Cisco Secure Firewall Threat Defense REST API ガイド

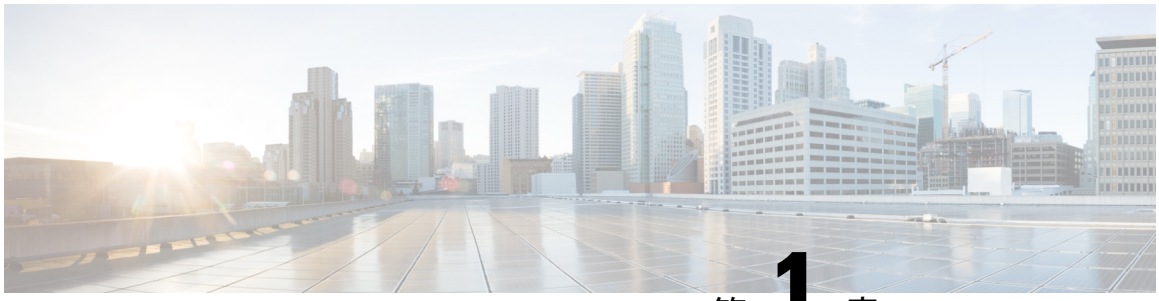
初版：2018 年 3 月 29 日

最終更新：2024 年 5 月 23 日

シスコシステムズ合同会社
〒107-6227 東京都港区赤坂9-7-1 ミッドタウン・タワー
http://www.cisco.com/jp
お問い合わせ先：シスコ コンタクトセンター
0120-092-255 （フリーコール、携帯・PHS含む）
電話受付時間：平日 10:00～12:00、13:00～17:00
http://www.cisco.com/jp/go/contactcenter/

【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。

目 次



第 1 章

Secure Firewall Threat Defense REST API について

HTTPS 経由で Secure Firewall Threat DefenseREpresentational State Transfer（REST）アプリケーションプログラミングインターフェイス（API）を使用すると、クライアントプログラムを使用して 脅威に対する防御 デバイスと通信できます。REST API は、JavaScript Object Notation（JSON）形式を使用してオブジェクトを表します。

Secure Firewall Device Managerには、プログラムで使用可能なすべてのリソースおよびJSON オブジェクトを説明するAPIエクスプローラが含まれます。エクスプローラは各オブジェクトの属性と値のペアについて詳細情報を提供するため、さまざまな HTTP メソッドを試して各リソースに必要なコーディングを理解することができます。APIエクスプローラでは、各リソースに必要な URL の例も示します。

https://developer.cisco.com/site/ftd-api-reference/ では、参照情報と例をオンラインで検索することもできます。

API には独自のバージョン番号があります。API の 1 つのバージョン用に設計されたクライアントが将来のバージョンでエラーなく動作したり、プログラムへの変更が不要である保証はありません。

このプログラミング ガイドの対象読者

このガイドは、プログラミングの一般的な知識と、REST API および JSON の一定の理解があることを前提に書かれています。これらのテクノロジーになじみがない場合は、最初に REST API の一般的なガイドをお読みください。

サポートされる HTTP メソッド

次の HTTP メソッドのみを使用できます。他のメソッドはサポートされません。

- GET：システムからデータを読み取ります。
- POST：新しいオブジェクトを作成します。
- PUT：既存のオブジェクトを変更します。PUT を使用する場合は、JSON オブジェクト全体を含める必要があります。オブジェクト内の個々の属性を選択的に更新することはできません。
- DELETE：ユーザ定義オブジェクトを削除します。

API のベース URL

指定した 脅威に対する防御 デバイスのベース URL を決定する最も簡単な方法は、API エクスプローラで GET メソッドを試し、結果から URL のオブジェクト部分を削除することです。

たとえば、GET /object/networks を実行して、返される出力の要求 URL の下に、次に示すようなものを見ることができます。

```
https://ftd.example.com/api/fdm/v1/object/networks
```

URL のサーバー名の部分は、脅威に対する防御デバイスのホスト名または IP アドレスで、「ftd.example.com」の部分が使用デバイスによって異なります。この例では、パスから /object/networks を削除してベース URL を取得します。

```
https://ftd.example.com/api/fdm/v1/
```

