケーション パーティション：完全な IPS システム イメージ。
- メンテナンス パーティション：IDSM2 のアプリケーション パーティションのイメージを再作成するために使用される、特殊な目的の IPS イメージ。メンテナンス パーティションのイメージを再作成すると、すべての設定が失われます。
- リカバリ パーティション：センサーのリカバリに使用される、特殊な目的のイメージ。リカバリ パーティションで起動すると、アプリケーション パーティションを完全に再作成することができます。ネットワーク設定は保存されますが、それ以外のすべての設定は失われます。

# ユーザ対話

ユーザは、次の方法で Cisco IPS と対話します。

- デバイスのパラメータ設定

  システムとその機能の初期設定を生成します。これは頻繁に実行する作業ではなく、通常は一度だけ実行します。システムには妥当なデフォルト値が設定され、ユーザができるだけ変更を加えずに済むようになっています。Cisco IPS は CLI、IDM、IME、CSM、ASDM を使用して設定できるほか、別のアプリケーションで SDEE を使用して設定できます。

- 調整

  主に分析エンジンに対して、わずかな設定変更を加えます。分析エンジンはアプリケーションでネットワーク トラフィックをモニタする部分です。ネットワークに最初にシステムをインストールした後、システムが効率的に動作し、有用と考えられる情報だけを生成するようになるまで何度も調整を加えることができます。カスタム シグニチャの作成、機能のイネーブル化、サービス パックまたはシグニチャ アップデートの適用が可能です。Cisco IPS は CLI、IDM、IME、CSM、ASDM を使用して調整できるほか、別のアプリケーションで SDEE を使用して調整できます。

- アップデート

  自動アップデートをスケジュールしたり、アプリケーションやシグニチャ データ ファイルに今すぐアップデートを適用したりすることができます。Cisco IPS は CLI、IDM、IME、CSM、ASDM を使用して調整できるほか、別のアプリケーションで SDEE を使用してアップデートできます。

- 情報の取得

  CLI、IDM、IME、CSM、ASDM、CS MARS を使用して、または、別のアプリケーションで SDEE を使用して、システムからデータ（ステータス メッセージ、エラー、およびアラート）を取得できます。

**詳細情報**

SDEE の詳細については、「SDEE」（P.A-33）を参照してください。

# セキュリティ機能

Cisco IPS には次のセキュリティ機能があります。

- ネットワーク アクセスは、アクセスを許可された特定のホストに限定されます。
- Web サーバ、SSH と SCP または Telnet を使用して接続を試みるリモート ホストには、すべて認証が実行されます。

- デフォルトでは、Telnet アクセスはディセーブルになっています。Telnet をイネーブルにするように選択できます。
- デフォルトでは、SSH アクセスはイネーブルになっています。
- FTP サーバはセンサー上では実行されません。SCP を使用して、ファイルをリモートでコピーできます。
- デフォルトでは、Web サーバでは TLS または SSL が使用されます。TLS と SSL をディセーブルにするように選択できます。
- 不要なサービスはディセーブルになっています。
- CISCO-CIDS-MIB 内では、Cisco MIB Police で必要とされる SNMP セットのみが許可されます。パブリック ドメイン SNMP エージェントによって実装される OID は、MIB によって指定される場合は書き込み可能になります。

# MainApp

ここでは MainApp について説明します。次のような構成になっています。



## MainApp について

MainApp には、SensorApp と CLI を除くすべての IPS コンポーネントが含まれています。起動時にオペレーティング システムによってロードされ、SensorApp をロードします。その後、MainApp は次のサブシステム コンポーネントを起動します。

- 認証
- ロガー
- ARC
- Web サーバ
- 通知（SNMP）
- 外部製品インターフェイス
- インターフェイス マネージャ
- イベント ストア
- ヘルスおよびセキュリティ モニタリング

# MainApp の役割

MainApp には、次の役割があります。

- シスコがサポートするハードウェア プラットフォームの検証
- ソフトウェアのバージョンおよび PEP 情報の報告
- IPS コンポーネントの起動、停止、およびバージョンの報告
- ホスト システムの設定
- システム クロックの管理
- イベント ストアの管理
- ソフトウェア アップグレードのインストールおよびアンインストール

（注）　Cisco IPS では、MainApp はシグニチャおよびシグニチャ エンジンのアップデートを Cisco.com から自動的にダウンロードできます。

- オペレーティング システムのシャットダウンおよびリブート

MainApp は、**show version** コマンドへの応答として次の情報を表示します。

- センサーのビルド バージョン
- MainApp のバージョン
- 実行中の各アプリケーションのバージョン
- インストールされている各アップグレードのバージョンおよびタイムスタンプ
- インストールされている各アップグレードの次のダウングレード バージョン
- プラットフォームのバージョン
- 他のパーティションにあるセンサーのビルドのバージョン

MainApp は、ホスト統計情報の収集や、ヘルスおよびセキュリティ モニタリングのステータスの報告も行います。

# イベント ストア

ここではイベント ストアについて説明します。次のような構成になっています。



## イベント ストアについて

各 IPS イベントは、タイム スタンプと一意の単調な昇順 ID とともにイベント ストアに保存されます。このタイム スタンプは、固定サイズのインデックス付きイベント ストア内でイベントにインデックスを付ける際のプライマリ キーとして使用されます。循環式のイベント ストアが設定されたサイズに達すると、最も古い 1 つまたは複数のイベントが新しく保存されるイベントで上書きされます。イベント ストアにアラート イベントを書き込むアプリケーションは SensorApp だけです。ログ、ステータス、およびエラー イベントは、すべてのアプリケーションがイベント ストアに書き込みます。

固定サイズのインデックス付きイベント ストアでは、時刻、タイプ、プライオリティ、および限られた数のユーザ定義属性に基づいて、シンプルなイベントのクエリーを実行できます。各イベントに low、medium、または high のプライオリティが割り当てられている場合は、1 回のイベント クエリーで目的のイベント タイプ、侵入イベントのプライオリティ、および時間範囲のリストを指定できます。

表 A-1 に、いくつかの例を示します。

**表 A-1　IPS イベントの例**

| IPS イベント タイプ | intrusion イベントのプライオリティ | 開始タイムスタンプ値 | 停止タイムスタンプ値 | 意味 |
|---|---|---|---|---|
| status | — | 0 | 最大値 | 格納されているすべての status イベントを取得します。 |
| error<br>status | — | 0 | 65743 | 時刻 65743 よりも前に格納されたすべての error および status イベントを取得します。 |
| status | — | 65743 | 最大値 | 時刻 65743 以降に格納された status イベントを取得します。 |
| intrusion<br>attack response | low | 0 | 最大値 | プライオリティ low で格納されているすべての侵入および攻撃応答イベントを取得します。 |
| attack response<br>error<br>status<br>intrusion | medium<br>high | 4123000000 | 4123987256 | 時刻が 4123000000 から 4123987256 の間に格納された、プライオリティが medium または high の攻撃応答、エラー、ステータス、および侵入イベントを取得します。 |

イベント ストアのサイズは、センサーが IPS イベント コンシューマに接続されていないときに IPS イベントをバッファリングするのに十分な大きさです。バッファリングが十分であるかどうかは、ユーザの要件と、使用するノードの能力に依存します。循環バッファ内の最も古いイベントは、最新のイベントによって置き換えられます。

## イベントのデータ構造

さまざまな機能ユニットが、次の 7 種類のデータをやり取りします。

- intrusion イベント：SensorApp によって生成されます。intrusion イベントはセンサーが検出します。
- error イベント：ハードウェアまたはソフトウェアの不具合によって発生します。
- ステータス イベント：設定がアップデートされたなどの、アプリケーションのステータスの変更を報告します。
- control transaction log イベント：センサーが制御トランザクションの結果を記録します。
- 攻撃応答イベント：ブロック要求などの ARC 向けアクション。
- デバッグ イベント：アプリケーションのステータスの変更に関するきわめて詳細なレポートで、デバッグに使用されます。
- 制御トランザクション データ：制御トランザクションに関連するデータ。たとえば、アプリケーションの診断データ、セッション ログ、およびアプリケーションとやり取りされる設定データ。

これら 7 種類のデータを *IPS データ*と総称します。6 つのイベント タイプ（侵入、エラー、ステータス、制御トランザクション ログ、ネットワーク アクセス、およびデバッグ）は特徴が似ており、*IPS イベント*と総称されます。IPS イベントは、IPS を構成する数種類のアプリケーションによって生成され、他の IPS アプリケーションによってサブスクライブされます。IPS イベントには、次のような特徴があります。

- IDS イベントを生成するように設定されているアプリケーション インスタンスによって、自然発生的に生成されます。特定のイベントを生成するように他のアプリケーション インスタンスから要求されることはありません。
- 特定の宛先はありません。1 つまたは複数のアプリケーションによって格納され、その後、取得されます。

制御トランザクションは、次のタイプの要求に関係します。

- アプリケーション インスタンスの設定データをアップデートする要求
- アプリケーション インスタンスの診断データの要求
- アプリケーション インスタンスの診断データをリセットする要求
- アプリケーション インスタンスを再起動する要求
- ブロック要求などの ARC 向け要求

制御トランザクションには、次のような特徴があります。

- 常に、1 つの応答を伴う 1 つの要求によって構成されます。

  要求と応答には、任意の量のデータが関連付けられる可能性があります。すべての応答には、少なくとも肯定応答または否定応答が含まれます。
- ポイントツーポイントのトランザクションです。

  制御トランザクションは 1 つのアプリケーション インスタンス（発信側）からもう 1 つのアプリケーション インスタンス（応答側）に送信されます。

IPS データは、XML 形式で XML ドキュメントとして表されます。システムでは、ユーザ設定可能なパラメータがいくつかの XML ファイルに格納されます。

## IPS イベント

IPS アプリケーションは、ある種のできごとが発生したことを報告するために IPS イベントを生成します。イベントは、SensorApp が生成するアラートやアプリケーションが生成するエラーなどのデータです。イベントは、イベント ストアというローカル データベースに格納されます。

5 種類のイベントがあります。

- evAlert：ネットワーク アクティビティによってシグニチャがトリガーされたことを報告する alert イベント メッセージ
- evStatus：IPS アプリケーションのステータスとアクションを報告するステータス イベント メッセージ
- evError：応答アクションの試行中に発生したエラーを報告する error イベント メッセージ
- evLogTransaction：各センサー アプリケーションによって生成された制御トランザクションを報告する log transaction メッセージ
- evShunRqst：ARC がいつブロック要求を発行したかを報告するブロック要求メッセージ。

ステータス メッセージおよびエラー メッセージは、CLI、IME、および ASDM を使用して表示できます。

SensorApp および ARC は、応答アクション（TCP リセット、IP ロギングの開始と停止、ブロッキングの開始と停止、トリガー パケット）をステータス メッセージとしてログに記録します。

