管理ツール

この章は、次の内容で構成されています。

- 管理ツール （1 ページ）
- 管理 GUI について （1 ページ）
- CLI について （2 ページ）
- ユーザ ログインのメニュー オプション （2 ページ）
- GUI および CLI バナーのカスタマイズ （3 ページ）
- REST API （3 ページ）
- エクスポート/インポートの構成 （13 ページ）
- Puppet を使用したプログラマビリティ （18 ページ）

管理ツール

Cisco Application Centric Infrastructure（ACI）ツールは、ファブリックの管理者、ネットワークエンジニア、および開発者がテナントおよびアプリケーションの開発、構成、デバッグおよび自動展開に役立ちます。

管理 GUI について

次の管理 GUI の機能により、ファブリックおよびそのコンポーネント（リーフとスパイン）にアクセスできます。

- 世界共通の Web 標準（HTML5）に基づく。インストーラまたはプラグインは必要ありません。
- モニタリング（統計、障害、イベント、監査ログ）、操作および構成データへのアクセス。
- シングルサインオン メカニズムによる APIC とスパインおよびリーフスイッチへのアクセス。
- サードパーティが使用できる同じ RESTful API を使用した APIC との通信。

CLI について

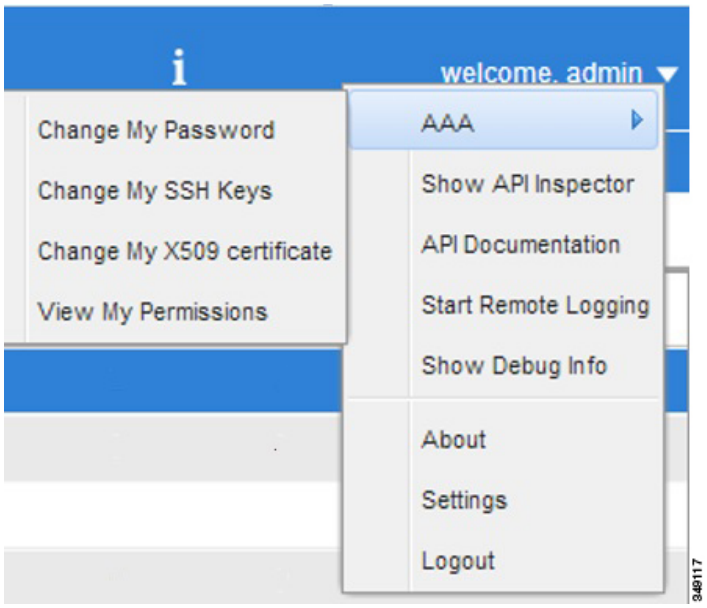
CLI は、APIC、リーフおよびスパイン スイッチへの操作インターフェイスおよび構成インターフェイスを特徴としています。

- Python で初めから実行され、Python インタープリタと CLI 間で切替えることができます。
- 拡張性のプラグイン アーキテクチャ
- 監視、操作、および構成データへの仮想ルーティングおよび転送（VRF）ベースのアクセス
- Python コマンドまたはバッチ スクリプティングによる自動化

ユーザ ログインのメニュー オプション

ユーザ ログインのドロップダウンメニューにより、複数の構成、診断、参照およびプリファレンスのオプションが提供されます。次の図は、このドロップダウンメニューを示します。

図 *1*: ユーザ ログインのメニュー オプション



オプションには次のものが含まれます。

- ユーザ パスワード、SSH キー、X509 証明書を変更、およびログインしたユーザの権限を表示するための AAA オプション。

（注） ACI ファブリックは、すべてのデバイスのシステムクロックが正しいことを保証するために、アクティブな Network Time Protocol（NTP）ポリシーを使用して構成する必要があります。そうでない場合、同期がとれていないノードで証明書が拒否される可能性があります。

- [API インスペクタの表示（Show API Inspector）] では、API インスペクタが開きます。
- [API ドキュメンテーション（API Documentation）] では、管理情報モデルの参照を開きます。
- リモート ロギング
- デバッグ情報
- ソフトウェアの現在のバージョン番号について。
- GUI を使用するためのプリファレンスの設定。
- システムを終了するためのログアウト。

GUI および CLI バナーのカスタマイズ

GUI および CLI バナーは、GUI の [管理（Admin）] > [AAA] > [セキュリティ管理（Security management）] セクションにあります。CLI バナーは、ユーザのログイン認証の前に表示されます。CLI バナーは、コンソールにそのまま出力されるテキスト ベースの文字列です。GUI バナーは、ユーザのログイン認証の前に表示されます。GUI バナーは URL です。URL は、iFrame に配置できるようにする必要があります。URL `x-frame-option` が `deny` または `sameorigin` に設定されている場合、ユーザのログイン認証の前に URL は表示されません。

REST API

REST API について

Application Policy Infrastructure Controller (APIC) REST API は、REST アーキテクチャを使用するプログラム的なインターフェイスです。API は JavaScript オブジェクトの表記（JSON）または拡張マークアップ言語（XML）のドキュメントを含む HTTP（デフォルトでは無効）または HTTPS のメッセージを受け入れ、返します。プログラミング言語を使用して、API メソッドまたは管理対象オブジェクト（MO）の説明を含むメッセージおよび JSON または XML ドキュメントを生成できます。

REST API は、管理情報ツリー（MIT）へのインターフェイスであり、オブジェクト モデルの状態を操作できます。APIC CLI、GUI、および SDK は同じ REST インターフェイスを使用す

るため、情報を表示する場合は常に、REST API を介して読み込まれ、構成変更が行われた場合は REST API を通じて書き込まれます。REST API は、統計、障害、監査イベントなど、他の情報を取得できるインターフェースも提供します。プッシュベースのイベント通知に登録する手段も提供されているので、MIT で変更が発生すると、Web ソケットを介してイベントが送信されます。

API では、HTTP を通じた POST 操作、GET 操作、DELETE 操作など、標準的な REST メソッドがサポートされています。POST メソッドと DELETE メソッドは、同じ入力パラメータで複数回呼び出されても、それ以上の効果を持たないべき等です。GET メソッドはべきゼロで、何らの変更も行うことなく（つまり、読み取り専用操作）ゼロ回または複数回呼び出すことができます。

REST インターフェイスに出入りするペイロードは、XML エンコーディングまたは JSON エンコーディングによりカプセル化できます。XML の場合、エンコーディング操作は簡単です。要素タグはパッケージとクラスの名前で、そのオブジェクトのプロパティはその要素の属性として指定します。含有は、子要素を作成して定義します。

JSON の場合、エンコーディングにはツリーベースの階層を反映する特定のエントリの定義が必要です。ただし、その定義はツリーのすべてのレベルで繰り返されるため、最初に理解していれば実装はかなり簡単です。