リソースのすべての呼び出しは、要求 URL のベースとしてこの URL を使用します。

HTTPS データポートを変更した場合は、URL にカスタムポートを含める必要があります。たとえば、ポートを 4443 に変更した場合は、https://ftd.example.com:4443/api/fdm/v1/ のような URL にします。

URL の要素「v」は API バージョンです。通常、これはソフトウェアバージョンに応じて変化します。たとえば、脅威に対する防御 バージョン 6.3.0 の API バージョンは v2 です。そのため、ベース URL は次のようになります。

```
https://ftd.example.com/api/fdm/v2/
```

（注） 脅威に対する防御 6.4 以降では、パス内の v 要素の代わりに **latest** を使用することで、API コール内のパスを更新する必要性を省くことができます。たとえば、https://ftd.example.com/api/fdm/latest/ などとします。**latest** エイリアスは、デバイスでサポートされている最新の API バージョンに解決されます。

APIエクスプローラで、ページの一番下にスクロールすると、ベースURL（サーバ名を除く）および API のバージョンに関する情報を見ることができます。

REST API の SSL/TLS 通信の保護

Threat Defense デバイスには自己署名証明書が付属しているため、デバイスとの HTTPS 通信を開始できます。ただし、証明書は既知の認証局（CA）によって署名されていないため、SSL/TLS によるアクセス試行はすべて、接続が安全でないと見なされます。

ブラウザに接続すると、自己署名証明書を受け入れるように求められますが、curl などのコマンドでは証明書が拒否されます。Curl の場合は、**--insecure** キーワードを追加することによって、証明書チェックの失敗をバイパスできます。次に例を示します。

```
curl --insecure -X GET --header 'Accept: application/json'
'https://ftd.example.com/api/versions'
```

最初に行う必要がある操作の 1 つは、脅威に対する防御 デバイス用の CA 署名付きデバイス証明書の取得です。次に、Device Manager または API を使用して、この証明書を管理証明書として割り当てます。それ以降、SSL/TLS 証明書のチェックは失敗せず、API コールで安全でない通信を使用する必要はなくなります。

手順

ステップ 1 **POST /object/internalcertificates** リソースを使用して、CA 署名付きデバイス証明書をアップロードします。

ステップ 2 **PUT /devicesettings/default/webuicertificates/{objId}** リソースを使用して、この証明書を管理証明書にします。

GET /devicesettings/default/webuicertificates リソースを使用して、Web UI 証明書のオブジェクト ID を調べます。

ステップ 3 **POST /operational/deploy** リソースを使用して、変更を展開します。

サポートされる API バージョンの判別

GET /api/versions（ApiVersions）メソッドを使用して、デバイスでサポートされる API バージョンを判別できます。このメソッドでは認証は不要で、パスに version 要素を含める必要もありません。次に例を示します。

```
curl -X GET --header 'Accept: application/json' 'https://ftd.example.com/api/versions'
```

「ftd.example.com」を Threat Defense デバイスのホスト名または IP アドレスに置き換えます。

このメソッドは、使用可能な API バージョンのリストを返します。次に例を示します。

```
{
    "supportedVersions":["v3", "latest"]
}
```

バージョン文字列は、以降の API コールの URL で使用するものと同じです。特定のバージョン識別子の代わりに **latest** を使用した場合は、以降のリリースのコールを更新する必要性を省くことができます。ただし、この手法を使用しても、コールで使用するオブジェクトモデルに対する変更は解決されず、場合によってはリリースごとに調整が必要になります。

通常、次の手順は、OAuth を使用した REST API クライアントの認証 （13 ページ） で説明されているようにアクセス トークンを取得することです。

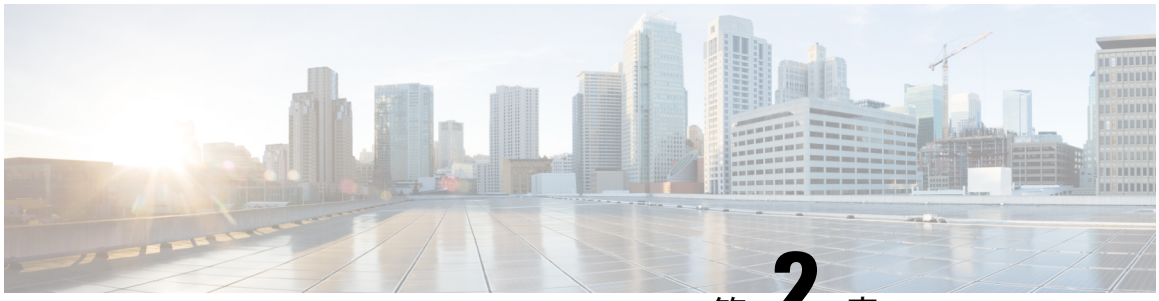
API バージョンの後方互換性

Threat Defense API のバージョンは、Threat Defense ソフトウェアのメジャーリリースごとに変更されます。新しい機能は、追加または変更された機能の API コールに影響します。