# NotificationApp

NotificationApp は、センサーがアラートおよびシステム エラー メッセージを SNMP トラップとして送信できるようにします。イベント ストアのイベントをサブスクライブし、それらのイベントを SNMP MIB に変換して、パブリック ドメイン SNMP エージェントを使用して宛先に送信します。NotificationApp は、set および get の送信をサポートします。SNMP GET により、センサーの基本的なヘルス情報を提供できます。

スパース モードでは、NotificationApp は evAlert イベントから次の情報を送信します。

- 発信者情報
- イベント ID
- イベントの重大度
- 時刻（UTC および現地時間）
- シグニチャ名
- シグニチャ ID
- サブシグニチャ ID
- 参加者情報
- アラーム特性

詳細モードでは、NotificationApp は evAlert イベントから次の情報を送信します。

- 発信者情報
- イベント ID
- イベントの重大度
- 時刻（UTC および現地時間）
- シグニチャ名
- シグニチャ ID
- サブシグニチャ ID
- バージョン
- サマリー
- インターフェイス グループ
- VLAN
- 参加者情報
- アクション
- アラーム特性
- シグニチャ
- IP ログ ID

NotificationApp は、ユーザが定義したフィルタに従って、どの evError イベントをトラップとして送信するかを決定します。エラーの重大度（error、fatal、および warning）に基づいてフィルタリングできます。NotificationApp は、evError イベントから次の情報を送信します。

- 発信者情報
- イベント ID
- イベントの重大度

- 時刻（UTC および現地時間）
- エラー メッセージ

NotificationApp は、センサーからの次の全般的なヘルスおよびシステム情報の GET をサポートします。

- パケット損失
- パケット拒否
- 生成されたアラーム
- FRP 内のフラグメント
- FRP 内のデータグラム
- 初期状態の TCP ストリーム
- 確立状態の TCP ストリーム
- 終了状態の TCP ストリーム
- システム内の TCP ストリーム
- 再構成用のキューに入れられた TCP パケット
- アクティブなノードの合計
- 両方の IP アドレスと両方のポートでキー付けされた TCP ノード
- 両方の IP アドレスと両方のポートでキー付けされた UDP ノード
- 両方の IP アドレスでキー付けされた IP ノード
- センサーのメモリの重要なステージ
- インターフェイスのステータス
- コマンドおよび制御パケットの統計情報
- フェールオーバーの状態
- システムの稼働時間
- CPU 使用率
- システムのメモリ使用状況
- PEP

**（注）** すべての IPS プラットフォームが PEP をサポートしているわけではありません。

NotificationApp は次の統計情報を提供します。

- エラー トラップ数
- イベント アクション トラップ数
- SNMP GET 要求の数
- SNMP SET 要求の数

# CtlTransSource

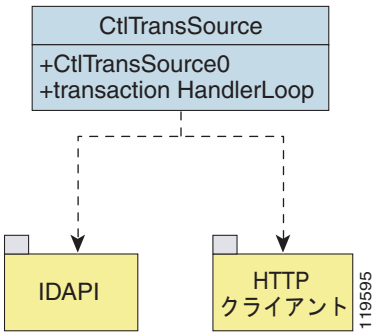
CtlTransSource は、ローカルで開始されたリモート制御トランザクションを HTTP プロトコルを使用してリモートの宛先に転送するアプリケーションです。CtlTransSource は、TLS または非 TLS 接続を開始し、その接続を使用してリモート制御トランザクションを HTTP サーバに伝えます。

CtlTransSource は、リモート HTTP サーバでリモート制御トランザクションを実行するために必要なクレデンシャルを確立する必要があります。リモート ノード上の HTTP サーバにユーザ名とパスワードの形式で ID を提示することによってクレデンシャルを確立します（基本認証）。認証が成功すると、要求者には、その接続の各要求で提示する必要のあるユーザ認証を格納したクッキーが割り当てられます。

CtlTransSource サーバ内の transactionHandlerLoop メソッドは、リモート制御トランザクションに対するプロキシとして機能します。ローカル アプリケーションがリモート制御トランザクションを起動すると、IDAPI は最初にトランザクションを CtlTransSource に送信します。transactionHandlerLoop メソッドは、CtlTransSource に送信されたリモート制御トランザクションを待機するループです。

図 A-2 に、CtlTransSource 内の transactionHandlerLoop メソッドを示します。

**図 A-2　CtlTransSource**




transactionHandlerLoop は、リモート アドレッシングされたトランザクションを受信すると、そのリモート制御トランザクションをリモートの宛先に転送するように試みます。transactionHandlerLoop は、トランザクションを制御トランザクション メッセージの形式にします。transactionHandlerLoop は、HttpClient クラスを使用して、リモート ノード上の HTTP サーバに対して制御トランザクション要求を発行します。リモート HTTP サーバは、リモート制御トランザクションを処理し、適切な応答メッセージを HTTP 応答として返します。リモート HTTP サーバが IPS Web サーバである場合、Web サーバは CtlTransSource サーブレットを使用してリモート制御トランザクションを処理します。

transactionHandlerLoop は、制御トランザクションの応答として、応答か障害応答のいずれかをリモート制御トランザクションの発信側に返します。HTTP サーバが許可されていないステータス応答（HTTP クライアントが HTTP サーバに対する十分なクレデンシャルを持っていないことを示す）を返す場合、transactionHandlerLoop は CtlTransSource 専用のユーザ名とパスワードを使用して要求者の ID を認証して、トランザクション要求を再発行します。transactionHandlerLoop は、終了を指示する制御トランザクションを受信するか終了イベントが発生するまでループします。

# Attack Response Controller

ここでは ARC について説明します。次のような構成になっています。

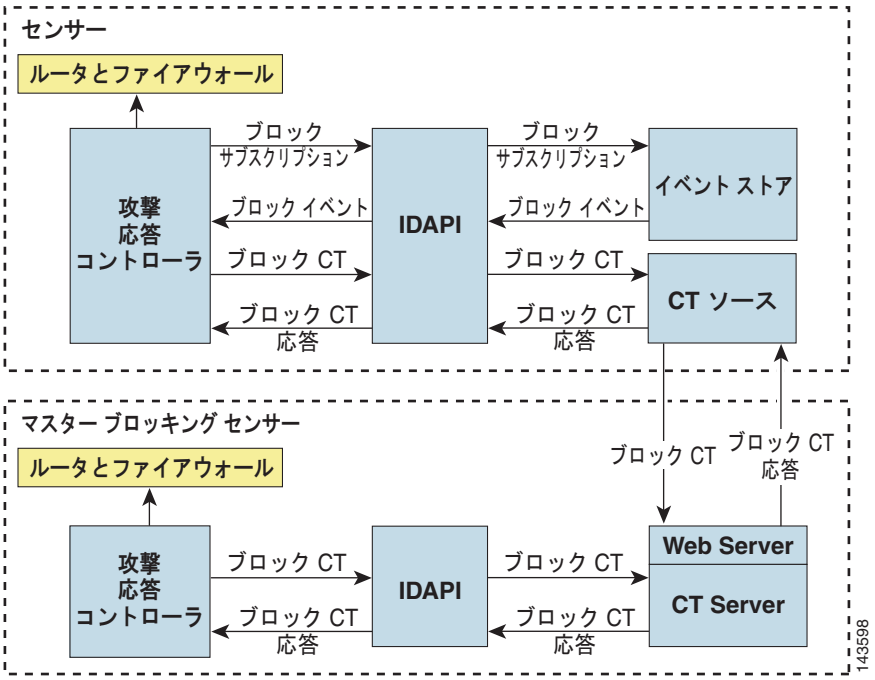


## ARC について

ARC の主な役割は、イベントをブロックすることです。NAC アプリケーションはブロックに対応するとき、管理対象のデバイスと直接対話してブロックを有効化するか、Control Transaction Server を通じてマスター ブロッキング センサーにブロック要求を送信します。マスター ブロッキング センサー上の Web サーバは、制御トランザクションを受け取るとそれを Control Transaction Server に渡し、Control Transaction Server は ARC に渡します。次に、マスター ブロッキング センサー上の ARC は、管理対象のデバイスと対話してブロックをイネーブルにします。

図 A-3 は ARC を図示したものです。

**図 A-3　　ARC**



（注） ARC のインスタンスは、ネットワーク デバイスを制御できない場合、1 つだけ制御できる場合、多数を制御できる場合があります。ARC は、他の ARC アプリケーション、IPS 管理ソフトウェア、他のネットワーク管理ソフトウェア、システム管理者との間でネットワーク デバイスの制御を一切共有しません。1 つのセンサー上で実行できる ARC インスタンスは 1 つだけです。

ARC は、次のいずれかに対応してブロックを開始します。

- ブロック アクションが設定されたシグニチャから生成された alert イベント
- CLI、IDM、IME、または ASDM から手動で設定されたブロック
- ホストまたはネットワーク アドレスに対して永続的に設定されたブロック

デバイスをブロックするように ARC を設定すると、ARC はそのデバイスと Telnet または SSH 接続を開始します。ARC は、デバイスごとに接続を維持します。ブロックが開始されると、ARC は制御対象の各デバイスに新しい設定または ACL のセットを（インターフェイス方向ごとに 1 つずつ）プッシュします。ブロックが完了すると、すべての設定または ACL はブロックを削除するようにアップデートされます。

## ARC の機能

ARC には、次の機能があります。

- 3DES を使用した Telnet および SSH 1.5 による通信

  そのデバイスの ARC 設定で指定されたプロトコルのみが試行されます。何らかの理由で接続が失われると、ARC は再確立を試みます。

- ルータ上の既存 ACL およびスイッチ上の既存 VACL。

  ARC によって制御されるルータのインターフェイスまたは方向に既存の ACL がある場合は、その ACL を ARC によって生成された設定にマージするように指定できます。これは、preblock ACL を指定するとすべてのブロックの前に、postblock ACL を指定するとブロックの後に行われます。Catalyst 6000 VACL デバイス タイプには、ARC が制御するインターフェイスごとに preblock および postblock VACL を指定できます。ファイアウォール デバイス タイプでは、ブロックを実行するために別の API が使用され、ARC はファイアウォール上の既存の ACL には影響を与えません。

  （注） Catalyst 5000 RSM および Catalyst 6000 MSFC2 ネットワーク デバイスは、Cisco ルータとして同じようにサポートされます。

- リモート センサーのリストに対するブロックの転送

  ARC は、リモート センサーのリストにブロックを転送できます。そのため、複数のセンサーが実質的に共同で 1 つのネットワーク デバイスを制御することができます。このようなリモート センサーをマスター ブロッキング センサーと言います。

