CHAPTER 1

Cisco VNMC XML API の概要

ここでは、Cisco VNMC XML アプリケーション プログラミング インターフェイス（API）について簡単に説明します。

この章の内容は、次のとおりです。

Cisco VNMC および XML API について

ここでは、次の内容について説明します。

Cisco VNMC の概要

Cisco Virtual Network Management Center（VNMC）は、Cisco Virtual Security Gateway（VSG）および Cisco Nexus 1000V シリーズ スイッチのデバイスおよびセキュリティ ポリシーを一元的に管理できる仮想アプライアンスです。Cisco VNMC は、マルチテナント操作用に設計されており、仮想データセンターおよびクラウド環境向けに、シームレスでスケーラブルな、自動化を中心とした管理を実現します。Cisco VNMC では、Web ベースの GUI、CLI、および XML API メソッドを使用することにより、データセンター全体に展開された Cisco VSG を一元的に管理できます。

マルチテナント機能とは、複数のクライアント組織またはテナントが、共有された物理インフラストラクチャに展開される、独自の仮想コンピュータ、ネットワーク、およびストレージ リソースを所有できるアーキテクチャ原則を指します。複数のテナントは同じインフラストラクチャ上で共存できますが、各テナントは仮想リソースの管理権限を持つことができます。システムは、コンピュータ、ネットワーク、ストレージ、およびセキュリティ ポリシーを含む、各テナントに対して指定された SLA を満たすよう設計されています。

Cisco VNMC は、情報モデル方式アーキテクチャに基づいて構築され、各管理対象デバイスはそのサブコンポーネントによって表されます。このアーキテクチャにより、Cisco VNMC では、マルチテナント インフラストラクチャを素早く簡単にセキュアにすることができます。

VNMC 管理情報モデル

Cisco VNMC サービス コンポーネントを構成するすべての物理および論理コンポーネントは、階層管理情報モデルで表されます。このモデルを管理情報ツリー（MIT）といいます。このツリー内の各ノードは、管理ステータスと動作ステータスを含む、管理対象オブジェクト（またはオブジェクトのグループ）を表します。階層構造は、最上部から始まり、親ノードと子ノードを含みます。このツリー内の各ノードは管理対象オブジェクトであり、各オブジェクトは、オブジェクトとツリー内の位置を示す一意の識別名（DN）を持ちます。

管理対象オブジェクトは、ポリシー、ルール、セキュリティ プロファイル、Cisco VSG インスタンスなどの Cisco VNMC 管理対象エンティティを抽象化したものです。特定の管理対象オブジェクトは、ユーザが作成するのではなく、Cisco VNMC によって自動的に作成されます。API を呼び出すことにより、オブジェクトは MIT から読み取られ、MIT に書き込まれます。個々の Cisco VNMC サービス コンポーネントの情報モデルは、データ管理エンジン（DME）によって一元的に保存および管理されます。ユーザが Cisco VNMC サービス コンポーネントに対して管理上の変更（コンピュータ ファイアウォール プロファイルと Cisco VSG の関連付けなど）を開始すると、DME により最初にその変更が情報モデルに適用され、続いてその変更が実際の Cisco VSG に適用されます。この方法を、モデル方式フレームワークといいます。

Cisco VNMC のコンポーネント

Cisco VNMC は、管理、テナント管理、ポリシー管理、リソース管理などを行うモジュール形式の機能を提供する複数のサービス コンポーネントで構成されます。これらのコンポーネントは、サービス プロバイダーまたはアプリケーションとも呼ばれます。各コンポーネントは、一意の URL でアクセスされます。

次の項では、これらのコンポーネントについて説明します。

- 「Management Controller」（P.1-3）
- 「サービス レジストリ」（P.1-3）
- 「Resource Manager」（P.1-3）
- 「Policy Manager」（P.1-4）
- 「VM Manager」（P.1-4）

各コンポーネントは、独自のデータ モデルと、モデル方式サービス要求を処理する DME（データ管理エンジン）を持ちます。各コンポーネントは、独自の管理情報ツリー ストレージ（メモリ内とパーシスタンス ストアの両方）を保持します。異なるサービス コンポーネント間でのデータ共有は、アドホックのサービス間 API 通信、パブリッシュまたはサブスクライブ メソッドによって実現されます。