とはいえ、リリースごとに多くの機能が変更されることはありません。たとえば、ネットワークおよびポートオブジェクトに関連した API は、新しいリリースでは変更されないことがよくあります。

Threat Defense バージョン 6.7 以降、ある機能の API リソースモデルがリリース間で変更されない場合、Threat Defense API は古い API バージョンに基づくコールを受け入れることができます。機能モデルが変更された場合でも、古いモデルを新しいモデルに変換する論理的な方法があれば、古いコールが機能します。たとえば、v5 コールを v6 システムで受け入れることができます。コールのバージョン番号として「latest（最新）」を使用する場合、「古い」コールは、このシナリオでは v6 コールとして解釈されるため、下位互換性を利用するかどうかは、API コールの構築方法によって決まります。

後方互換性がサポートされない方法で API バージョン間で機能モデルが変更された場合、エラーメッセージが表示されます。これらのエラーを確認し、特定のコールのコードを更新する必要があります。



第 2 章

API エクスプローラ

REST API について学習するには、API エクスプローラを使用します。さまざまなメソッドやそれらが正しく構成されていることを確認するためのリソースをテストすることもできます。出発点として JSON モデルをコピーし、コードに貼り付けることができます。

ヒント　API エクスプローラの目的は、API についての学習を手助けすることです。API エクスプローラ経由の呼び出しのテストでは、通常の操作を妨げる可能性があるアクセスロックの作成が必要です。非実稼働デバイスで API エクスプローラを使用することをお勧めします。

API エクスプローラを開く

API エクスプローラでは、プログラムで使用可能なすべてのリソースおよび JSON オブジェクトが説明されます。エクスプローラは各オブジェクトの属性と値のペアについて詳細情報を提供するため、さまざまなHTTPメソッドを試して各リソースに必要なコーディングを理解することができます。

手順

ステップ 1　ブラウザを使用して、システムのホームページ（https://ftd.example.com など）を開きます。

ステップ 2　、Device Manager にログインします。

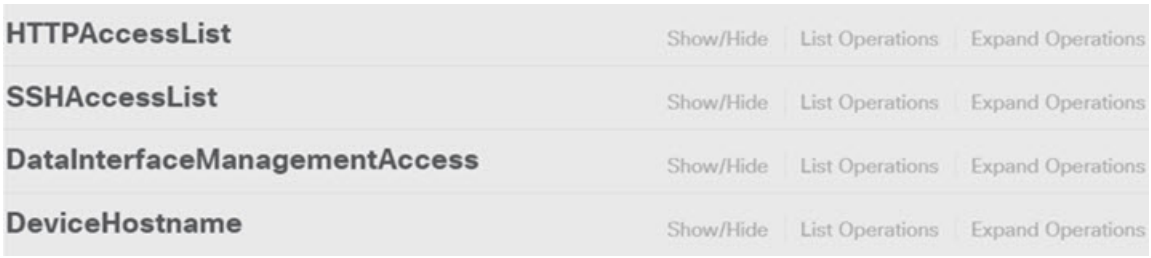
ステップ 3　（6.4 以前）。/#/Api-explorer を指すように URL を編集します（たとえば、https://ftd.example.com/#/api-explorer）。

ステップ 4 （6.5 以降）。[詳細オプション（More options）] ボタン（⋮）をクリックし、[APIエクスプローラ（API Explorer）] を選択します。

ブラウザの設定に応じて、API エクスプローラが別のタブまたはウィンドウで開きます。

API エクスプローラでの方法の検索

API エクスプローラを開始すると、リソース グループの一覧が表示されます。これらのグループには、API で利用可能なリソースが含まれます。次の図は、小さなリストの例を示しています。