- ネットワーク デバイスのインターフェイスに対するブロッキングの指定

  ルータの ARC 設定で、ブロッキングが実行されるインターフェイスと方向を指定できます。VACL 設定では、ブロッキングが実行されるインターフェイスを指定できます。

  （注） シスコのファイアウォールでは、インターフェイスまたは方向に基づくブロックは行わないため、この設定が指定されることはありません。

ARC は、同時に 250 までのインターフェイスを制御できます。

- ホストまたはネットワークに対する指定された時間のブロッキング

  ARC は、分単位で指定された時間だけ、または永続的にホストまたはネットワークをブロックできます。ARC は、ブロックの期限がいつ切れたかを判断し、その時点でホストまたはネットワークのブロックを解除します。

- 重要なイベントのロギング

  ARC は、ブロックまたはブロック解除アクションが正常に完了するか何らかのエラーが発生すると、確認イベントを書き込みます。また、ARC は、ネットワーク デバイスの通信セッションの切断と回復、設定エラー、ネットワーク デバイスから報告されるエラーなどの重要なイベントも記録します。

- ARC 再起動時におけるブロッキング状態の維持

  シャットダウンまたは再起動が発生したとき、ARC は期限が切れていないブロックを再適用します。シャットダウン中に期限が切れたブロックは削除されます。

  （注） ARC がブロッキング状態を正しく維持するためには、アプリケーションのシャットダウン中にシステムの時刻が変更されないことが条件です。

- ネットワーク デバイス再起動時におけるブロッキング状態の維持

  ネットワーク デバイスがシャットダウンされ、再起動されると、ARC は必要に応じてブロックを再適用し、期限が切れたブロックを削除します。ARC は、ARC のシャットダウンおよび再起動が同時に、または重複して発生しても影響を受けません。

- 認証と認可

  ARC は、リモート TACACS+ サーバの使用を含め、AAA 認証および認可を使用するネットワーク デバイスとの間で通信セッションを確立できます。

- 2 種類のブロッキング

  ARC は、ホスト ブロックとネットワーク ブロックをサポートしています。ホスト ブロックは、接続ベースまたは無条件です。ネットワーク ブロックは、常に無条件です。

- NAT アドレス指定

  ARC は、センサーに対して NAT アドレスを使用するネットワーク デバイスを制御できます。ネットワーク デバイスを設定する際に NAT アドレスを指定すると、そのデバイスに対するブロックからセンサーのアドレスがフィルタ処理されるときに、ローカル IP アドレスの代わりにそのアドレスが使用されます。

- シングル ポイント制御

  ARC はネットワーク デバイスの制御を管理者や他のソフトウェアとの間で共有しません。設定をアップデートする必要がある場合は、変更が完了するまで ARC をシャットダウンしておきます。ARC は、CLI または任意の Cisco IPS マネージャからイネーブルまたはディセーブルにすることができます。ARC は、再イネーブルされると、自身を完全に初期化し直します。これには、制御対象のネットワーク デバイスごとに現在の設定を再読み込みすることも含まれます。

  （注） ファイアウォールを含むすべてのネットワーク デバイスを設定する際は、ARC によるブロッキングをディセーブルにすることを推奨します。

- 常に最大 250 のアクティブなブロックを維持

  ARC は、同時に 250 までのアクティブなブロックを維持できます。ARC は 65535 までのブロックをサポートしていますが、許可は 250 までにすることを推奨します。

**（注）** ブロックの数は、インターフェイスおよび方向の数とは異なります。

## サポートされているブロッキング デバイス

ARC は、次のデバイスを制御できます。

- Cisco IOS 11.2 以降を実行する Cisco ルータ

**（注）** レート制限を実行するためには、ルータで Cisco IOS 12.3 以降を実行している必要があります。

- スーパーバイザ エンジン上で実行される Supervisor Engine ソフトウェア 5.3(1) 以降、および RSM 上で実行される IOS 11.2(9)P 以降を使用する Catalyst 5000 シリーズ スイッチ

**（注）** ブロッキングは RSM 上で実行されるため、RSM が必要です。

- PFC がインストールされ、Catalyst ソフトウェア 5.3 以降が実行される Catalyst 6000 シリーズ スイッチ
- Catalyst ソフトウェア 5.4(3) 以降、および MSFC2 上の Cisco IOS 12.1(2)E 以降を使用する Catalyst 6000 MSFC2
- Cisco ASA 500 シリーズのモデル ASA 5510、ASA 5520、および ASA 5540
- FWSM

**（注）** FWSM はマルチモードの管理コンテキストではブロックを実行できません。

## ACL と VACL

ARC が制御するインターフェイスまたは方向のパケットをフィルタ処理する場合は、すべてのブロックの前に ACL を適用したり（preblock ACL）、すべてのブロックの後に ACL を適用する（postblock ACL）ように ARC を設定できます。これらの ACL は、ネットワーク デバイス上で非アクティブな ACL として設定されます。preblock および postblock ACL は、インターフェイスおよび方向ごとに定義できます。ARC は、ネットワーク デバイス上のアクティブな ACL をアップデートする際、リストを取得してキャッシュしてから、ブロッキングを行う ACE にマージします。ほとんどの場合は、ブロックの効果を妨げないように、既存 ACL を postblock ACL として指定します。ACL はパケットを、最初に検索された ACE と照合することにより機能します。この最初の ACE でパケットが許可された場合、その後の deny ステートメントは検索されません。

インターフェイスおよび方向ごとに異なる preblock および postblock ACL を指定することも、同じ ACL を複数のインターフェイスおよび方向に再利用することもできます。preblock リストを適用しない場合は、ホストやネットワークに対して never block オプションを使用したり、既存の設定ステートメントを使用して常にブロックしたりすることができます。forever block は、normal block でタイムアウト値を -1 にした場合と同じです。

ARC は、所有する ACL のみを変更します。ユーザによって定義された ACL は変更しません。ARC が維持する ACL は、ユーザ定義の ACL では使用が禁じられている特殊な形式になっています。命名規則は、**IPS_**<interface_name>**_[in | out]_[0 | 1]** です。<interface_name> は、ブロッキング インターフェイスに対して ARC 設定で指定された名前に対応します。

Catalyst スイッチでは、これは、ブロッキング インターフェイスの VLAN 番号です。これらの名前は、preblock および postblock ACL では使用しないでください。

Catalyst 6000 VACL では、preblock および postblock VACL を指定できます。また、インターフェイスのみが指定可能です（VLAN では方向は使用されません）。

ファイアウォールでは、ブロッキングに異なる API が使用されるため、preblock または postblock ACL は使用できません。代わりに、ファイアウォールでは ACL を直接作成する必要があります。

## 再起動時の状態の維持

センサーがシャットダウンされると、ARC はすべてのブロックとレート制限を（開始タイムスタンプとともに）ARC によって維持されるローカル ファイル（nac.shun.txt）に書き込みます。ARC が起動すると、このファイルを使用して、制御対象のネットワーク デバイスにブロックのアップデートが必要かどうかが判断されます。ファイル内に期限が切れていないブロックが見つかると、ネットワーク デバイスの起動時に適用されます。ARC がシャットダウンされるときは、未処理の有効なブロックが存在していても ACL に対して特別なアクションは行われません。nac.shun.txt ファイルが正確であるためには、ARC が実行されていない間にシステムの時刻が変更されないことが必要です。

**注意**　nac.shun.txt ファイルには手動による変更を加えないでください。

次のシナリオに、ARC が再起動時にどのように状態を維持するかを示します。

### シナリオ 1

ARC が停止したときには 2 つのブロックが有効で、そのうちの 1 つは ARC が再起動する前に期限が切れます。再起動した ARC は、最初に nac.shun.txt ファイルを読み取ります。次に、preblock および postblock ACL または VACL を読み取ります。アクティブな ACL または VACL は、次の順序で構築されます。

1. **allow** *sensor_ ip_address* コマンド（**allow sensor shun** コマンドが設定されていない場合）
2. preblock ACL
3. 設定にある **always block** コマンドのエントリ
4. nac.shun.txt にある期限が切れていないブロック
5. postblock ACL

ARC 設定でホストが never block と指定されている場合、ACL の permit ステートメントには変換されません。代わりに、ARC にキャッシュされて、着信 addShunEvent イベントおよび addShunEntry 制御トランザクションをフィルタ処理するために使用されます。

### シナリオ 2

preblock ACL および postblock ACL は指定されていませんが、アクティブな ACL が存在しています。新しい ACL は、次の順序で構築されます。

1. **allow** *sensor_ ip_address* コマンド（**allow sensor shun** コマンドが設定されていない場合）
2. 設定にある **always block** コマンドのエントリ
3. nac.shun.txt にある期限が切れていないブロック
4. **permit IP any any** コマンド

## 接続ベースおよび無条件のブロッキング

ARC は、ホストに関しては 2 種類、ネットワークに関しては 1 種類のブロッキングをサポートしています。ホスト ブロックは、接続ベースまたは無条件です。ネットワーク ブロックは、常に無条件です。

ホスト ブロックを受信すると、ARC はホスト ブロックの connectionShun 属性を調べます。connectionShun が true に設定されていると、ARC は接続ブロッキングを実行します。すべてのホスト ブロックは、宛先 IP アドレス、ソース ポート、宛先ポート、プロトコルといったオプションのパラメータを含むことができます。接続ブロックが実行されるためには、少なくとも送信元および宛先 IP アドレスが存在している必要があります。接続ブロックに送信元ポートが存在する場合、その送信元ポートは無視され、ブロックには含まれません。

次の条件のとき、ARC は必要に応じて接続タイプからブロックを変換して、ブロックを無条件にします。

- 指定されたソース IP アドレスに対して、いずれかのタイプのブロックがアクティブである。
- そのソース IP アドレスに対して、いずれかのタイプの新しいブロックが受信された。
- 新しいブロックのいずれかのオプション パラメータ（ソース ポートを除く）が以前のブロックと異なる。

ブロックがアップデートされると（既存のブロックがすでに有効になっている送信元 IP アドレスまたはネットワークに関して新しいブロックが受信された場合など）、既存のブロックの残り時間（分）が決定されます。新しいブロックの時間がこの残り時間以下の場合、アクションは何も発生しません。そうでない場合は、新しいブロックのタイムアウトによって既存のブロックのタイムアウトが置き換えられます。

**注意**　シスコのファイアウォールは、ホストの接続ブロッキングをサポートしません。接続ブロックが適用されると、ファイアウォールはそれを無条件ブロックのように扱います。シスコのファイアウォールは、ネットワーク ブロッキングもサポートしません。ARC がシスコのファイアウォールに対してネットワーク ブロックの適用を試みることはありません。