Management Controller

Management Controller（コア サービスとも呼ばれます）は、Cisco VNMC 仮想マシンにシステム関連サービスを提供します。提供されるサービスは次のとおりです。

- ローカルまたは LDAP モードでユーザ ログイン認証および承認を管理します。
- ロケール、ロール、トラスト ポイントなどのアクセス コントロールを管理します。
- IP アドレス、サブネット マスク、ゲートウェイ、ホスト名などのシステム情報を管理します。
- データベース バックアップ、データ エクスポート、データ インポートなどのシステム保守を実行します。
- 監査ログ、障害、イベント ログ、コア ダンプ ファイルなどのシステム診断情報を保持します。

Management Controller のタイプは mgmt-controller です。すべての Management Controller 関連の要求に対して API URL でこのサービス タイプを使用します。

サービス レジストリ

Cisco VNMC は分散アーキテクチャに基づいて設計されており、サービス レジストリが中央サービス レポジトリとなって、登録されたすべての管理対象エンドポイント（VSM、Cisco VSG）とサービス プロバイダー（Policy Manager や Resource Manager など）に関する情報を保持します。

サービス エンドポイントとサービス プロバイダーは、サービス レジストリに動的に登録され、サービス レジストリから該当するサービス コンポーネントに関する情報を取得します。また、サービス レジストリはテナント管理も行います。提供されるサービスは次のとおりです。

- 組織（テナント、データセンター、vApp、階層）を作成、削除、および更新します。組織の変更は Policy Manager と Resource Manager に自動的に伝播され、ポリシーとリソースが、対象となる組織に割り当てられます。
- 登録されたサービス（プロバイダー、エンドポイント、管理コントローラ、サービス レジストリ）に関する情報を保持します。
- 監査ログ、障害、イベント ログなどの診断情報を保持します。

サービス レジストリのタイプは service-reg です。すべてのサービス レジストリ関連の要求に対して API URL でこのサービス タイプを使用します。

Resource Manager

Resource Manager は、論理的なコンピュータ ファイアウォールと、実際の Cisco VSG との関連付けを管理します。コンピュータ ファイアウォールを Cisco VSG に関連付けると、デバイス設定プロファイル情報（コンピュータ ファイアウォールによって定義されます）は実際の Cisco VSG にプッシュされます。これにより、Cisco VSG が Policy Manager からセキュリティ プロファイルおよびポリシーをダウンロードします。提供されるサービスは次のとおりです。

- Cisco Virtual Security Gateways（VSG）と Virtual Supervisor Modules（VSM）のインベントリを保持します。
- コンピュータ ファイアウォールを定義し、プロビジョニングのために Cisco VSG と関連付けます。
- VMWare vCenter インスタンスと統合し、VM 属性を取得します。
- 検出された仮想マシンのインベントリを保持し、これらの仮想マシンを次の情報とともに Cisco VSG インスタンスに分散します。
  - VM 属性（名前、ハイパーバイザ、親 vApp、クラスタ）
  - vNIC 属性（ポート プロファイル名、IP アドレス）

- Cisco VSG のプールを管理します。
- VSG のヘルス ステータスと障害を保持します。
- 監査ログ、障害、イベント ログなどの診断情報を保持します。

Resource Manager のタイプは resource-mgr です。すべての Resource Manager 関連の要求に対して API URL でこのサービス タイプを使用します。

Policy Manager

Policy Manager は、デバイス設定プロファイル、セキュリティ ポリシー、および関連するすべてのアーティファクトの中央レポジトリです。コンピュータ ファイアウォールが Resource Manager から Cisco VSG と関連付けられた場合、Cisco VSG はデバイス プロファイル、セキュリティ プロファイル、および参照されたすべてのポリシーを解決するために Policy Manager に問い合わせます。Cisco VSG は、Policy Manager から取得された情報に基づいて設定されます。この場合のサービスは次のとおりです。