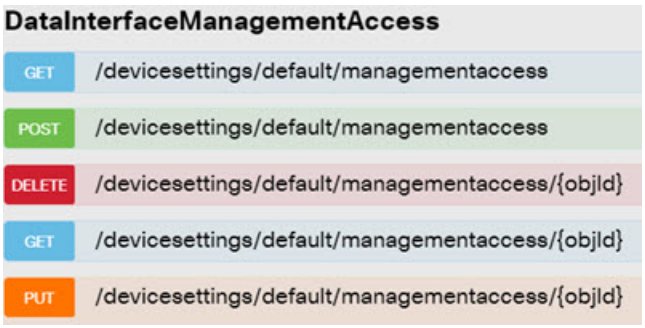


これらのグループ名はリンクです。リンクをクリックしてグループを開くと、グループのリソースで使用できるメソッドを確認できます。各グループの右側には次のコマンドも含まれています。

- [表示/非表示（Show/Hide）] ではグループが開閉します。これは、グループ名をクリックするのと同じです。展開すると最初は単にメソッドを示します（[操作の一覧表示（List Operations）] と同じ）が、システムは（閉じる前の）最後の展開状態を記憶するため、同じ展開レベルで再度開きます。
- [操作の一覧表示（List Operations）] は、グループの各リソースで利用可能な HTTP メソッドを示します。情報には、リソースごとのユニバーサル リソース ロケータ（URL）テンプレートの相対パスが含まれます。パス変数は、標準の規則によって示されます：{*variable*}。カッコを含む {*variable*} は適切な値に置き換える必要があります。この相対パスにベース URL を追加する必要があります。API のベース URL （2 ページ）を参照してください。

  そのメソッドの完全なドキュメントを参照するには、操作の URL テンプレートをクリックします。
- [操作の展開（Expand Operations）] では、グループ内の利用可能なすべての HTTP メソッドおよびリソースが開きます。

一部のグループには、多数の子リソースがあります。たとえば、DataInterface ManagementAccess グループには、/devicesettings/default/managementaccess に対する GET、POST、および DELETE 操作、/devicesettings/default/managementaccess/{objId} に対する GET および PUT 操作が含まれます。