## シスコのファイアウォールによるブロッキング

ARC は、**shun** コマンドを使用して、ファイアウォールでのブロックを実行します。**shun** コマンドの形式は次のとおりです。

- IP アドレスをブロックする。

  **shun srcip** [*destination_ip_address source_port destination_port* [*port*]]

- IP アドレスのブロックを解除する。

  **no shun ip**

- すべてのブロックをクリアする。

  **clear shun**

- アクティブなブロック、または実際にブロックされているグローバル アドレスを表示する。

  **show shun** [*ip_address*]

ARC は、**show shun** コマンドに対する応答を使用して、ブロックが実行されたかどうかを判断します。

**shun** コマンドは既存の ACL、コンジット、outbound コマンドを置き換えるものではないので、既存のファイアウォール設定をキャッシュしたり、ブロックをファイアウォール設定にマージしたりする必要はありません。

**注意** ARC の実行中は、手動でブロックを実行したり既存のファイアウォール設定を変更したりしないでください。

**block** コマンドでソース IP アドレスのみを指定すると、既存のアクティブな TCP 接続は維持されますが、ブロックされたホストからの着信パケットはすべてドロップされます。

ARC が最初に起動したとき、ファイアウォール内のアクティブなブロックが内部のブロッキング リストと比較されます。内部のリストに対応するエントリがないブロックは削除されます。

ARC は、ローカル ユーザ名または TACACS+ サーバを使用して、ファイアウォールでの認証をサポートします。ファイアウォールで認証に AAA を使用して TACACS+ サーバは使用しないように設定すると、ARC は予約済みのユーザ名 *pix* を使用してファイアウォールと通信します。

ファイアウォールで認証に TACACS+ サーバを使用する場合は、TACACS+ ユーザ名を使用します。AAA ログインを使用する一部のファイアウォール設定では、3 つのパスワード プロンプトが表示されます。初期ファイアウォール パスワード、AAA パスワード、およびイネーブル パスワードです。ARC では、初期ファイアウォール パスワードと AAA パスワードを同じにすることが要求されます。

NAT または PAT を使用するようにファイアウォールを設定し、ファイアウォール外部のネットワークのパケットをセンサーが確認する場合、ファイアウォール内部のネットワークから開始されたホスト攻撃が検出されると、センサーはファイアウォールから提供された変換アドレスをブロックしようとします。ダイナミック NAT アドレッシングを使用している場合は、ブロックが効果を発揮しなかったり、無害なホストがブロックされることがあります。PAT アドレッシングを使用している場合は、ファイアウォールが内部ネットワーク全体をブロックする可能性があります。これらの状況を回避するには、センサーを内部インターフェイスに配置するか、センサーがブロックを行わないように設定します。

## Catalyst スイッチによるブロッキング

PFC を搭載した Catalyst スイッチは、VACL を使用してパケットをフィルタ処理します。VACL は、VLAN 間および VLAN 内のすべてのパケットをフィルタ処理します。

WAN カードが取り付けられている場合は MSFC ルータ ACL がサポートされ、MSFC2 を通じてセンサーがインターフェイスを制御するようにすることができます。

**（注）** MSFC2 カードは、Catalyst スイッチで VACL によるブロッキングを行うための設定の一部として必要なわけではありません。

**注意** Catalyst スイッチで ARC を設定する場合は、制御対象のインターフェイスで方向を指定しないでください。インターフェイス名は VLAN 番号です。preblock および postblock のリストは、VACL である必要があります。

Catalyst の VACL には、次のコマンドを使用できます。

- 既存の VACL を表示する。

  **show security acl info** *acl_name*

- アドレスをブロックする（*address_spec* は、ルータの ACL で使用されるものと同じです）。

  **set security acl ip** *acl_name* **deny** *address_spec*

- リストの構築後に VACL をアクティブにする。

  **commit security acl all**

- 1 つの VACL をクリアする。

  **clear security acl map** *acl_name*

- すべての VACL をクリアする。

  **clear security acl map all**

- VACL を VLAN にマップする。

  **set sec acl** *acl_name vlans*

## ロガー

センサーは、すべてのイベント（alert、error、status、debug の各メッセージ）を永続的な循環バッファに記録します。また、センサーは IP ログも生成します。このメッセージと IP ログには、CLI、IDM、ASDM、および SDEE クライアントからアクセスできます。

IPS アプリケーションは、ロガーを使用してメッセージをログに記録します。ロガーは、ログ メッセージを debug、timing、warning、error、fatal の 5 段階の重大度のいずれかで送信します。ロガーは、ログ メッセージを、循環式のテキスト ファイルである /usr/cids/idsRoot/log/main.log に書き込みます。ファイルがサイズの上限に達すると、古いメッセージは新しいメッセージによって上書きされます。したがって、main.log では、最後に書き込まれたメッセージが末尾にあるとは限りません。main.log に書き込まれた最新の行を見つけるには、「= END OF FILE =」を検索してください。

main.log は、**show tech-support** コマンドの出力に含まれます。メッセージが warning またはそれよりも上（error または fatal）のレベルで記録されると、ロガーはメッセージを evError イベントに変換して（対応するエラーの重大度で）、イベント ストアに挿入します。

ロガーは、informational 以上のレベルで cron メッセージ以外のすべての syslog メッセージ（*.info;cron.none）を受信し、エラーの重大度を Warning に設定してから evErrors としてイベント ストアに挿入します。ロガーおよびアプリケーションのロギングは、service logger コマンドによって制御されます。

ロガーは、ロギング ゾーンごとにロギング重大度を制御することにより、各アプリケーションが生成するログ メッセージを制御できます。ユーザは、TAC のエンジニアまたは開発者の依頼および指示の下で、ロガー サービスの individual-zone-control にのみアクセスします。トラブルシューティングのために、TAC はデバッグ ロギングを依頼することがあります。

# AuthenticationApp

ここでは AuthenticationApp について説明します。次のような構成になっています。

- 「AuthenticationApp について」（P.A-19）
- 「ユーザの認証」（P.A-20）
- 「センサーにおける認証の設定」（P.A-20）
- 「TLS および SSH 信頼関係の管理」（P.A-20）

## AuthenticationApp について

AuthenticationApp には、次の役割があります。

- ユーザの ID の認証
- ユーザのアカウント、権限、キー、および証明書の管理
- AuthenticationApp、およびセンサー上の他のアクセス サービスで使用する認証方法を設定する。

## ユーザの認証

ユーザ アクセスに適切なセキュリティを確立するために、センサーで認証を設定する必要があります。センサーをインストールすると、初期アカウントとして、パスワードの期限が切れている cisco というアカウントが作成されます。センサーに対する管理アクセス権を持ったユーザは、デフォルトの管理アカウント（cisco）を使用してセンサーにログインすることにより、CLI または IDM や ASDM などの IPS マネージャを通じてセンサーにアクセスします。CLI で、管理者はパスワードの変更を要求されます。IPS マネージャは、setEnableAuthenticationTokenStatus 制御トランザクションを開始することによってアカウントのパスワードを変更します。

CLI または IPS マネージャを通じて、管理者は使用する認証方法（ユーザ名とパスワード、SSH 認証キーなど）を設定します。管理者用のアプリケーションは、認証設定を確立するために setAuthenticationConfig 制御トランザクションを起動します。

認証設定には、アカウント ロッキングの処理方法を指定するログイン試行の上限値が含まれています。アカウント ロッキングは、ログインの試みが連続して失敗した回数が、指定されたログイン試行の上限値を超えると起動されます。アカウントがロックされると、その後のログインの試行はすべて拒否されます。アカウントのロックを解除するには、setEnableAuthenticationTokenStatus 制御トランザクションを使用してアカウントの認証トークンをリセットします。アカウント ロッキング機能は、ログイン試行の上限値を 0 に設定すると無効になります。

管理者は、CLI または IPS マネージャから新しいユーザ アカウントを追加できます。

## センサーにおける認証の設定

ユーザが Web サーバや CLI などのサービスを通じてセンサーにアクセスしようとするときは、ユーザの ID が認証され、ユーザの権限が確立される必要があります。ユーザにアクセスを提供するサービスは、ユーザの ID を認証するために、AuthenticationApp に対して execAuthenticateUser 制御トランザクション要求を開始します。通常、制御トランザクション要求にはユーザ名とパスワードが含まれています。または、SSH 認証キーを使用してユーザの ID を認証することもできます。

AuthenticationApp は、execAuthenticateUser 制御トランザクション要求に対して、ユーザの ID の認証を試みることによって応答します。AuthenticationApp は、ユーザの認証ステータスおよび権限を含む制御トランザクション応答を返します。ユーザの ID を認証できない場合、AuthenticationApp は、非認証ステータスと匿名ユーザ権限を制御トランザクション応答として返します。制御トランザクション応答では、アカウントのパスワードが期限切れであるかどうかも示されます。execAuthenticateUser 制御トランザクションを開始することによってユーザを認証するユーザ インターフェイス アプリケーションは、ユーザにパスワードの変更を要求します。

AuthenticationApp は、基盤となるオペレーティング システムを使用してユーザの ID を確認します。すべての IPS アプリケーションは、AuthenticationApp に制御トランザクションを送信します。AuthenticationApp は、オペレーティング システムを使用してその応答を作成します。

リモート シェル サービスである Telnet と SSH は、IPS アプリケーションではありません。これらは、オペレーティング システムを直接呼び出します。ユーザが認証されていれば、オペレーティング システムは IPS CLI を起動します。この場合、CLI は特殊な形式の execAuthenticateUser 制御トランザクションを送信することにより、ログイン ユーザの権限レベルを判断します。次に CLI は、この権限レベルに応じて、使用可能にするコマンドを用意します。

## TLS および SSH 信頼関係の管理

IP ネットワーク上の暗号化通信は、パケット内のデータを復号化するために必要な秘密キーを、交換されるパケットだけから受動的攻撃者が発見できないようにすることで、データ プライバシーを実現します。

しかし、同じような危険性を持つ攻撃ベクトルとして、接続のサーバ側であるように装う詐称があります。すべての暗号化プロトコルには、クライアントがこの種の攻撃から身を守るための手段が用意されています。IPS は、SSH と TLS という 2 つの暗号化プロトコルをサポートしています。また、AuthenticationApp は、センサーが暗号化通信のクライアントまたはサーバになる場合の信頼を管理するのに役立ちます。

IPS Web サーバおよび SSH サーバは、暗号化通信のサーバ エンドポイントです。これらは、秘密キーによって ID を保護し、接続してくるクライアントに公開キーを提供します。TLS では、この公開キーは X.509 証明書の中に含まれています。X.509 証明書には他の情報も格納されています。センサーに接続するリモート システムは、接続確立時に受け取った公開キーが、目的のものであることを確認する必要があります。