- プロファイルを定義します。
- ポリシー セットを定義し、ポリシーを割り当てます。
- 次の条件に基づいてポリシー ルールを定義します。
  - プロトコル、送信元または宛先、IP アドレス、ポートなどのネットワーク属性
  - インスタンス名、ゲスト OS、ゾーン、親アプリケーション、ポート プロファイル、クラスタ、リソース プール、ホスト名、ハイパーバイザなどの仮想マシン属性
  - カスタム属性
- ゾーンを定義します。
- オブジェクト グループを定義します。
- セキュリティ プロファイル ディクショナリとカスタム属性を定義します。
- セキュリティ プロファイルを定義し、ポリシー セットを割り当てます。
- ファイアウォール デバイス プロファイルを定義します。
- NTP、DNS、syslog、および障害に対して Cisco VNMC システム管理デバイス プロファイルおよびポリシーを定義します。
- ポリシー、セキュリティ プロファイル、デバイス プロファイル、および関連するオブジェクトを Cisco VSG インスタンスに分散します。
- 監査ログ、障害、イベント ログなどの診断情報を保持します。

Policy Manager のサービス タイプは policy-mgr です。すべての Policy Manager 関連の要求に対して API URL でこのサービス タイプを使用します。

VM Manager

VM Manager は、VMWare vCenter と対話し、vCenter から取得された VM 情報を保持するために使用します。VM Manager は、ユーザがアクセスできるサービスがないバックエンド サービスです。VM Manager のサービス タイプは vm-manager です。

Cisco VNMC XML API の概要

Cisco VNMC XML API を使用すると、Cisco Virtual Network Management Center との統合または対話をプログラミングによって実行できます。API インターフェイスは、HTTPS プロトコルを使用して XML ドキュメントを受け取ります。開発者は、任意のプログラミング言語を使用して API メソッドを含む XML ドキュメントを生成できます。設定およびステータス情報は、管理情報ツリー（XML API を介して完全に公開されます）という階層ツリー構造に格納されます。

API モデルは、再帰的に駆動され、アプリケーション開発のために主要な機能を提供します。たとえば、変更は、単一のオブジェクト、オブジェクトのサブツリー、またはオブジェクト ツリー全体に対して行うことができます。また、変更は、オブジェクトの単一の属性に行ったり、単一の API コールで Cisco VNMC 構造全体に適用したりできます。

API は寛容モードで動作します。不明な属性は、内部のデータ管理エンジン（DME）で保持されているデフォルト値（該当する場合）で置き換えられます。複数の管理対象オブジェクト（つまり、ポリシー）を設定するときにその管理対象オブジェクトのいずれかを設定できない場合、API はその動作を停止し、設定を元の状態に戻し、API プロセスを停止して障害を通知します。

API は、スケーラビリティとパフォーマンスを向上させるために非同期動作モデルを使用します。完了するのに時間がかかるプロセスはノンブロッキングです（高速な API プロセスは処理を続行できます）。プロセスは、有効な要求時に成功メッセージを受け取り、タスクが完了すると完了メッセージを受け取ります。

完全なイベント サブスクリプションがサポートされます。Cisco VNMC は、発生したイベント（管理対象オブジェクトの変更など）に関する通知をすべてのサブスクライバに送信し、ステータス変更のタイプを示します。

管理対象オブジェクト データ モデルの今後の更新では、後方互換性を維持するために既存のオブジェクト モデルに準拠します。製品アップグレード時に既存のプロパティが変更された場合、変更はアップグレード後のデータベース ロード中に処理されます。新しいプロパティにはデフォルト値が割り当てられます。

Cisco VNMC では、モデル方式アーキテクチャを使用しています。このアーキテクチャでは、変更は最初に管理対象オブジェクトの形式の論理構造に適用されます。次に、管理対象オブジェクトは、エンドポイントに変更を適用し、必要な状態（設定）のエンドポイントを実現します。

API の動作はトランザクション型で、単一のデータ モデルで終了します。Cisco VNMC は、すべてのエンドポイント（Cisco VSG、VSM）通信（ステータス更新など）を行います。ユーザはエンドポイントと直接通信できないため、開発者は分離された個々のコンポーネント設定を管理する必要がありません。

Cisco VNMC API モデルには、次のプログラミング エンティティが含まれます。

- クラス：管理情報ツリーのオブジェクトのプロパティとステータス
- メソッド：1 つまたは複数のオブジェクトに対して API が実行するアクション
- タイプ：オブジェクト ステータスに値をマッピングするオブジェクト プロパティ（つまり、policyDeviceProfile）

一般的な要求は Cisco VNMC サービス コンポーネントの DME に送信され、トランザクタ キューに FIFO の順序で配置されます。トランザクタはこのキューから要求を取得し、要求を解釈して認可チェックを実行します。要求が確認されると、トランザクタは管理情報ツリーを更新します。このプロセスは、単一のトランザクションで実行されます。