- すべてのオブジェクトは JSON ディクショナリとして記述され、キーはパッケージとクラスの名前です。値は、属性と子の 2 つのキーを持つ別のネストされたディクショナリです。
- 属性キーには、オブジェクト上の属性を定義するキー値ペアを記述する、さらにネストされたディクショナリが含まれています。
- 子キーには、すべての子オブジェクトを定義するリストが含まれています。このリストの子オブジェクトは、ここで説明したように定義された、ネストされたオブジェクトを含むディクショナリです。

認証

REST API のユーザー名ベースおよびパスワードベースの認証は、POST 操作の DN ターゲットとして、**aaaLogin**、**aaaLogout**、および **aaaRefresh** などの特殊なリクエストのサブセット、ユニバーサル技術情報識別子（URI）を使用します。ペイロードには、シンプルな XML ペイロードまたは JSON ペイロードが含まれ、これらには **aaaUser** オブジェクトの MO 表現と、ユーザー名とパスワードを定義する属性名および **pwd** が含まれています。たとえば、**<aaaUser name='admin' pwd='password'/>** のようになります。POST 操作の応答には、Set-Cookie ヘッダーと、応答の名前付きトークンの **aaaLogin** オブジェクトの属性の両方として認証トークンが含まれます。この場合の XPath はエンコーディングが XML の場合は **/imdata/aaaLogin/@token** です。REST API の後続の操作では、このトークン値を **APIC-cookie** という名前の cookie としてその後の要求の認証に使用できます。

サブスクリプション

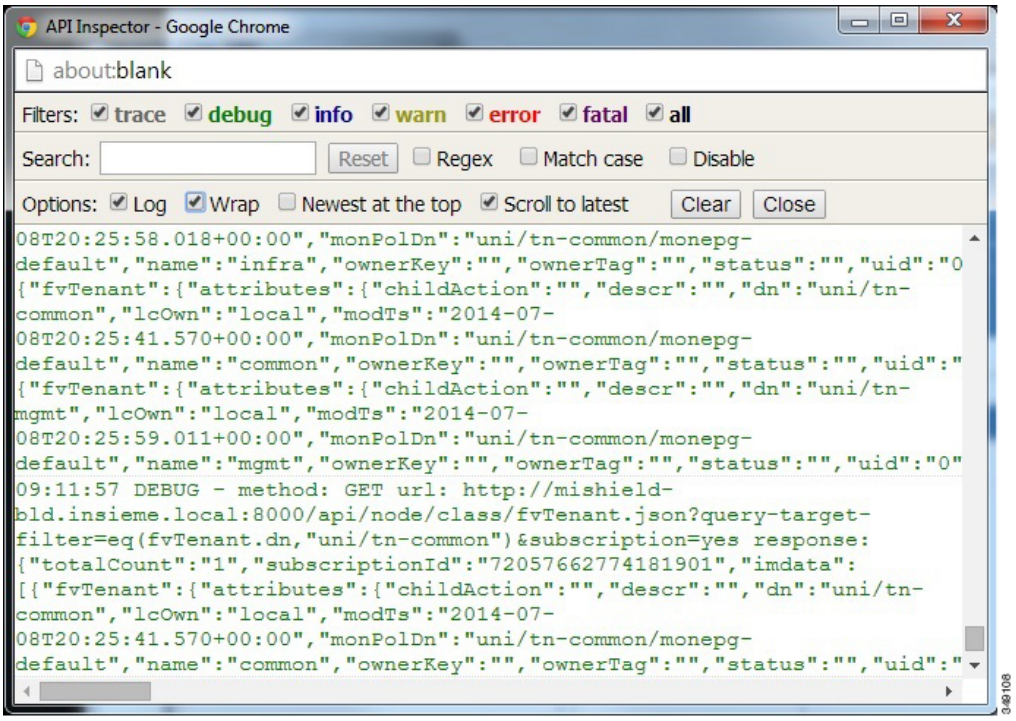
REST API は、アクティブな API セッション中の 1 つ以上の MO へのサブスクリプションをサポートします。ユーザまたはシステムにより開始されたアクションによって、MO が作成、変

更、または削除されると、イベントが生成されます。サブスクライブされたアクティブなクエリ上のデータがイベントにより変更されると、APICはそのサブスクリプションを作成したAPIクライアントに通知を送信します。

API インスペクタ

API インスペクタでは、APIC が GUI インタラクションを実行するために処理する REST API コマンドのリアルタイム表示が提供されます。下の図は、APIインスペクタがGUIの主要テナントのセクションに移動する場合に表示する REST API コマンドを示します。