クライアントは、中間者攻撃を防御するため、信頼できる公開キーのリストを維持する必要があります。この信頼性を確立するための詳細な手順は、プロトコルおよびクライアント ソフトウェアによって異なります。一般的に、クライアントは 16 ～ 20 バイトのフィンガープリントを表示します。クライアントが信頼を確立するように設定する人間のオペレータは、信頼の確立を試行する前に、アウトオブバンド方式を使用してサーバのキー フィンガープリントを取得する必要があります。フィンガープリントが一致すると信頼関係が確立され、その後、クライアントは自動的にそのサーバに接続でき、リモート サーバが詐称者でないことを確信できます。

**show ssh server-key** および **show tls fingerprint** を使用して、センサーのキー フィンガープリントを表示できます。センサー コンソールに直接接続したときにこれらのコマンドの出力を記録しておくと、後から信頼関係を確立する際、その情報を使用することにより、ネットワークを通じてセンサーの ID を確認できます。

たとえば、Microsoft Internet Explorer Web ブラウザを使用して最初にセンサーに接続したときには、セキュリティ警告ダイアログボックスで、証明書が信頼されていないことが示されます。Internet Explorer のユーザ インターフェイスを使用して証明書のサムプリントを調べることができます。この値は、**show tls fingerprint** コマンドによって表示される SHA1 フィンガープリントと正確に一致する必要があります。確認が終わったら、この証明書をブラウザの信頼済み CA のリストに追加して、永続的な信頼を確立します。

この信頼性を確立するための手順は、TLS クライアントごとに異なります。センサー自体に TLS クライアントが含まれており、制御トランザクションを他のセンサーに送信したり、アップグレードおよびコンフィギュレーション ファイルを TLS Web サーバからダウンロードするために使用されます。センサーの通信相手となる TLS サーバの信頼性を確立するには、**tls trusted-host** コマンドを使用します。

同様に、センサーには SSH クライアントが含まれており、管理対象ネットワーク デバイスとの通信、アップグレードのダウンロード、リモート ホストへのコンフィギュレーション ファイルおよびサポート ファイルのコピーに使用されます。センサーが接続する SSH サーバとの信頼関係を確立するには、**ssh host-key** コマンドを使用します。

TLS 信頼済み証明書および SSH 既知ホストのリストは、**service trusted-certificates** コマンドおよび **service ssh-known-hosts** コマンドで管理できます。

X.509 証明書には、信頼関係のセキュリティを向上させる追加情報が含まれていますが、これは混乱を招く場合があります。たとえば、X.509 証明書には、その証明書を信頼できる有効期間が含まれています。通常、この期間は証明書が作成された時点から始まる年数です。使用時点で X.509 証明書が有効であることを厳密に確認するには、クライアント システムで正確なクロックを維持する必要があります。

また、X.509 証明書は、特定のネットワーク アドレスと結び付けられています。センサーはこのフィールドに、センサーのコマンドおよびコントロール インターフェイスの IP アドレスを挿入します。そのため、センサーのコマンドおよびコントロール IP アドレスを変更すると、サーバの X.509 証明書は再生成されます。新しい IP アドレスでこのセンサーを見つけ、新しい証明書を信頼するには、以前の証明書を信頼していた、ネットワーク上のすべてのクライアントを再設定しなければなりません。

AuthenticationApp の SSH 既知ホストおよび TLS 信頼済み証明書サービスを使用することにより、センサーを高いセキュリティ レベルで運用することができます。

## Web サーバ

Web サーバは SDEE サポートを提供します。これによってセンサーは、セキュリティ イベントの報告、IDIOM トランザクションの受信、および IP ログの提供が可能になります。

Web サーバは HTTP 1.0 および 1.1 をサポートします。Web サーバとの通信には、パスワードなどの機密情報が関係することがよくあります。これらを攻撃者が盗聴することが可能になると、システムの安全性が大きく損なわれます。そのため、センサーは TLS がイネーブルな状態で出荷されます。TLS プロトコルは、SSL と互換性のある暗号化プロトコルです。

（注）　RDEP イベント サーバ サービスは、IPS 6.1 では非推奨であり、現在の IPS システム アーキテクチャから削除されました。Web サーバでは、現在は SDEE イベント サーバが使用されます。

# SensorApp

ここでは SensorApp について説明します。次のような構成になっています。



## SensorApp について

SensorApp は、パケットの取り込みと分析を実行します。SensorApp では、シグニチャを使用してポリシー違反が検出され、違反に関する情報がアラートの形式でイベント ストアに転送されます。

パケットは、センサー上のネットワーク インターフェイスからパケットを収集するように設計されたプロデューサが提供する、プロセッサのパイプラインを通過します。

SensorApp では、次のプロセッサがサポートされています。

- タイム プロセッサ

  このプロセッサは、タイムスライス カレンダーに格納されたイベントを処理します。主なタスクは、失効したデータベース エントリを期限切れにし、時間に依存する統計を計算することです。

- 拒否フィルタ プロセッサ

  このプロセッサは、攻撃者拒否機能を処理します。拒否された送信元 IP アドレスのリストを管理します。リスト内の各エントリは、グローバルな拒否タイマーに基づいて有効期限切れになります。拒否タイマーは、仮想センサー設定内で設定できます。

- シグニチャ イベント アクション プロセッサ

  このプロセッサは、イベント アクションを処理します。次のイベント アクションがサポートされます。

  - TCP フローのリセット
  - IP ログ
  - パケットの拒否

- フローの拒否
- 攻撃者の拒否
- アラート
- ホストのブロック
- 接続のブロック
- SNMP トラップの生成
- トリガー パケットのキャプチャ

イベント アクションは、イベント リスク レーティングしきい値と関連付けできます。アクションが実行されるには、このしきい値を超える必要があります。

- 統計情報プロセッサ

  このプロセッサは、パケット カウントやパケットの到着レートなど、システム統計情報を継続して追跡します。

- レイヤ 2 プロセッサ

  このプロセッサは、レイヤ 2 関連イベントを処理します。また、不正な形式のパケットを識別し、処理パスから削除します。アラート、パケットのキャプチャ、パケットの拒否など、不正な形式のパケットの検出に対して実行可能なイベントを設定することができます。レイヤ 2 プロセッサは、ユーザが設定したポリシーのために拒否されたパケットに関する統計情報をアップデートします。

- データベース プロセッサ

  このプロセッサは、シグニチャの状態とフロー データベースを管理します。

- フラグメント再構成プロセッサ

  このプロセッサは、フラグメント化された IP データグラムを再構成します。センサーがインライン モードの場合、IP フラグメントの正規化も処理します。

- ストリーム再構成プロセッサ

  このプロセッサは、さまざまなストリームベース インスペクタでパケットが適切な順序で到着するよう、TCP ストリームを並べ替えます。また、TCP ストリームの正規化も行います。ノーマライザ エンジンを使用すると、アラートおよび拒否アクションをイネーブルまたはディセーブルにできます。

  TCP ストリーム再構成プロセッサのノーマライザはホールドダウン タイマーを備え、これによって再構成イベントの後でストリームの状態を再構築することができます。このタイマーは設定できません。ホールドダウン間隔の間、システムは、ストリームの状態を、システムを通過するストリーム内の最初のパケットと同期化します。ホールドダウンの有効期限が切れると、SensorApp は設定されたポリシーを実施します。このポリシーで、スリーウェイ ハンドシェイクを使用して開かれていないストリームの拒否が要求される場合、ホールドダウン期間中に休止していた確立済みストリームは転送されず、タイムアウトが許可されます。ホールドダウン期間中に同期化されたストリームは、続行を許可されます。

- シグニチャ分析プロセッサ

  このプロセッサは、ストリームベースでなく、処理中のパケット用に設定されているインスペクタにパケットをディスパッチします。

- スレーブ ディスパッチ プロセッサ

  デュアル CPU システムにのみ存在するプロセスです。

一部のプロセッサは、シグニチャ分析を実行するためにインスペクタを呼び出します。すべてのインスペクタは、アラーム チャネルを呼び出して、必要に応じてアラートを生成することができます。

SensorApp は次のユニットもサポートします。

- 分析エンジン

  分析エンジンは、センサーの設定を処理します。インターフェイスをマッピングし、またシグニチャおよびアラーム チャネル ポリシーを設定済みインターフェイスにマッピングします。

- アラーム チャネル

  アラーム チャネルは、インスペクタによって生成されたすべてのシグニチャ イベントを処理します。主な機能は、渡された各イベントのアラートを生成することです。

# インライン、正規化、およびイベント リスク レーティング機能

（注）　Cisco ASA 5585-X（IPS SSP 搭載）は正規化をサポートしません。

SensorApp には、次のインライン、正規化、およびイベント リスク レーティング機能が含まれています。

- パケットのインライン処理

  センサーがデータ パスのパケットを処理するときには、ポリシー設定で明示的に拒否されていない限り、すべてのパケットは変更されずに転送されます。TCP の正規化により、一部のパケットが適切なカバレッジを確保するために遅延することがあります。ポリシー違反が発生すると、SensorApp はアクションの設定を許可します。パケットの拒否、フローの拒否、攻撃者の拒否など、追加のアクションがインライン モードで使用できるようになります。

  IPS にとって不明であるか、対象でないすべてのパケットは、分析されずにペアのインターフェイスに転送されます。すべてのブリッジングおよびルーティング プロトコルは、ポリシー違反が原因で発生する可能性のある拒否の場合のみ、転送されます。インライン（または無差別）のデータ処理に使用されるインターフェイスには、IP スタックは関連付けられません。無差別モードにおける 802.1q パケットの現在のサポートが、インライン モードにまで拡張されます。

- IP の正規化

  意図的または意図しない IP データグラムのフラグメンテーションによって、不正利用が隠され、それらの検出が困難または不可能になることがあります。フラグメンテーションは、ファイア ウォールやルータにあるようなアクセス コントロール ポリシーを回避するために使用される場合もあります。また、さまざまなオペレーティング システムがさまざまな方法を使用して、フラグメント化されたデータグラムをキューに入れたりディスパッチしたりします。エンド ホストがデータグラムを再構成する際に使用可能なすべての方法をセンサーで確認する必要がある場合、センサーはサービス拒否攻撃に対して脆弱になります。フラグメント化したすべてのデータグラムをインラインで再構成して、完成したデータグラムのみを転送し、必要に応じてデータグラムを再度フラグメント化することが、この問題の解決策です。IP フラグメンテーションの正規化の装置は、この機能を実行します。

- TCP の正規化

  意図的または意図しない TCP セッション セグメンテーションによって、いくつかの攻撃クラスが隠されることがあります。ポリシーが偽陽性や偽陰性なしに実施されるようにするには、2 つの TCP エンドポイントの状態が追跡され、実際のホスト エンドポイントによって処理されたデータだけが渡される必要があります。TCP ストリームで重複が発生する可能性がありますが、TCP セグメントの再転送以外は、非常にまれです。TCP セッションでの上書きは発生してはならないものです。上書きが発生する場合は、誰かがセキュリティ ポリシーを意図的に回避しようとしているか、TCP スタックの実装が壊れています。センサーが TCP プロキシとして動作していない限り、両方のエンドポイントの状態について完全な情報を保持することはできません。センサーが TCP プロキシとして動作する代わりに、セグメントが適切に並べられ、ノーマライザによって回避や攻撃に関係する異常なパケットが検索されます。