Cisco VNMC データ モデル スキーマ

すべての Cisco VNMC サービス プロバイダーに対するデータ モデル スキーマ ファイルは Cisco VNMC サーバでパッケージ化され、次の URL を使用して取得できます。

```
https://<vnmc ip address>/schema
```

スキーマ リストには、次のものが含まれます。

- core.in.xsd：Management Controller 設定 API とデータ モデル
- core.out.xsd：Management Controller クエリー API とデータ モデル
- policy-mgr.in.xsd：Policy Manager 設定 API とデータ モデル
- policy-mgr.out.xsd：Policy Manager クエリー API とデータ モデル
- resource-mgr.in.xsd：Resource Manager 設定 API とデータ モデル
- resource-mgr.out.xsd：Resource Manager クエリー API とデータ モデル
- service-reg.in.xsd：サービス レジストリ設定 API とデータ モデル
- service-reg.out.xsd：サービス レジストリ クエリー API とデータ モデル

Cisco VNMC サービスへのアクセス

Cisco VNMC サービスには、HTTPS プロトコルを介した XML API 要求を使用してアクセスできます。HTTPS 要求の URL 形式は次のとおりです。

```
https://<vnmc ip address>/xmlIM/<service type>
```

<service type> は、「Cisco VNMC のコンポーネント」の項で説明されている、該当するサービス プロバイダーのサービス タイプです。たとえば、Policy Manager に要求を送信するには、次の URL でサービス タイプ policy-mgr を使用します。

```
https://<vnmc ip address>/xmlIM/policy-mgr
```

オブジェクトの命名

特定のオブジェクトは、識別名（DN）または相対名（RN）で識別できます。

ここでは、次の内容について説明します。

- 「識別名」（P.1-6）
- 「相対名」（P.1-7）

識別名

DN を使用すると、ターゲット オブジェクトを明確に識別できます。DN は、一連の相対名から構成される次の形式を持ちます。

```
dn = {rn}/{rn}/{rn}/{rn}...
```

次に例を示します。

```
org-root/org-Cisco/zone-trustedServers-0
```

例では、DN は、オブジェクト ツリーの最上部から Zone オブジェクト（zone-trustedServers-0）までの完全修飾パスを提供しています。DN は、API コールが動作する管理対象オブジェクトを指定します。

```
< ... dn = "org-root/org-Cisco/zone-trustedServers-0" />
```

相対名

相対名（RN）は、親オブジェクトのコンテキスト内でオブジェクトを識別します。オブジェクトの DN は、次の形式で親 DN と RN から構成されます。

```
dn = <parent dn>/{rn}
```

たとえば、テナント Cisco に名前 trustedServers-0 が定義されている Zone オブジェクトの場合、その RN は zone-trustedServers-0 です。親テナント DN は org-root/org-Cisco であり、DN は org-root/org-Cisco/zone-trustedServers-0 です。

API メソッド カテゴリ

Cisco VNMC との対話に使用されるメソッドは、4 つのカテゴリに分けられます。各 API はメソッドであり、各メソッドは XML ドキュメントに対応します。これらのメソッドについては、次の項で説明します。

- 「認証方式」（P.1-7）
- 「クエリー メソッド」（P.1-8）
- 「設定メソッド」（P.1-10）
- 「イベント サブスクリプション メソッド」（P.1-10）

（注）　このマニュアルのいくつかのコード例では、用語 <real_cookie> は 1217377205/85f7ff49-e4ec-42fc-9437-da77a1a2c4bf などの実際の Cookie に置き換えられます。Cisco VNMC の Cookie は 47 文字の文字列です。これは、Web ブラウザがセッション情報を保持するためにローカルに保存する種類の Cookie ではありません。

認証方式

認証方式は、セッションを認証して保持します。次のような認証方式があります。

- aaaLogin：この方式は、Cisco VNMC にログインする最初の方式です。
- aaaRefresh：この方式は、現在の認証 Cookie を更新します。
- aaaLogout：この方式は現在のセッションを終了し、現在の認証 Cookie を非アクティブ化します。

認証方式は、アクティブな Cisco VNMC セッションを開始し、保持します。他の API コールが許可される前に、正常な認証を実行する必要があります。API 要求は Cookie により認証されます。