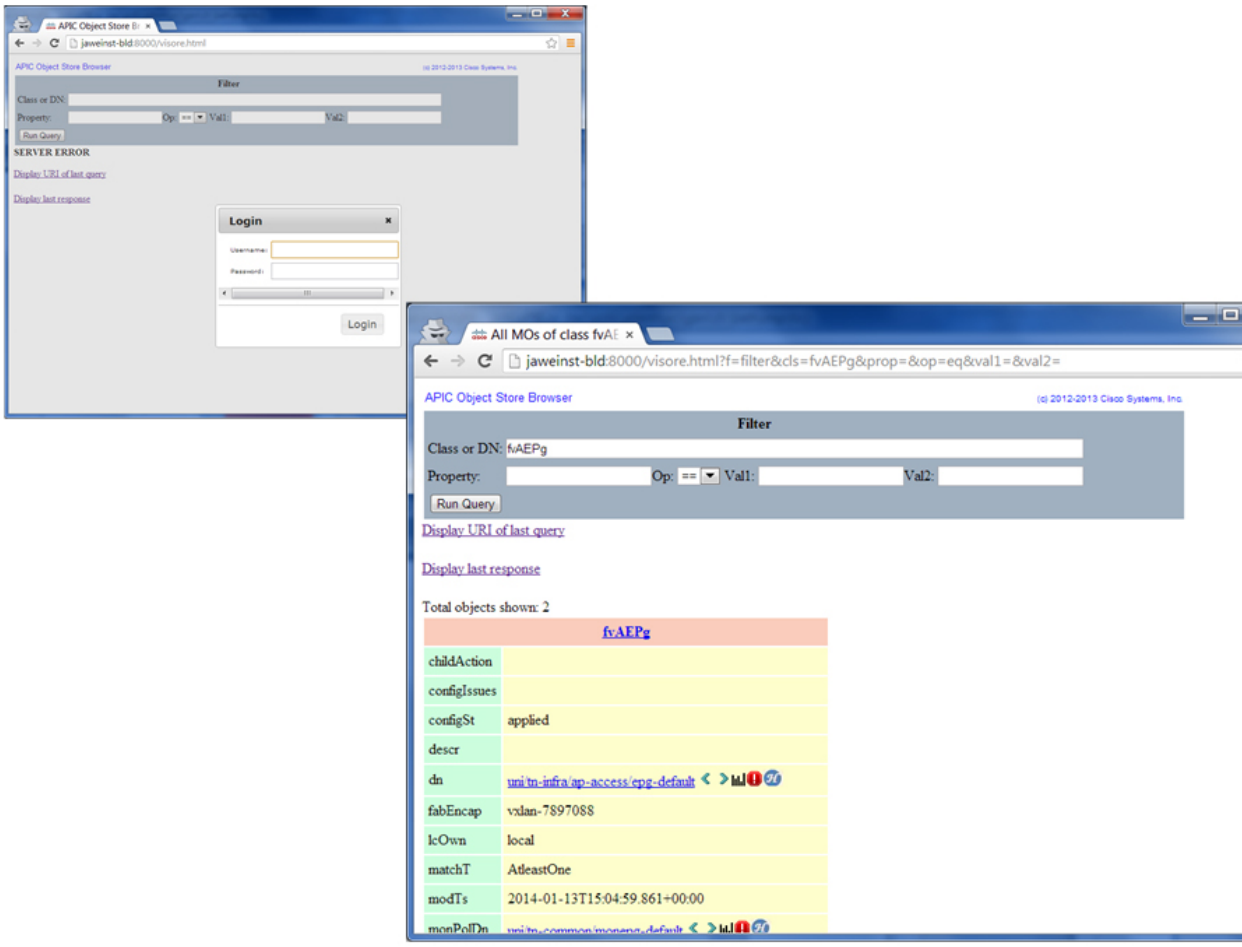
図 *2 : API* インスペクタ



Visore 管理対象オブジェクト ビューア

Visore は、下の図に示すように、読み取り専用の管理情報ツリー（MIT）ブラウザです。これにより、オプションのフィルタを使用して、識別名（DN）とクラスのクエリが可能になります。

図 *3 : Visore MO* ビューア



Visore 管理対象オブジェクト ビューアは次の場所にあります。

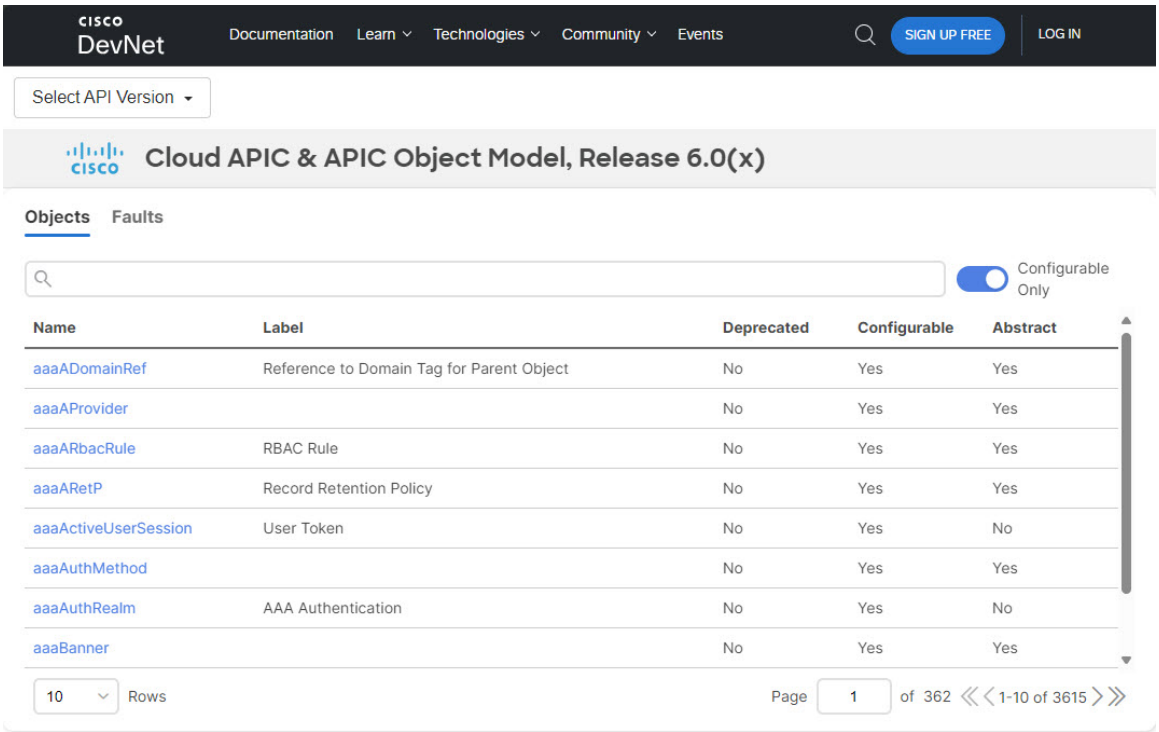
`http(s)://host[:port]/visore.html`

管理情報モデルのリファレンス

管理情報モデル（MIM）には、システム内のすべての管理対象オブジェクトとそのプロパティが含まれます。詳細については、『*Cisco APIC Management Information Model* リファレンス ガイド』を参照してください。

Cisco APIC 6.0 リリースで MIT 内のオブジェクトを検索するために管理者がどのように MIM を使用できるかに関する例については、次の図を参照してください。