- イベント リスク レーティング

  イベント リスク レーティングには、潜在的に悪意のあるアクションの検出だけでなく、次の追加情報が組み込まれています。

  - 攻撃が成功した場合の攻撃の重大度
  - シグニチャの忠実度
  - ターゲット ホストに関する潜在的な攻撃の関連性
  - ターゲット ホストの全体的な価値

  イベント リスク レーティングはシステムからの偽陽性の減少に役立ち、アラート発生原因のより適切な制御を可能にします。

# SensorApp の新機能

SensorApp には、次の新機能が含まれます。

- ポリシー テーブル：リスク カテゴリの設定（high、medium、および low）のリストを提供します。
- 回避保護：ノーマライザのインライン インターフェイス モード センサーをストリクト モードから非対称モードに切り換えます。
- センサー ヘルス メーター：センサー全体のヘルスの統計情報を提供します。
- トップ サービス：TCP、UDP、ICMP、および IP プロトコルの上位 10 個のインスタンスを提供します。
- セキュリティ メーター：脅威のカテゴリに入るアラートのプロファイルを設定し、この情報を赤、黄、および緑のバケットで報告します。これらのバケットの移行ポイントを設定できます。
- フロー状態のクリア：データベースをクリアできます。データベースをクリアすると、センサーが再起動時と同様に初期状態で起動します。
- 再起動ステータス：センサーの現在の起動段階および再起動段階を定期的に報告します。

# パケット フロー

パケットは、NIC によって受信され、IPS 共有ドライバにより、ユーザによってマッピングされたカーネル内のメモリ スペースに配置されます。パケットの前には、IPS ヘッダーが付加されます。また、各パケットには、パケットがシグニチャ イベント アクション プロセッサに到達したときにそのパケットを通過させるか拒否するかを示すフィールドも含まれます。

プロデューサは、ユーザによってマッピングされた共有カーネルのパケット バッファからパケットを取り出し、センサーのモデルに適したプロセッサを実装する処理機能を呼び出します。順序は、次のようになります。

- シングル プロセッサの実行

  タイム プロセッサ --> レイヤ 2 プロセッサ --> 拒否フィルタ プロセッサ --> フラグメント再構成プロセッサ --> 統計情報プロセッサ --> データベース プロセッサ --> シグニチャ分析プロセッサ --> ストリーム再構成プロセッサ --> シグニチャ イベント アクション プロセッサ

- デュアル プロセッサの実行

  実行スレッド 1 タイム プロセッサ --> レイヤ 2 プロセッサ --> 拒否フィルタ プロセッサ --> フラグメント再構成プロセッサ --> 統計情報プロセッサ --> データベース プロセッサ --> シグニチャ分析プロセッサ --> スレーブ ディスパッチ プロセッサ --> | 実行スレッド 2 データベース プロセッサ --> ストリーム再構成プロセッサ --> シグニチャ イベント アクション プロセッサ

# シグニチャ イベント アクション プロセッサ

シグニチャ イベント アクション プロセッサは、アラーム チャネル内のシグニチャ イベントから、シグニチャ イベント アクション オーバーライド、シグニチャ イベント アクション フィルタの処理を経由してシグニチャ イベント アクション ハンドラで処理されるまでのデータ フローを調整します。シグニチャ イベント アクション プロセッサは、次のコンポーネントから構成されます。

- アラーム チャネル

  SensorApp 検査パスからシグニチャ イベント処理にシグニチャ イベントを伝達するための領域となるユニット。

- シグニチャ イベント アクション オーバーライド

  リスク レーティング値に基づいて、アクションを追加します。シグニチャ イベント アクション オーバーライドは、設定済みのリスク レーティングしきい値の範囲内にあるすべてのシグニチャに適用されます。各シグニチャ イベント アクション オーバーライドは独立しており、各アクション タイプには別個の値が設定されています。

- シグニチャ イベント アクション フィルタ

  シグニチャ イベントのシグニチャ ID、アドレス、およびリスク レーティングに基づいてアクションを削除します。シグニチャ イベント アクション フィルタへ入力するのは、シグニチャ イベント アクション オーバーライドによって追加される可能性のあるアクションを持つシグニチャ イベントです。

  **（注）** シグニチャ イベント アクション フィルタが実行できるのはアクションの削除だけであり、新しいアクションの追加はできません。

  シグニチャ イベント アクション フィルタには、次のパラメータが適用されます。

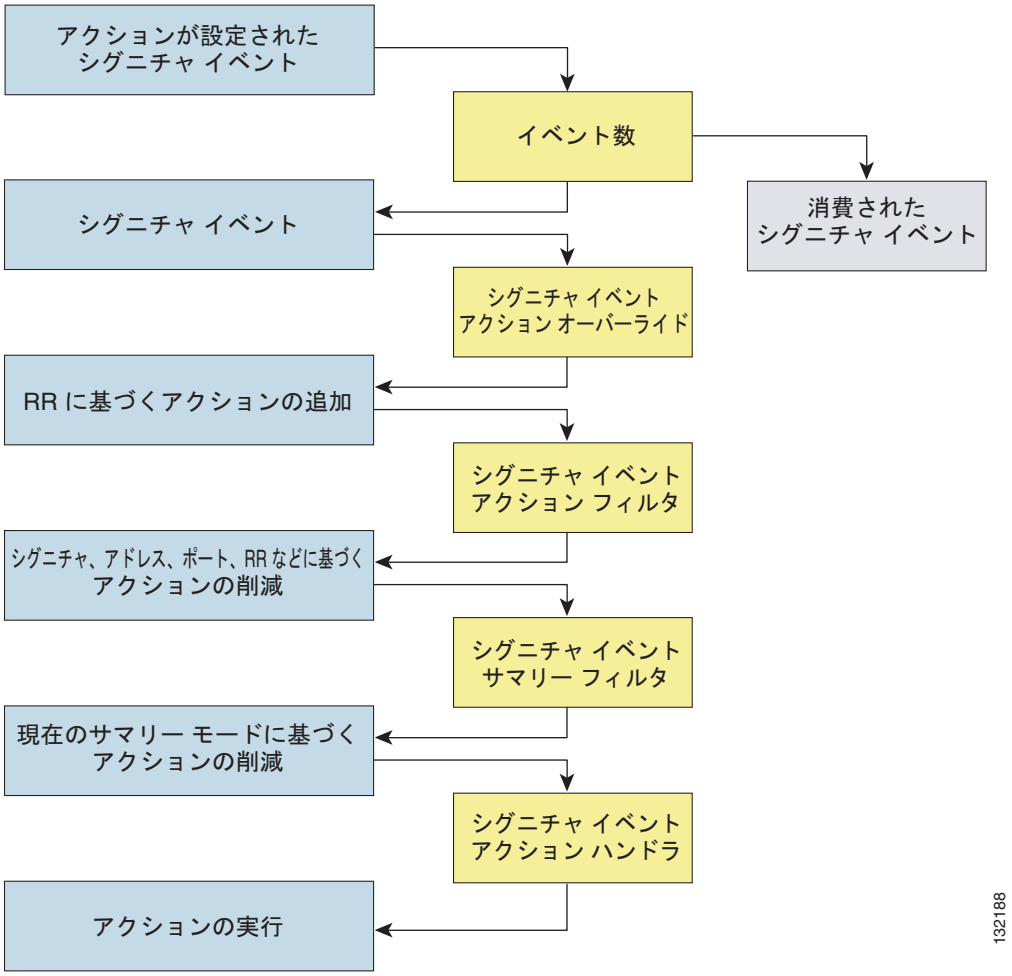
  - シグニチャ ID
  - サブシグニチャ ID
  - 攻撃者のアドレス
  - 攻撃者のポート
  - 攻撃対象のアドレス
  - 攻撃対象のポート
  - リスク レーティングしきい値の範囲
  - 削除するアクション
  - シーケンス識別子（任意）
  - ストップ ビットまたは継続ビット
  - アクション フィルタ行をイネーブルにするビット
  - 攻撃対象の OS の関連性または OS の関連性

- シグニチャ イベント アクション ハンドラ

  要求されたアクションを実行します。シグニチャ イベント アクション ハンドラから出力されるのは、実行中のアクションだけでなく、イベント ストアに書き込まれる evIdsAlert である可能性があります。

図 A-4（P.A-27）に、シグニチャ イベント アクション プロセッサを通過するシグニチャ イベントの論理的なフローと、このイベントのアクションで実行される操作を示します。これは、アラーム チャネルで受信された、アクションが設定されているシグニチャ イベントで開始され、シグニチャ イベントがシグニチャ イベント アクション プロセッサの機能コンポーネントを通過するときに上から下へ流れます。

**図 A-4　シグニチャ イベント アクション プロセッサを通過するシグニチャ イベント**




# CollaborationApp

ここでは、CollaborationApp について説明します。次のような構成になっています。

- 「CollaborationApp について」（P.A-28）
- 「アップデート コンポーネント」（P.A-28）
- 「error イベント」（P.A-29）

# CollaborationApp について

CollaborationApp は、MainApp および SensorApp のピアです。IDAPI 制御トランザクション、セマフォ、共有メモリ、ファイル交換などのさまざまなプロセス間通信テクノロジーを使用して、それらとのインターフェイスをとります。

グローバル相関サーバと CollaborationApp の間で、レピュテーション アップデートが交換されます。CollaborationApp は、4 つのアップデート コンポーネントを使用してセンサーと通信します。

- スコアの重みに関する一連の規則
- 一連の IP アドレスおよびアドレス範囲（規則およびアラートとともに、レピュテーション スコアの計算に必要な情報を提供します）
- トラフィックを常に拒否する必要のある、IP アドレスおよびアドレス範囲のリスト
- ネットワーク参加の設定（これにより、サーバは、センサーがテレメトリ データをサーバに送信するレートを制御できるようになります）

センサーは、コラボレーション情報をネットワーク参加サーバに送信します。センサーはグローバル相関サーバに対し、使用可能なコラボレーションの更新、およびグローバル相関サーバによる更新ファイルのダウンロード元のリストを照会します。

（注）　SensorApp は CollaborationApp よりも先に開始しますが、これらは非同期で初期化されます。したがって、レピュテーション更新サーバは、SensorApp でアップデートを受け入れる準備ができる前に、1 つ以上のグローバル相関の更新をダウンロードして適用を試みる可能性があります。更新サーバはアップデートをダウンロードして部分的に処理することもありますが、アップデートをコミットできるようになるには、SensorApp の準備完了まで待つ必要があります。