接続セッションが確立および認証されると、応答で Cookie が返されます。これは 7200 秒（120 分）有効です。Cookie は、有効期限が切れないようセッション期間中に更新する必要があります。各更新操作により、デフォルトの期間の間有効な Cookie が作成されます。

接続セッションを確立し、有効な Cookie を取得する場合は、aaaLogin を使用します。セッションを保持し、Cookie をアクティブに保つ場合は、aaaRefresh を使用します。セッションを終了（また、Cookie を無効化）する場合は、aaaLogout を使用します。Cisco VNMC に対して一度に最大 256 個のセッションを開くことができます。最大セッション制限に到達すると、後続するログイン要求が拒否されます。

操作は HTTP の post メソッドを使用して実行されます。Cisco VNMC は、ポート 443 で HTTPS プロトコルだけをサポートします。HTTP のエンベロープには XML の設定が含まれます。

クエリー メソッド

クエリー メソッドは、Cisco VNMC オブジェクトの現在の設定ステータスに関する情報を取得します。次にクエリー例を示します。

- configResolveDn：DN によりオブジェクトを取得します。
- configResolveDns：DN セットによりオブジェクトを取得します。
- configResolveClass：該当するクラスのオブジェクトを取得します。
- configResolveClasses：複数のクラスのオブジェクトを取得します。
- configFindDnsByClassId：指定されたクラスの DN を取得します。
- configResolveChildren：オブジェクトの子オブジェクトを取得します。
- configResolveParent：オブジェクトの親オブジェクトを取得します。
- configScope：管理情報ツリーの DN に対してクラス クエリーを実行します。

ほとんどのクエリー メソッドは、引数 inHierarchical、ブール値 true/yes または false/no を持ちます。true の場合、inHierarchical 引数はすべての子オブジェクトを返します。次に例を示します。

```
<configResolveDn … inHierarchical="false"></>
<configResolveDn … inHierarchical="true"></>
```

また、API クエリー メソッドは、コールを再帰的にするかどうか（つまり、他のオブジェクトまたは親オブジェクトを参照し返すオブジェクトに従うかどうか）を指定する inRecursive 引数を持つ場合があります。

クエリー フィルタ

この API は、クエリー メソッドの有用性を高めるためにフィルタ セットを提供します。これらのフィルタは、クエリーの一部として渡すことができ、必要な結果セットを特定するために使用されます。フィルタは次のように分類されます。

- Simple フィルタ
- Property フィルタ
- Composite フィルタ
- Modifier フィルタ

Simple フィルタ

Simple フィルタには true と false の 2 つがあります。これら 2 つのフィルタは、それぞれ true または false の単純なステータスに反応します。

- True フィルタ：オブジェクトの結果セットが true のブール条件を持ちます。
- False フィルタ：オブジェクトの結果セットが false のブール条件を持ちます。

Property フィルタ

Property フィルタは、結果セットに含める基準としてオブジェクトのプロパティの値を使用します。ほとんどの場合、Property フィルタを作成するには、比較の値とともにターゲット オブジェクトとプロパティの classId と propertyId が必要です。

- Equality フィルタ：結果セットを、特定のプロパティが、指定したプロパティ値と「等しい」オブジェクトに限定します。
- Not equal フィルタ：結果セットを、特定のプロパティが、指定したプロパティ値と「等しくない」オブジェクトに限定します。
- Greater than フィルタ：結果セットを、特定のプロパティが、指定したプロパティ値よりも「大きい」オブジェクトに限定します。
- Greater than or Equal to フィルタ：結果セットを、特定のプロパティが指定したプロパティ値「以上」であるオブジェクトに限定します。
- Less than フィルタ：結果セットを、特定のプロパティが、指定したプロパティ値「未満」であるオブジェクトに限定します。
- Less than or Equal to フィルタ：結果セットを、特定のプロパティが、指定されたプロパティ値「以下」であるオブジェクトに限定します。
- Wildcard フィルタ：結果セットを、特定のプロパティがワイルドカードを含むプロパティと一致するオブジェクトに限定します。サポートされるワイルドカードは、「%」または「*」（任意の文字シーケンス）、「?」または「-」（任意の単一の文字）です。
- Any Bits フィルタ：結果セットを、特定のプロパティが、渡されたビット セットの少なくとも 1 つを含むオブジェクトに限定します。（ビットマスク プロパティにだけ使用します）。
- All Bits フィルタ：結果セットを、特定のプロパティが、渡されたビット セットのすべてを含むオブジェクトに限定します。（ビットマスク プロパティにだけ使用します）。