図 *4 : Cisco APIC 6.0* リリースの *MIM* リファレンス

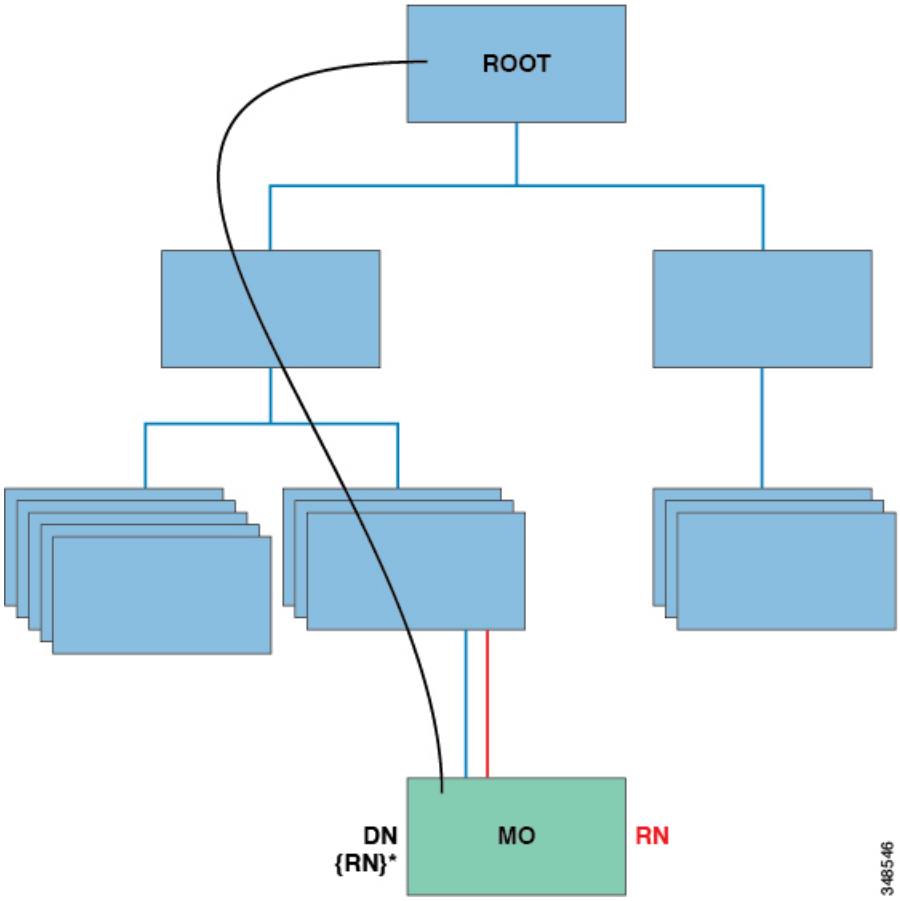


MIT 内のオブジェクトの検索

Cisco ACI は情報モデルベースのアーキテクチャ（管理情報ツリー（MIT））を使用しており、管理プロセスによって制御できるすべての情報がモデルによって説明されます。オブジェクトインスタンスは管理対象オブジェクト（MO）と呼ばれます。

次の図は、任意の MO インスタンスを一意的に表す識別名と、親 MO の下にある MO をローカル的に表す相対名を示します。MIT 内のオブジェクトはすべて、ルート オブジェクトの下に存在します。

図 *5: MO* の識別名と相対名



システム内のすべての MO は固有の識別名（DN）によって識別されます。このアプローチにより、グローバルにオブジェクトを参照できます。またオブジェクトの識別名のほか、各オブジェクトを相対名（RN）で参照することもできます。相対名は、親オブジェクトに対して相対的にオブジェクトを識別します。指定されたオブジェクトの識別名は、親オブジェクトの識別名に相対名を加えることで取得できます。

DN は、オブジェクトを一意的に識別する一連の相対名です。

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

```
dn ="sys/ch/lcslot-1/lc/leafport-1"
```

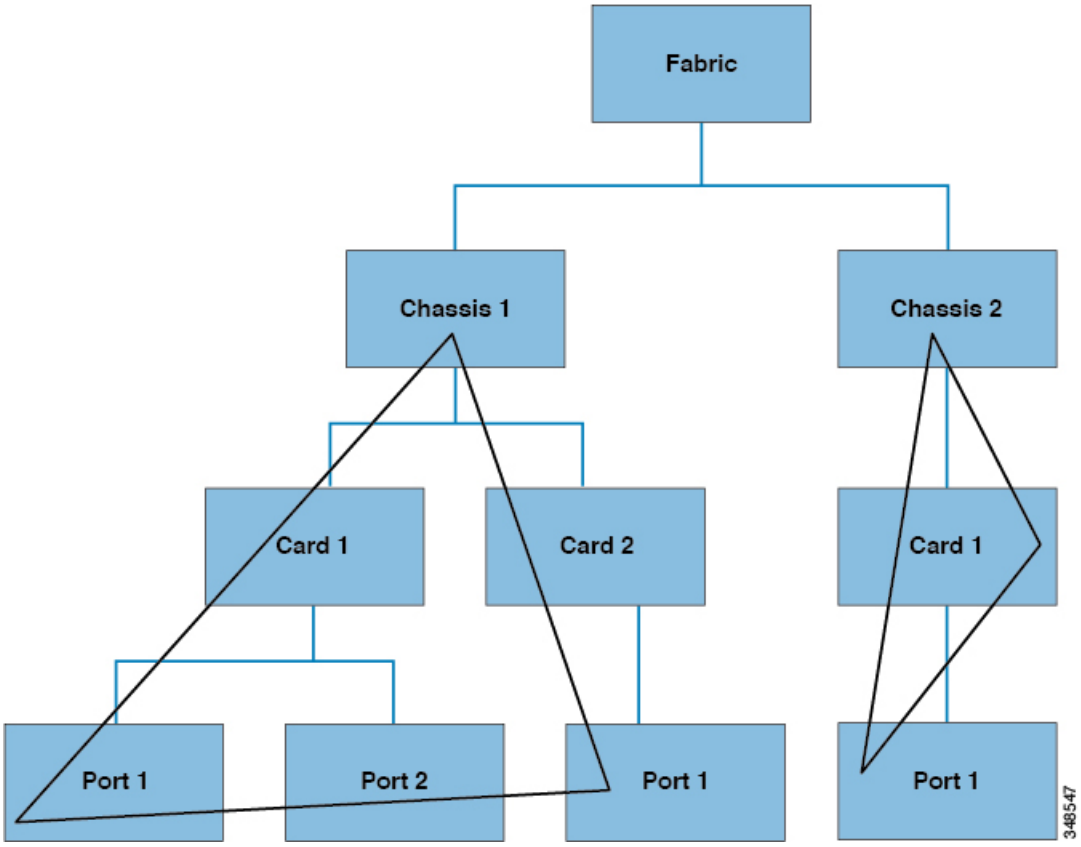
識別名はURLに直接マッピングされます。MIT内におけるオブジェクトの現在位置に応じて、相対名または識別名のいずれかを使用してオブジェクトにアクセスできます。

ツリーは階層型で構成され、属性システムを使用してオブジェクトクラスを識別できるため、さまざまな方法で管理対象オブジェクトの情報を取得するためにツリー内を照会できます。クエリは、識別名を使用してオブジェクト自体に対して実行するか、スイッチシャーシなどのオブジェクトのクラスに対して実行するか、ツリー レベルで実行してオブジェクトのすべてのメンバーを検出できます。