### 詳細情報

グローバル相関の詳細および設定方法については、第 10 章「グローバル相関の設定」を参照してください。

# アップデート コンポーネント

グローバル相関アップデート クライアントは、グローバル相関更新サーバとマニフェストを交換します。サーバ マニフェストを解析して、ダウンロード可能な新しいアップデートとそれらが存在する場所を判別した後、インストールされるアップデートのリストを作成します。すべてのアップデートが正常に適用されると、グローバル相関アップデート クライアントは、適用された各コンポーネントのアップデートをコミットし、新しいアップデートが使用可能になったことを SensorApp に通知し、クライアント マニフェストをアップデートして、コミットされた各コンポーネントの最新のアップデートを反映します。

クライアント マニフェストには、センサーの UDI が含まれています。これには、センサーのシリアル番号と、センサーが真正な Cisco IPS センサーであることを確認するためにサーバが使用する暗号化された共有秘密が含まれます。サーバ マニフェストには、各コンポーネントが使用できるアップデート ファイルのリストが含まれています。リスト内の各アップデート ファイルについて、サーバ マニフェストにはアップデートのバージョン、タイプ、順序、場所、ファイル転送プロトコルなどのデータが含まれています。

アップデート ファイルには 2 種類あります。コンポーネントのデータベースにある既存のデータをすべて置き換える完全なアップデート ファイルと、情報の追加、削除、または置換によって既存のレピュテーション データを変更する差分アップデートです。すべてのコンポーネントにすべてのアップデート ファイルが適用されると、作業データベースを置き換えることによって一時データベースがコミットされます。

認証と許可は、秘密暗号化メカニズムと復号キー管理によって実現されます。グローバル相関更新サーバは、クライアント マニフェストに含まれている共有秘密暗号化メカニズムを使用して、センサーを認証します。グローバル相関アップデート クライアントは、復号キー管理を使用してセンサーを許可します。グローバル相関更新サーバによって認証されたセンサーには、アップデート ファイルを復号化できるように、サーバ マニフェスト内の有効なキーが送信されます。

**注意**　グローバル相関をイネーブルにしているけれども DNS または HTTP プロキシ サーバを設定していない場合は、警告メッセージを受け取ります。この警告は、グローバル相関をディセーブルにするか、DNS または HTTP プロキシ サーバを追加するように通知するものです。

### 詳細情報

DNS またはプロキシ サーバを追加してグローバル相関をサポートする手順については、「ネットワーク設定値の変更」（P.4-1）を参照してください。

## error イベント

グローバル相関のアップデートが失敗した場合は、evError イベントが生成されます。エラー メッセージはセンサー統計情報に含まれています。次の条件が発生すると、重大度が Error のステータス メッセージになります。

- センサーがライセンスされていない
- DNS または HTTP プロキシ サーバが設定されていない
- マニフェストの交換が失敗した
- アップデート ファイルのダウンロードが失敗した
- アップデートの適用またはコミットが失敗した

グローバル相関がイネーブルになるようにホストまたはグローバル相関設定を編集して保存したけれども、DNS または HTTP プロキシ サーバが設定されていない場合、重大度レベルが Warning の evError イベントが生成されます。

### 詳細情報

センサー統計情報を表示する手順については、「統計情報の表示」（P.17-20）を参照してください。

# CLI

ここでは、IPS CLI について説明します。次のような構成になっています。

# CLI について

CLI は、Telnet、SSH、シリアル インターフェイスなどのすべての直接ノード アクセスについて、センサーのユーザ インターフェイスを提供します。センサー アプリケーションは CLI で設定します。基盤となる OS への直接接続は、サービス ロールを通じて許可されます。

# ユーザ ロール

ユーザ ロールには次の 4 つがあります。

- ビューア（Viewer）：設定およびイベントを表示できますが、自分のユーザ パスワード以外の設定データは修正できません。
- オペレータ（Operator）：すべてのデータを表示できるほか、次のオプションを修正できます。
  - シグニチャ チューニング（優先順位、無効 / 有効）
  - 仮想センサーの定義
  - 管理対象ルータ
  - 自分のユーザ パスワード
- 管理者（Administrator）：すべてのデータを表示できるほか、オペレータが修正できるすべてのオプションに加えて、次のオプションを修正できます。
  - センサーのアドレス設定
  - 設定エージェントまたはビュー エージェントとして接続が許可されたホストのリスト
  - 物理的なセンシング インターフェイスの割り当て
  - 物理インターフェイスの制御のイネーブル化またはディセーブル化
  - ユーザとパスワードの追加および削除
  - 新しい SSH ホスト キーおよびサーバ証明書の生成
- サービス（Service）：サービス権限を持つユーザはセンサーに 1 人だけ存在できます。サービス ユーザは、IDM または IME にログインできません。サービス ユーザは、CLI ではなく bash シェルにログインします。

  サービス ロールは、必要に応じて CLI をバイパスできる特殊なロールです。許可されるサービス アカウントは 1 つだけです。サービス ロールを持つアカウントは、トラブルシューティングの目的でのみ作成してください。管理者権限を持つユーザのみが、サービス アカウントを編集できます。

**注意**　サービス アカウントを作成するかどうかは、慎重に検討してください。サービス アカウントは、システムへのシェル アクセスを提供するため、システムが脆弱になります。ただし、管理者のパスワードが失われた場合は、サービス アカウントを使用してパスワードを作成できます。状況を分析して、システムにサービス アカウントを存在させるかどうかを決定してください。

サービス アカウントにログインすると、次の警告が表示されます。

```
************************ WARNING ****************************************************
UNAUTHORIZED ACCESS TO THIS NETWORK DEVICE IS PROHIBITED.
This account is intended to be used for support and troubleshooting purposes only.
Unauthorized modifications are not supported and will require this device to be
re-imaged to guarantee proper operation.
*************************************************************************************
```

# サービス アカウント

サービス アカウントは、サポートとトラブルシューティングのツールです。これによって TAC は、CLI シェルではなくネイティブ オペレーティング システムのシェルにログインすることができます。これは、デフォルトではセンサーには存在しません。TAC がセンサーのトラブルシューティングのために使用できるように、ユーザがこれを作成する必要があります。

1 つのセンサーで使用できるサービス アカウントは 1 つだけです。また、1 つのサービス ロールで使用できるアカウントも 1 つだけです。サービス アカウントのパスワードが設定またはリセットされると、root アカウントのパスワードが同じパスワードに設定されます。そのため、サービス アカウントのユーザはこの同じパスワードを使用して、root に su できます。サービス アカウントが削除されると、root アカウントのパスワードはロックされます。

サービス アカウントは、設定の目的で使用されることを想定していません。TAC の指示に基づき、サービス アカウントからセンサーに対して加えられた変更だけがサポートされます。サービス アカウントからオペレーティング システムに追加のサービスを加えること、およびそれを実行することは、他の IPS サービスの適切な実行に影響するため、シスコはこれをサポートしません。TAC は、追加のサービスが加えられたセンサーをサポートしません。

サービス アカウントへのログインは、ログ ファイル /var/log/.tac を確認することによって追跡できます。このファイルは、サービス アカウントによるログインの記録でアップデートされます。

（注）　Cisco IPS には、CLI、IDM、または IME から使用可能ないくつかのトラブルシューティング機能が組み込まれています。ほとんどのトラブルシューティングの状況で、サービス アカウントは不要です。TAC の指示の下にサービス アカウントを作成してトラブルシューティングを実行しなければならない、きわめてまれな問題もあります。サービス アカウントを使用すると、CLI に組み込まれた保護をバイパスし、root 権限でセンサーにアクセスすることができます。これは他の場合はディセーブルになっています。特別な理由で必要となる場合を除いては、サービス アカウントを作成しないことを推奨します。サービス アカウントは、不要になったら削除する必要があります。

# 通信

ここでは、Cisco IPS で使用される通信プロトコルについて説明します。次のような構成になっています。

# IDIOM

IDIOM は、IPS によって報告されるイベント メッセージ、および侵入検知システムの設定と制御に使用される操作メッセージを定義するデータ形式の標準です。これらのメッセージは、IDIOM XML スキーマに準拠した XML ドキュメントで構成されています。

IDIOM は、イベントと制御トランザクションという 2 種類のインタラクションをサポートしています。イベント インタラクションは、アラートなどの IPS イベントを交換するために使用されます。IDIOM は、イベント インタラクションにイベント メッセージとエラー メッセージという 2 種類のメッセージを使用します。制御トランザクションは、ホストが別のホストでアクションを開始したり、別のホストの状態を変更または読み取るための手段です。制御トランザクションでは、要求、応答、設定、エラーの 4 種類の IDIOM メッセージが使用されます。1 つのホスト内のアプリケーション インスタンス間で通信されるイベントおよび制御トランザクションは、ローカル イベントまたはローカル制御トランザクションと呼ばれ、ローカル IDIOM メッセージと総称されます。異なるホスト間で通信されるイベントおよび制御トランザクションは、リモート イベントおよびリモート制御トランザクションと呼ばれ、リモート IDIOM メッセージと総称されます。

（注）　IDIOM の大部分は、IDCONF、SDEE、および CIDEE によって置き換えられています。

# IDCONF

Cisco IPS は、XML ドキュメントを使用して自身の設定を管理します。IDCONF は、Cisco IPS 制御トランザクションを含めた XML スキーマを指定します。IDCONF スキーマは、設定ドキュメントの内容を指定するのではなく、設定ドキュメント作成の基になるフレームワークと構成単位を指定します。これは、特定の機能でサポートされる属性を使用することにより、それを設定することができないプラットフォームや機能の場合は IPS マネージャおよび CLI がその機能を無視できるメカニズムを提供します。