Composite フィルタ

Composite フィルタは、2 つ以上のコンポーネント フィルタから構成されます。Composite フィルタを使用すると、柔軟に結果セットを作成できます。たとえば、Composite フィルタによって、含まれているフィルタの少なくとも 1 つで受け入れられたオブジェクトだけに結果セットを限定できます。

- AND フィルタ：結果セットは、各コンポーネント フィルタのフィルタリング基準を満たす必要があります。たとえば、totalMemory が 64 メガバイトを超えた動作可能なすべてのコンピュータ ブレードを取得する場合、フィルタは 1 つの Greater than フィルタと 1 つの Equality フィルタから構成されます。
- OR フィルタ：結果セットは、少なくとも 1 つのコンポーネント フィルタのフィルタリング基準を満たす必要があります。たとえば、未割り当ての assignmentState 値または未割り当ての associationState 値を持つすべてのサービス プロファイルを取得する場合、フィルタは 2 つの Equality フィルタから構成されます。
- Between フィルタ：結果セットは、最初の指定値と 2 番めの指定値の範囲内にあるオブジェクトです。たとえば、2 つの日付間で発生したすべての障害などがあります。
- XOR フィルタ：結果セットは、たった 1 つの Composite コンポーネント フィルタのフィルタリング基準を満たすオブジェクトです。

Modifier フィルタ

Modifier フィルタは、含まれているフィルタの結果を変更します。1 つの Modifer フィルタ（NOT フィルタ）だけがサポートされます。このフィルタは、含まれているフィルタの結果を逆にします。含まれている基準に一致しないオブジェクトを取得する場合は、このフィルタを使用します。

設定メソッド

管理対象オブジェクトの設定を変更するメソッドが複数あります。これらの変更は、ツリー全体、サブツリー、または個々のオブジェクトに適用できます。これらのメソッドの例は次のとおりです。

- configConfMo：単一のサブツリー（つまり、DN）に影響を与えます。
- configConfMos：複数のサブツリー（つまり、複数の DN）に影響を与えます。
- configConfMoGroup：複数のサブツリー構造または管理対象オブジェクトに対して同じ設定の変更を加えます。

ほとんどの設定メソッドは、引数 inHierarchical を使用します。子オブジェクトが XML ドキュメントに含まれ、DME が寛容モードで動作するため、設定時にこれらの値は重大な役割を担いません。

イベント サブスクリプション メソッド

ユーザまたはシステムにより開始されたアクションによって、オブジェクトが作成、変更、または削除されると、イベントが生成されます。アプリケーションは、定期的なポーリングまたはイベント サブスクリプションにより、Cisco VNMC のステータス変更情報を取得できます。ポーリングはリソースを大量に消費するため、イベント サブスクリプションが推奨される通知方法です。

イベント サブスクリプションでは、クライアント アプリケーションが Cisco VNMC からのイベント通知を受けるよう登録できます。イベントが発生すると、Cisco VNMC はサブスクライブしているクライアント アプリケーションにイベントとそのタイプを送信します。実際の変更イベントだけが送信され、影響を受けないオブジェクトの属性は送信されません。このプロセスは、システム内のすべてのオブジェクトの変更に適用されます。

Cisco VNMC イベント通知をサブスクライブするには、HTTP セッションを開き、セッションを開いたままにします。次に、以下の例で示されているように eventSubcribe 要求を HTTP セッションを介して送信します。

```
<eventSubscribe cookie="<real_cookie>"></eventSubscribe>
```

eventSubscribe 要求が Cisco VNMC により受け取られると、HTTP セッションを介した、発生するすべての新しいイベントの送信が開始されます。イベント サブスクリプションに対して有効な Cookie を取得するには、Cisco VNMC に最初にログインする必要があります。ログインしていない場合、イベント サブスクリプション要求は拒否され、エラー応答が発生します。

各イベントは一意のイベント ID を持ちます。これらのイベント ID はカウンタとして動作し、すべてのイベント通知に含まれます。イベントが生成されると、イベント ID カウンタが増加し、新しいイベントに新しいイベント ID が割り当てられます。このプロセスにより、イベントを追跡することが可能になり、イベントを見逃すことがなくなります。クライアントがイベントを見逃した場合は、loggingSyncOcns を使用して、見逃したイベントを取得します。