ツリーレベルのクエリ

次の図は、クエリ対象の 2 つのシャーシをツリー レベルで示しています。

図 *6*: ツリーレベルのクエリ

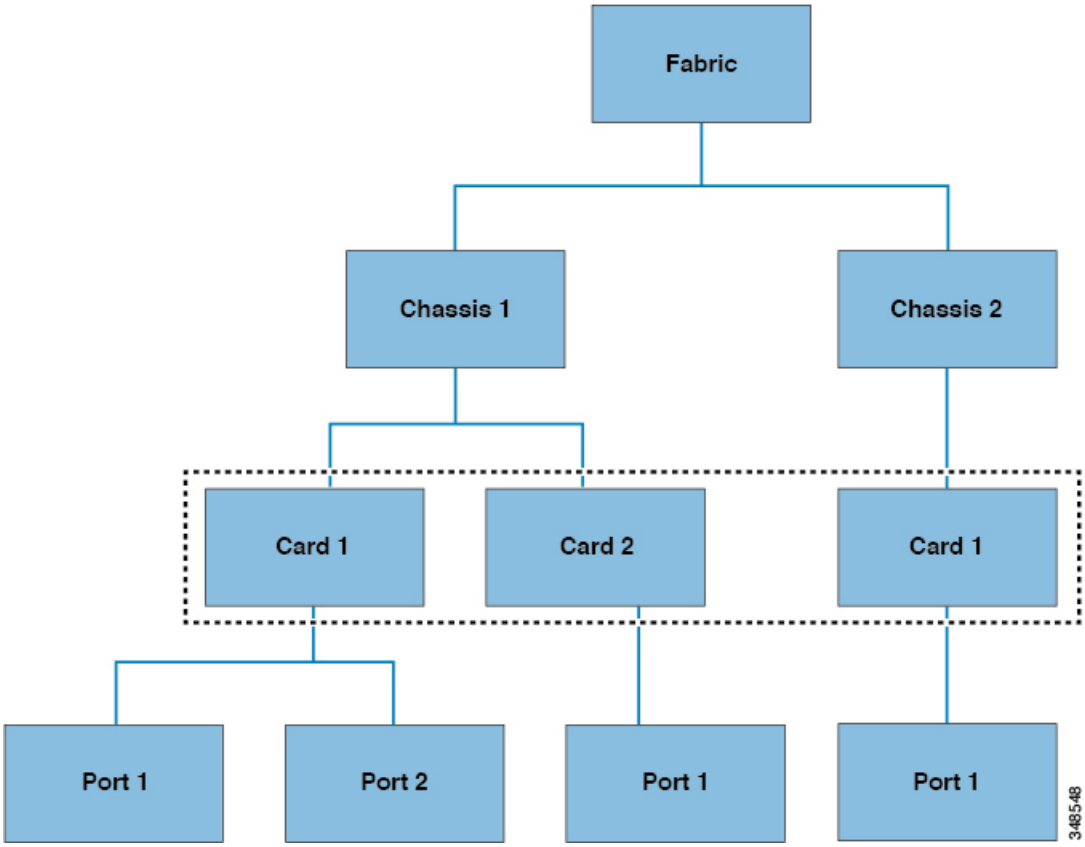


どちらのクエリも、参照されたオブジェクトと、その子オブジェクトを返します。このアプローチは、大規模なシステムのコンポーネントを検出するために役立ちます。この例では、クエリにより指定されたスイッチ シャーシのカードとポートが検出されます。

オブジェクトレベル クエリ

次の図は、2 番目のクエリ タイプ、クラスレベル クエリを示します。

図 7:オブジェクトレベル クエリ

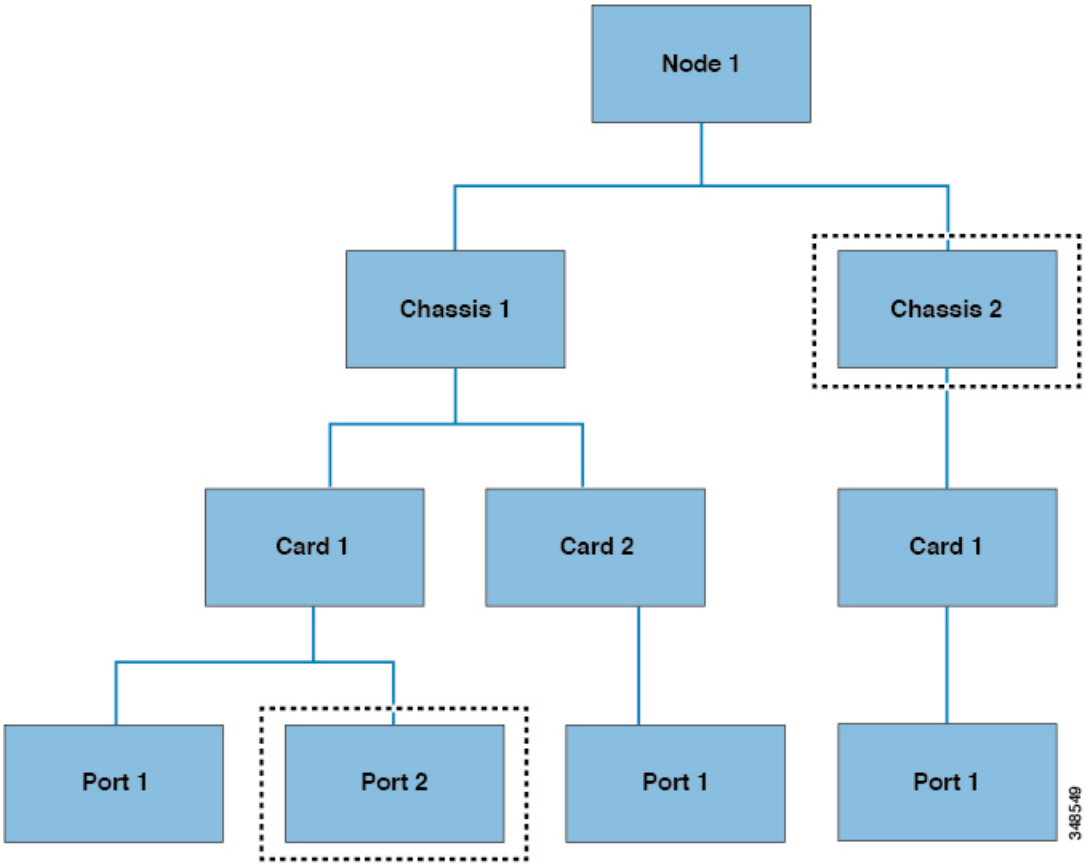


クラスレベル クエリは、任意のクラスのオブジェクトをすべて返します。このアプローチは、MIT で使用できる特定のタイプのオブジェクトをすべて検出する場合に役立ちます。この例で使用しているクラスはカードで、カード タイプのすべてのオブジェクトを返します。

オブジェクトレベル クエリ

3 つ目のクエリ タイプはオブジェクトレベル クエリです。オブジェクトレベル クエリでは、識別名を使用して特定のオブジェクトを返します。次の図は、2 つのオブジェクトレベル クエリを示しており、1 つはノード 1/シャーシ 2、もう 1 つはノード 1/シャーシ 1/カード 1/ポート 2 を照会しています。