IDCONF メッセージは IDIOM の要求および応答メッセージ内にラップされます。

次に、IDCONF の例を示します。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<request xmlns="http://www.cisco.com/cids/idiom" schemaVersion="2.00">
  <editConfigDelta xmlns="http://www.cisco.com/cids/idconf">
    <component name="userAccount">
      <config typedefsVersion="2004-03-01" xmlns="http://www.cisco.com/cids/idconf">
        <struct>
           <map name="user-accounts‟ editOp=‟merge”>
             <mapEntry>
               <key>
                <var name="name">cisco</var>
              </key>
               <struct>
                <struct name="credentials">
                  <var name="role">administrator</var>
                </struct>
               </struct>
            </mapEntry>
          </map>
        </struct>
      </config>
    </component>
  </editDefaultConfig>
</request>
```

# SDEE

IPS は、侵入アラートやステータス イベントなどのさまざまなタイプのイベントを生成します。IPS は、IPS 業界最先端の独自プロトコル、SDEE を使用して、管理アプリケーションなどのクライアントにイベントを伝達します。SDEE は、セキュリティ デバイスのイベントを伝達するための製品に依存しない標準です。SDEE には、各種のセキュリティ デバイスによって生成されるイベントの伝達に必要な拡張機能が追加されています。

SDEE を使用してクライアントにイベントを伝達するシステムは、SDEE プロバイダーと呼ばれます。SDEE では、HTTP または HTTP over SSL および TLS プロトコルを使用してイベントが転送されるように指定します。HTTP または HTTPS を使用する場合は、SDEE クライアントが HTTP 要求の発信側となり、SDEE プロバイダーは HTTP サーバとして動作します。

IPS には、HTTP または HTTPS 要求を処理する Web サーバが含まれています。Web サーバは、実行時にロード可能なサーブレットを使用して、各種の HTTP 要求を処理します。各サーブレットは、そのサーブレットに関連付けられた URL に送られる HTTP 要求を処理します。SDEE サーバは、Web サーバ サーブレットとして実装されます。

SDEE サーバは、許可された要求のみを処理します。要求は、クライアントの ID を認証し、クライアントの権限レベルを確認するために Web サーバから発信されている場合に許可されます。

# CIDEE

CIDEE は、Cisco IPS が使用する SDEE への拡張を指定します。CIDEE 標準は、Cisco IPS がサポートする可能性のあるすべての拡張を指定しています。特定のシステムでは、CIDEE の拡張のサブセットを実装する場合もあります。ただし、必須と指定された拡張は、すべてのシステムでサポートする必要があります。

CIDEE は、Cisco IPS 固有のセキュリティ デバイス イベント、および SDEE の evIdsAlert 要素に対する IPS の拡張を指定します。

CIDEE は次のイベントをサポートします。

- evError：エラー イベント

  CIDEE プロバイダーがエラーまたは警告状態を検出したときに、CIDEE プロバイダーによって生成されます。evError イベントには、エラー コードとテキストによるエラーの説明が含まれます。

- evStatus：ステータス メッセージ イベント

  ホストで潜在的に注意が必要な何らかの状態が発生したことを示すために、CIDEE プロバイダーによって生成されます。ステータス イベントでは、異なるタイプのステータス メッセージが報告される可能性があります（イベントごとに 1 つのメッセージ）。各タイプのステータス メッセージには、そのステータス メッセージで説明されている発生タイプに固有のデータ要素のセットが含まれます。ステータス メッセージの多くに含まれる情報は、監査の目的で役立ちます。エラーと警告はステータス情報とは見なされず、evStatus ではなく evError を使用して報告されます。

- evShunRqst：ブロック要求イベント

  ネットワークのブロッキングを処理するサービスがブロック アクションを開始することを示すために生成されます。

次に、CDIEE 拡張イベントの例を示します。

```
<sd:events xmlns:cid="http://www.cisco.com/cids/2004/04/cidee"
xmlns:sd="“http://example.org/2010/08/sdee”>
 <sd:evIdsAlert eventId="1042648730045587005" vendor="Cisco‟ severity="medium">
  <sd:originator>
    <sd:hostId>Beta4Sensor1</sd:hostId>
    <cid:appName>sensorApp</cid:appName>
```

```
  <cid:appInstanceId>8971</cid:appInstanceId>
</sd:originator>
<sd:time offset="0" timeZone="UTC">1043238671706378000</sd:time>
<sd:signature description="IOS Udp Bomb" id="4600" cid:version="S37">
  <cid:subsigId>0</cid:subsigId>
</sd:signature> …
```

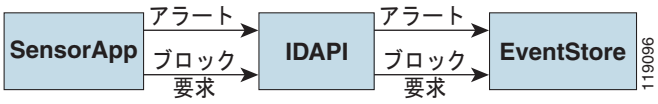
# IDAPI

IPS アプリケーションは、IDAPI というプロセス間通信 API を使用して内部的な通信を処理します。IDAPI はイベント データを読み書きし、制御トランザクションのメカニズムを提供します。IDAPI は、すべてのアプリケーションが通信の際に使用するインターフェイスです。

SensorApp は、そのインターフェイス上のネットワーク トラフィックをキャプチャし、分析します。シグニチャが一致すると、SensorApp はアラートを生成します。このアラートはイベント ストアに格納されます。シグニチャがブロッキング応答アクションを実行するように設定されていると、SensorApp はブロック イベントを生成します。このイベントもイベント ストアに格納されます。

図 A-5 に、IDAPI インターフェイスを示します。

**図 A-5　　IDAPI**




各アプリケーションは、イベントおよび制御トランザクションを送受信するように IDAPI に登録します。IDAPI は次のサービスを提供します。

- 制御トランザクション
  - 制御トランザクションを開始する。
  - インバウンド制御トランザクションを待機する。
  - 制御トランザクションに応答する。
- IPS イベント
  - リモート IPS イベントをサブスクライブする。受信したリモート IPS イベントは、イベント ストアに格納されます。
  - イベント ストアから IPS イベントを読み取る。
  - イベント ストアに IPS イベントを書き込む。

IDAPI は、アトミックなデータ アクセスを実現するために必要な同期メカニズムを備えています。

# Cisco IPS のファイル構造

Cisco IPS のディレクトリ構造は次のとおりです。

- /usr/cids/idsRoot：メインのインストール ディレクトリ。
- /usr/cids/idsRoot/shared：システムのリカバリ中に使用されるファイルが格納されます。
- /usr/cids/idsRoot/var：センサーの実行中に動的に作成されたファイルが格納されます。
- /usr/cids/idsRoot/var/updates：アップデート インストレーション用のファイルとログが格納されます。

- /usr/cids/idsRoot/var/virtualSensor：SensorApp が正規表現を分析するために使用するファイルが格納されます。
- /usr/cids/idsRoot/var/eventStore：イベント ストア アプリケーションが含まれます。
- /usr/cids/idsRoot/var/core：システムのクラッシュ時に作成される重要なファイルが格納されます。
- /usr/cids/idsRoot/var/iplogs：iplog ファイルのデータが格納されます。
- /usr/cids/idsRoot/bin：バイナリ実行可能ファイルが含まれます。
- /usr/cids/idsRoot/bin/authentication：認証アプリケーションが含まれます。
- /usr/cids/idsRoot/bin/cidDump：技術サポート向けのデータを収集するスクリプトが含まれます。
- /usr/cids/idsRoot/bin/cidwebserver：Web サーバ アプリケーションが含まれます。
- /usr/cids/idsRoot/bin/cidcli：CLI アプリケーションが含まれます。
- /usr/cids/idsRoot/bin/nac：ARC アプリケーションが含まれます。
- /usr/cids/idsRoot/bin/logApp：ロガー アプリケーションが含まれます。
- /usr/cids/idsRoot/bin/mainApp：メイン アプリケーションが含まれます。
- /usr/cids/idsRoot/bin/sensorApp：センサー アプリケーションが含まれます。
- /usr/cids/idsRoot/bin/collaborationApp：コラボレーション アプリケーションが含まれます。
- /usr/cids/idsRoot/bin/falcondump：IDSM2 のセンシング ポートでパケット ダンプを取得するアプリケーションが含まれます。
- /usr/cids/idsRoot/etc：センサーのコンフィギュレーション ファイルが含まれます。
- /usr/cids/idsRoot/htdocs：Web サーバの IDM ファイルが含まれます。
- /usr/cids/idsRoot/lib：センサー アプリケーションのライブラリ ファイルが含まれます。
- /usr/cids/idsRoot/log：デバッグ用のログ ファイルが含まれます。
- /usr/cids/idsRoot/tmp：センサーの実行中に作成される一時ファイルが格納されます。

# Cisco IPS アプリケーションの概要

表 A-2 に、IPS を構成するアプリケーションの概要を示します。

**表 A-2　アプリケーションの概要**

| アプリケーション | 説明 |
|---|---|
| AuthenticationApp | IP アドレス、パスワード、およびデジタル証明書に基づいてユーザを許可および認証します。 |
| Attack Response Controller | ARC は、各センサー上で実行されます。各 ARC は、ローカル イベント ストアのネットワーク アクセス イベントをサブスクライブします。ARC の設定には、センサー、およびそのローカル ARC が制御するネットワーク アクセス デバイスのリストが含まれます。ネットワーク アクセス イベントをマスター ブロッキング センサーに送信するように設定されている場合、ARC は、デバイスを制御するリモート ARC に対してネットワーク アクセス コントロール トランザクションを開始します。これらのネットワーク アクセス アクション制御トランザクションは、IPS マネージャがネットワーク アクセス アクションを発行するときにも使用されます。 |

表 A-2　アプリケーションの概要 （続き）

| アプリケーション | 説明 |
|---|---|
| CLI | コマンドライン入力を受け付け、IDAPI を使用してローカル設定を変更します。 |
| CollaborationApp | グローバル相関データベースを通じて他のデバイスと情報を共有することにより、すべてのデバイスの総合的効果を高めます。 |
| Control Transaction Server[1] | リモート クライアントからの制御トランザクションを受け付け、ローカル制御トランザクションを開始して、リモート クライアントに応答を返します。 |
| Control Transaction Source[2] | リモート アプリケーションに向けられた制御トランザクションを待機し、制御トランザクションをリモート ノードに転送し、応答を発信側に返します。 |
| IDM | HTML IPS 管理インターフェイスを提供する Java アプレットです。 |
| IME | イベントを表示およびアーカイブするためのインターフェイスを提供する Java アプレットです。 |
| InterfaceApp | バイパスおよび物理設定を処理し、ペアになっているインターフェイスを定義します。物理設定は、速度、デュプレックス、および管理状態です。 |
| ロガー | アプリケーションのすべてのログ メッセージをログ ファイルに書き出し、アプリケーションのエラー メッセージをイベント ストアに書き出します。 |
| MainApp | 設定を読み取ってアプリケーションを起動し、アプリケーションの開始および終了とノードの再起動を扱い、ソフトウェアのアップグレードを処理します。 |
| NotificationApp | アラート、ステータス、およびエラー イベントによってトリガーされたときに SNMP トラップを送信します。NotificationApp はパブリック ドメイン SNMP エージェントを使用します。SNMP GET は、センサーの全般的なヘルスに関する情報を提供します。 |
| SDEE サーバ[3] | リモート クライアントからイベントの要求を受け付けます。 |
| SensorApp | モニタされているネットワーク上のトラフィックをキャプチャして分析し、intrusion および network access イベントを生成します。ロギングをオン/オフする IP ロギング制御トランザクション、および IP ログ ファイルを送信および削除する IP ロギング制御トランザクションに応答します。 |
| Web サーバ | リモート HTTP クライアント要求を待機し、適切なサーブレット アプリケーションを呼び出します。 |

1. これは Web サーバ サーブレットです。
2. これは、リモート制御トランザクション プロキシです。
3. これは Web サーバ サーブレットです。