（注）　Cisco VNMC 1.0 の場合は、イベント サブスクリプションに対して HTTP プロトコルだけがサポートされます。イベント サブスクリプション要求を送信する場合は HTTP を使用します。

Cisco VNMC GUI と Cisco VNMC サーバ間での XML 交換の取得

Cisco VNMC GUI は Web ベースのアプリケーションです。API が実際に使用されたことを知るために、GUI と Cisco VNMC サーバ間の XML 交換を取得します。Cisco VNMC は、Adobe FLEX GUI フレームワークを使用して開発されているため、Adobe フラッシュ プレーヤーのデバッグ バージョンをインストールしてください。インストールにより、ユーザのホーム ディレクトリ下のログ ファイルに格納されたログ出力が取得されます。Windows 7 では、ログ ファイルは C:¥Users¥<usename>¥AppData¥Roaming¥Macromedia¥Flash Player¥Logs¥flashlog.txt です。たとえば、Cisco VNMC XML API 使用例の項で指定されたほとんどの要求例および応答ペイロードはこのように取得されます。

成功または失敗の応答

Cisco VNMC は、どのような API 要求に対してもほとんど即時に応答します。要求を完了できない場合は、失敗が返されます。クエリーまたはログイン メソッドの場合は、実際の結果が返されます。設定メソッドの場合、成功の応答は要求が有効であることを示しますが、操作が完了したことを示しません。たとえば、DB バックアップ要求が受け入れられ、Cisco VNMC から成功の応答があった場合でも、Cisco VNMC サーバで実際のバックアップ ジョブを完了するのに長い時間がかかることがあります。

ここでは、次の内容について説明します。

- 「成功の応答」（P.1-11）
- 「失敗した要求」（P.1-12）
- 「空の結果」（P.1-12）

成功の応答

成功の応答時、XML ドキュメントが、要求された情報または変更が行われたことを示す確認とともに返されます。

次に、DN org-root/org-tenant d3337/pol-pl を持つポリシーに対する configResolveDn 要求の例を示します。

```
<configResolveDn cookie="<real_cookie>"
    dn="org-root/org-tenant_d3337/pol-p1"
    inHierarchical="false"/>
```

この応答には次の詳細が含まれます。

```
<configResolveDn
    dn="org-root/org-tenant_d3337/pol-p1"
    cookie="<real_cookie>"
    commCookie="7/13/0/a3c"
    srcExtSys="10.193.34.70"
    destExtSys="10.193.34.70"
    srcSvc="sam_extXMLApi"
    destSvc="policy-mgr_dme"
    response="yes">
    <outConfig>
        <policyRuleBasedPolicy
            descr=""
            dn="org-root/org-tenant_d3337/pol-p1"
            intId="10811"
```

```
            name="p1"/>
    </outConfig>
</configResolveDn></configResolveDn>
```

失敗した要求

失敗した要求に対する応答には、errorCode と errorDescr に対する XML 属性が含まれます。次に、システムにすでに存在するポリシーを作成しようとした場合の失敗した要求の例を示します。

```
<configConfMo
    dn="org-root/org-tenant_d3337/pol-p1"
    cookie="<real_cookie>"
    commCookie="7/13/0/2038"
    srcExtSys="10.193.34.70"
    destExtSys="10.193.34.70"
    srcSvc="sam_extXMLApi"
    destSvc="policy-mgr_dme"
    response="yes"
    errorCode="103"
    invocationResult="unidentified-fail"
    errorDescr="can't create; object already exists.">
 </configConfMo>
```

空の結果

存在しないオブジェクトに対するクエリー要求は、DME によって失敗として扱われません。オブジェクトが存在しない場合は、成功メッセージが返されますが、XML ドキュメントには、要求されたオブジェクトが見つからないことを示す空のデータ フィールド <outConfig> </outConfig> が含まれます。

次に、DN による存在しないポリシーの解決の例を示します。

```
<configResolveDn
    dn="org-root/org-tenant_d3337/pol-p1"
    cookie="<real_cookie>"
    commCookie="7/13/0/203e"
    srcExtSys="10.193.34.70"
    destExtSys="10.193.34.70"
    srcSvc="sam_extXMLApi"
    destSvc="policy-mgr_dme"
    response="yes">
    <outConfig>
    </outConfig>
</configResolveDn>
```