図 *8*:オブジェクトレベル クエリ

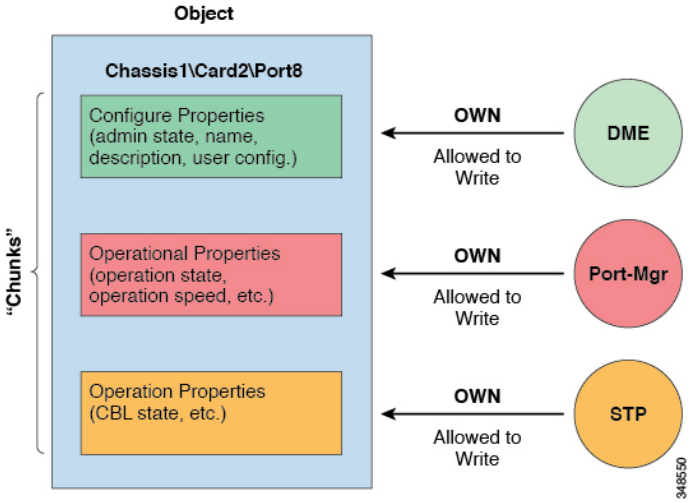


すべての MIT クエリで、管理者はサブツリー全体またはサブツリーの一部を返すよう選択できます。また、システム内のロールベースアクセスコントロール（RBAC）メカニズムによって、返されるオブジェクトが決まります。必ず、ユーザが表示権限を持つオブジェクトのみが返されます。

管理対象オブジェクトのプロパティ

Cisco ACI の管理対象オブジェクトには、管理対象オブジェクトを定義するプロパティが含まれています。管理対象オブジェクトのプロパティはチャンクに分割され、オペレーティングシステム内でプロセスによって管理されます。オブジェクトには、複数のプロセスがアクセスする場合があります。これらのプロパティはすべて実行時にまとめてコンパイルされ、単一のオブジェクトとしてユーザに提示されます。次の図は、この関係の例を示します。

図 *9:* 管理対象オブジェクトのプロパティ



オブジェクトの例には、オブジェクト内のプロパティチャンクに書き込むプロセスが3つあります。Cisco APIC（ユーザ）とオブジェクトとの間のインターフェイスとなるデータ管理エンジン（DME）、ポートの構成を処理するポート マネージャ、およびスパニングツリープロトコル（STP）のすべてが、このオブジェクトのチャンクとやり取りします。APICは、実行時にコンパイルされる単一のエンティティとしてオブジェクトをユーザに提示します。

REST インターフェイスによるオブジェクト データへのアクセス

REST は、World Wide Web などの分散型システム用ソフトウェア アーキテクチャの形式で、形式がシンプルであるため、Simple Object Access Protocol（SOAP）や Web サービス記述言語（WSDL）など、その他の設計モデルに代わって採用される機会が増えています。Cisco APIC は REST インターフェイスをサポートしており、Cisco ACI ソリューション全体へのプログラムを通じたアクセスを実現します。

Cisco ACI のオブジェクトベース情報モデルは、REST インターフェイスに非常にうまく適合しています。URL と URI は識別名に直接マッピングされ、MIT 上のオブジェクトを識別でき、MIT 上のデータを XML または JSON 形式でエンコードされた自己完結型の構造化テキスト ツリードキュメントとして記述できます。オブジェクトには、識別名とプロパティを使用して識別される親子関係があり、この関係は一連の作成、読み取り、更新、および削除（CRUD）操作によって読み取りと変更が可能です。

オブジェクトにアクセスするには、明確に定義されたアドレスである REST URL を使用します。Cisco APIC オブジェクト データを取得および操作するには標準の HTTP コマンドを使用します。使用できる URL の形式は次のとおりです。

```
<system>/api/[mo|class]/[dn|class][:method].[xml|json]?{options}
```

URL の前に指定する各構成要素は、次のとおりです。

- `system`：システム識別子、IP アドレスまたは DNS で解決可能なホスト名
- `mo | class`：これが MIT 内の MO かまたはクラスレベルのクエリかどうかの表示

- `class`：照会するオブジェクトのMOクラス（情報モデルでの指定に従う）。クラス名は、`<pkgName><ManagedObjectClassName>` で表されます。
- `dn`：照会するオブジェクトの識別名（MIT 内のオブジェクトの一意の階層名）
- `method`：オブジェクトに対して呼び出すメソッドの指定（オプション）。HTTP POST リクエストにのみ適用されます。
- `xml | json`：エンコード形式
- `options`：クエリ オプション、フィルタ、引数

RESTURLで個々のオブジェクトまたはオブジェクトクラスのアドレスを指定してアクセスできる機能により、管理者はオブジェクトツリー全体、つまりシステム全体にプログラムを通じて完全にアクセスできます。

次に、REST クエリの例を示します。

- テナント solar 下のすべての EPG と障害を検索します。

  http://192.168.10.1:7580/api/mo/uni/tn-solar.xml?query-target=subtree&target-subtree-class=fvAEPg&rsp-subtree-include=faults

- フィルタされた EPG クエリ

  http://192.168.10.1:7580/api/class/fvAEPg.xml?query-target-filter=eq(fvAEPg.fabEncap,%20"vxlan-12780288")

エクスポート/インポートの構成

すべての APIC ポリシーおよび構成データは、バックアップの作成のためにエクスポートできます。これは、エクスポートポリシーを使用して構成でき、リモートサーバーにスケジュール済みバックアップまたは即時バックアップできます。スケジュール済みバックアップは、定期バックアップ ジョブまたは繰り返しバックアップ ジョブを実行するように構成できます。デフォルトでは、すべてのポリシーおよびテナントがバックアップされますが、管理者は任意に管理情報ツリーの特定のサブツリーのみを指定できます。バックアップは、インポート ポリシーによって APIC にインポートでき、システムを以前の構成に復元できます。

データベースのシェーディング

APIC クラスタは、シャーディングと呼ばれる大規模なデータベース テクノロジを使用します。このテクノロジは、APIC によって生成および処理されるデータ セットに拡張性と信頼性を提供します。APIC 構成のデータは、データベース シャードに類似したシャードと呼ばれる論理的にバインドされたサブセットに分割されます。シャードはデータ管理の単位であり、APIC は次の方法でシャードを管理します。

- 各シャードには 3 つのレプリカがあります。
- シャードは、APIC クラスタを構成するアプライアンス全体に均等に分散されます。

1 つ以上のシャードが各 APIC アプライアンスにあります。シャード データの割り当ては事前に決定されたハッシュ関数に基づいており、静的なシャード レイアウトによってアプライアンスへのシャードの割り当てが決定されます。

設定ファイルの暗号化

リリース 1.1(2) 以降では、AES-256 暗号化を有効にすることにより APIC 設定ファイルのセキュア プロパティを暗号化できます。AES 暗号化はグローバル設定オプションです。すべてのセキュア プロパティは AES 構成設定に従っています。テナント設定などの ACI ファブリック設定のサブセットを AES 暗号化を使用してエクスポートするが、ファブリック設定の残りの部分は暗号化しないということはできません。セキュア プロパティのリストについては、*Cisco Application Centric Infrastructure Fundamentals*の「Appendix K: Secure Properties」を参照してください。

APIC は、16 ～ 32 文字のパスフレーズを使用して AES-256 キーを生成します。APIC GUI には、AES パスフレーズのハッシュが表示されます。このハッシュを使用して、2 つの ACI ファブリックで同じパスフレーズが使用されているかどうかを確認できます。このハッシュをクライアント コンピュータにコピーして、別の ACI ファブリックのパスフレーズ ハッシュと比較できます。これにより、それらのハッシュが同じパスフレーズを使用して生成されたかどうかを確認できます。ハッシュを使用して、元のパスフレーズまたは AES-256 キーを再構築することはできません。

暗号化された設定ファイルを使用する際は、次のガイドラインに従ってください。

- AES 暗号化設定オプションを使用しているファブリックに古い ACI 設定をインポートするための後方互換性がサポートされています。

（注） 逆の互換性はサポートされていません。AES 暗号化が有効になっている ACI ファブリックからエクスポートされた設定を古いバージョンの APIC ソフトウェアにインポートすることはできません。

- ファブリック バックアップ設定のエクスポートを実行するときは、必ず AES 暗号化を有効にします。これにより、ファブリックを復元するときに、設定のすべてのセキュア プロパティが正常にインポートされるようになります。

（注） AES 暗号化を有効にせずにファブリック バックアップ設定がエクスポートされると、どのセキュア プロパティもエクスポートに含まれません。そのような暗号化されていないバックアップにはセキュア プロパティは何も含まれていないため、そのようなファイルをインポートしてシステムを復元すると、ファブリックの管理者およびすべてのユーザがシステムからロックアウトされてしまう可能性があります。

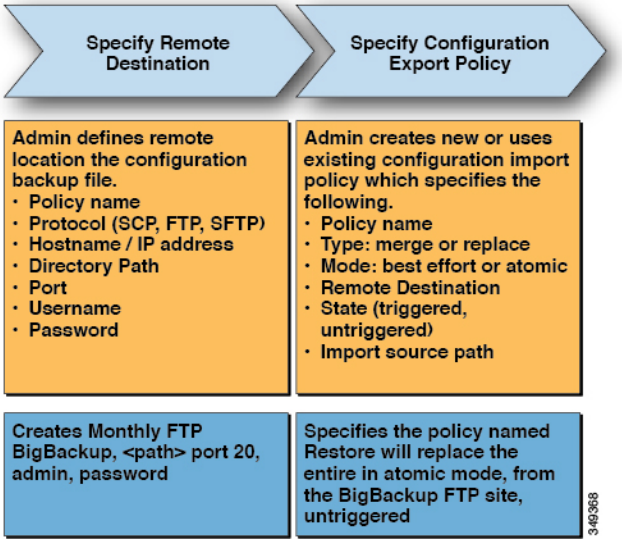
- 暗号化キーを生成する AES パスフレーズは、ACI 管理者やその他のユーザが復元したり読み取ったりすることはできません。AES パスフレーズは保存されません。APIC は AES パスフレーズを使用して AES キーを生成した後、そのパスフレーズを廃棄します。AES キーはエクスポートされません。AES キーは、エクスポートされず、REST API を使用して取得できないため、復元できません。
- 同じ AES-256 パスフレーズは、常に同じ AES-256 キーを生成します。設定のエクスポート ファイルは、同じ AES パスフレーズを使用する他の ACI ファブリックにインポートできます。
- トラブルシューティングを目的として、セキュアプロパティの暗号化データが含まれていない設定ファイルをエクスポートします。設定のエクスポートを実行する前に一時的に暗号化をオフにすると、エクスポートされた設定からすべてのセキュアプロパティ値が削除されます。すべてのセキュアプロパティが削除されたそのような設定ファイルをインポートするには、インポート マージ モードを使用します。インポート置換モードは使用しません。インポート マージ モードを使用すると、ACI ファブリック内の既存セキュア プロパティが保持されます。
- デフォルトで、APICは復号できないフィールドが含まれているファイルの設定のインポートを拒否します。この設定をオフにするときは注意してください。このデフォルト設定がオフになっているときに設定のインポートが適切に実行されないと、ファブリックのAES暗号化設定に一致しない設定ファイルのインポート時にACIファブリックのすべてのパスワードが削除される可能性があります。

（注） このガイドラインに従わないと、ファブリック管理者を含むすべてのユーザがシステムからロックアウトされる可能性があります。

設定のエクスポート

次の図は、エクスポート ポリシーを構成するプロセスがどのように動作するかを示します。

図 *10*：エクスポートポリシー構成のワークフロー



APIC は、このポリシーを次のように適用します。

- 完全なシステム構成のバックアップは月に一度実行されます。
- バックアップは BigBackup FTP サイトに XML 形式で保存されます。
- ポリシーがトリガーされます（有効になります）。

インポートの構成

管理者は、次の 2 つのモードのいずれかでインポートを実行するインポートポリシーを作成できます。

- ベストエフォート：インポートできないシャード内のオブジェクトを無視します。

（注） 受信構成のバージョンが既存のシステムと互換性がない場合、互換性のないシャードはインポートされませんが、インポート可能なシャードはインポートされません。

- アトミック：インポート可能なシャードの処理中に、インポートできないオブジェクトを含むシャードを無視します。

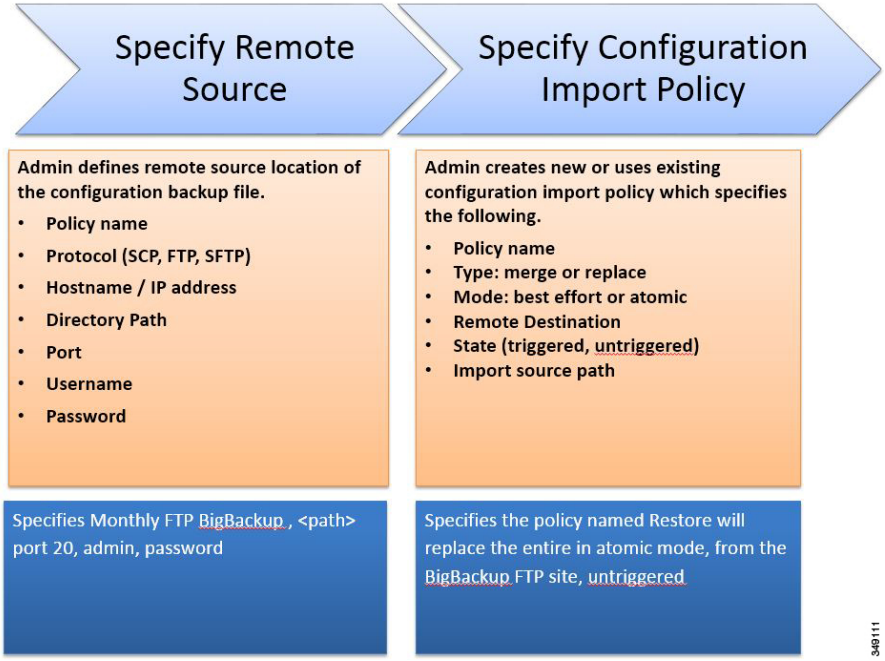
（注） 受信構成のバージョンが既存のシステムと互換性がない場合、インポートは終了します。この場合、代わりにベストエフォートを使用してください。

インポート ポリシーは、次のモードとタイプの組み合わせをサポートします。

- ベストエフォートマージ：インポートされた構成は既存の構成とマージされますが、インポートできないオブジェクトは無視されます。
- アトミックマージ：インポートされた構成は既存の構成とマージされますが、インポートできないオブジェクトを含むシャードは無視されます。
- アトミック置き換え：既存の構成をインポートされた構成データで上書きします。インポートされた構成に存在しない既存の構成のオブジェクトはすべて削除されます。オブジェクトは、既存の構成に子を持つが、インポートされた受信構成に子を持たない既存の構成から削除されます。たとえば、既存の構成に2つのテナント（solar と wind）があり、インポートされたバックアップ構成がテナントの wind が作成される前に保存されている場合、テナントの soar はバックアップから復元されますが、テナントの wind は削除されます。

次の図は、インポート ポリシーを構成するプロセスがどのように動作するかを示します。

図 *11* : インポート ポリシーを構成するワークフロー



APIC はこのポリシーを次のように適用します。

- 毎月のバックアップから完全なシステム構成の復元を実行するためのポリシーが作成されます。
- アトミック置換モードは次のことを行います。
  - 既存の構成を上書きします。
  - インポートされたファイルに存在しない既存の構成オブジェクトを削除します。

- 存在しない子オブジェクトを削除します。

- ポリシーはトリガーされません（使用できますが、アクティブ化されていません）。

テクニカル サポート、統計、コア

管理者は、APIC 内で、コア ファイルとデバッグ データを処理するために、統計情、テクニカル サポートの収集、障害およびイベントをファブリック（APIC およびスイッチ）から外部ホストにエクスポートするようエクスポートポリシーを構成できます。エクスポートは XML、JSON、Web ソケット、SCP、HTTP などのさまざまな形式にできます。エクスポートは登録可能で、定期的またはオンデマンドでストリーミングできます。

（注） 統計のエクスポートポリシーの最大数は、テナントの数とほぼ同じです。各テナントは複数の統計エクスポートポリシーを持つことができ、複数のテナントが同じエクスポートポリシーを共有できますが、ポリシーの合計数はテナントの数とほぼ同数に制限されます。

管理者は、転送プロトコル、圧縮アルゴリズム、転送の頻度などポリシーの詳細を設定できます。ポリシーは、AAA を使用して認証されたユーザによって設定できます。実際の転送のセキュリティメカニズムは、ユーザ名とパスワードに基づいています。内部的に、ポリシー要素はデータのトリガーを処理します。

Puppet を使用したプログラマビリティ

Puppet について

Puppet は Puppet Labs, Inc. の構成管理ツールです。Puppet はもともと大規模なサーバー管理用に設計されましたが、多くのデータセンターオペレーターは、同じツールを使用してサーバーとネットワークデバイスのプロビジョニングを統合したいと考えています。

次の項目は、Puppet 導入の主要なコンポーネントです。

- **Manifet**：Puppet マニフェストは、管理対象デバイス（ノード）の状態を設定するためのプロパティ定義の集合です。これらのプロパティ状態の確認および設定の詳細は抽象化されているため、マニフェストは複数のオペレーティングシステムまたはプラットフォームで使用できます。

- **Master**：通常、Puppet マスター（サーバー）は個別の専用サーバー上で実行され、複数のノードにサービスを提供します。Puppet マスターは構成マニフェストをコンパイルし、要求に応じてそれらをノードに提供します。

- **Agent または Device**：Puppet エージェントはノードで実行され、定期的に Puppet マスターに接続して構成マニフェストを要求します。エージェントは、受信したマニフェストをノードの現在の状態と調整し、相違点を解決するために必要に応じてノードの状態を更新

します。組み込みの Puppet エージェントを実行できない、または実行したくないノードの場合、Puppet は Puppet デバイスと呼ばれる構造をサポートします。Puppet デバイスは基本的に、ノードの外部にあるプロキシ メカニズムであり、ノードに代わって Puppet マスターからマニフェストを要求します。Puppetデバイスは、受信したマニフェストに必要な更新をノードに適用します。この機能を活用するには、ベンダーは、デバイスを利用する Puppet モジュールとともに、デバイス クラスのベンダー固有の実装を提供する必要があります。ベンダー固有のデバイス クラスは、独自のプロトコルまたは API を使用してリモート ノードを構成します。

Puppet の詳細とドキュメントについては、次の Puppet Web サイトを参照してください。URL：https://puppet.com/

Cisco ciscoacipuppet パペット モジュール

APIC コントローラは、組み込みの Puppet エージェントを実行しません。代わりに、シスコは Puppet モジュール（「ciscoacipuppet」）を提供します。これは、Cisco ACI 固有の Puppet デバイスを使用して、構成管理要求を APIC コントローラにリレーします。ciscoacipuppet モジュールは、受信した Puppet マニフェスト内の変更情報を解釈し、変更要求を APIC REST API メッセージに変換して、ACI ファブリックの構成変更を実装します。

ciscoacipuppet モジュールのインストール、セットアップ、および使用方法の詳細については、次の URL にある GitHub および Puppet Forge のドキュメントを参照してください。

- **GitHub** – https://github.com/cisco/cisco-network-puppet-module
- パペットフォージ – https://forge.puppet.com/puppetlabs/ciscoacipuppet

ACI に関する Puppet ガイドラインと制限事項

- ciscoacipuppet Puppet モジュールを使用してプロビジョニングできるのは、APIC 管理対象オブジェクトのサブセットのみです。サポートのレベルと制限を理解するには、GitHub および Puppet Forge の ciscoacipuppet モジュールのドキュメントを参照してください。

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。