リソースに関するドキュメントの表示

各リソースの属性は API エクスプローラに記載されています。

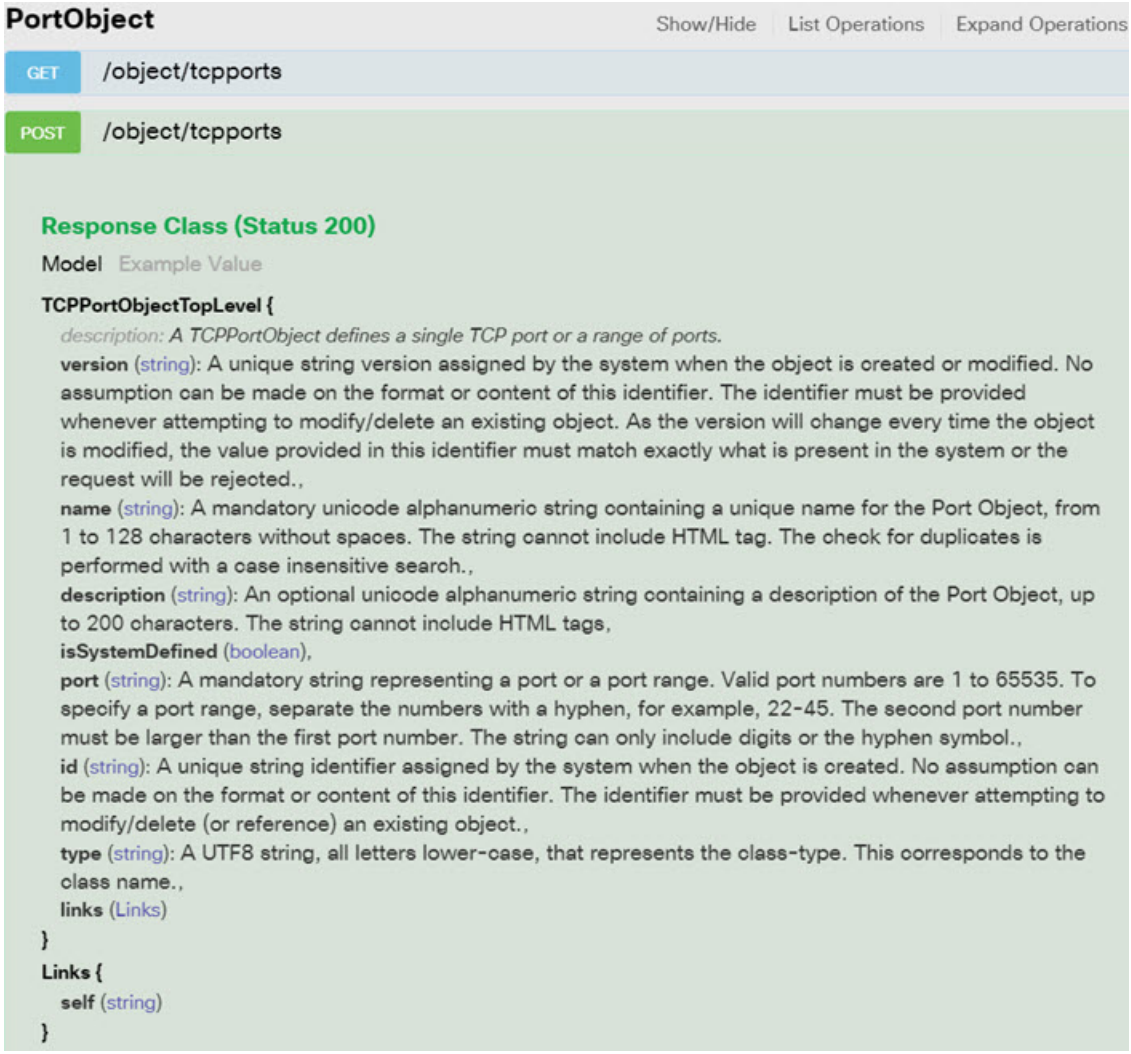
手順

ステップ 1 関心のある特定のリソースおよびメソッドまでドリル ダウンします。

ステップ 2 [応答クラス（Response Class）] セクションで、[モデル（Model）] タブをクリックします。

モデルには、属性が説明およびデータ型とともにリストされます。GET の場合、ページング オプションも返される可能性があります。応答で返されたより多くのオブジェクトが存在する場合は、次および前のオブジェクトの集合への URL が取得されます。

たとえば、次のグラフィックは POST /object/tcpports のメソッドおよびリソースを示しており、[モデル（Model）] タブが選択されています。デフォルトでは [サンプル値（Example Value）] タブが選択されているため、ドキュメンテーションを表示するためには常に [モデル（Model）] をクリックする必要があります。



オブジェクト ID（objId）と親 ID の検索

一部のリソースでは、URL に次のようなオブジェクト ID または関連する親オブジェクト ID が必要です。

- PUT /object/networks/{objId}
- GET /policy/intrusionpolicies/{parentId}/intrusionrules

ほとんどの場合、リソース階層の 1 レベル上で GET メソッドを使用すると、オブジェクトまたは親 ID を入手できます。オブジェクト/親 ID は、与えられたオブジェクトの **id** パラメータの UUID です。

たとえば、GET /object/networks は、現在定義されているすべてのネットワーク オブジェクトのリストを返します。場合によっては、目的のオブジェクトに到達するまでリスト中のページに複数のコールを実行したり、**limit** クエリパラメータを含めてコールで返されるオブジェクト数を増加させたりする必要があります。各オブジェクトは次の形式であり、オブジェクト ID は強調表示しています。

```
{
      "version": "9bbb9e5d-8115-11e7-8cb4-772d7eb1894d",
      "name": "any-ipv4",
      "description": null,
      "subType": "NETWORK",
      "value": "0.0.0.0/0",
      "isSystemDefined": true,
      "id": "9bbbc56e-8115-11e7-8cb4-01865c95f930",
      "type": "networkobject",
      "links": {
        "self": "https://ftd.example.com/api/fdm/[最新（latest）]/object/networks/
9bbbc56e-8115-11e7-8cb4-01865c95f930"
      }
```

（注） いくつかのケースでは、{objId} は階層のトップレベルで発生します。これらのケースでは、オブジェクトIDに任意の値を入力し、同じ結果が得られることがあります。他のケースでは、有効なオブジェクトのタイプに関する情報をオブジェクトモデル ドキュメントで確認します。ID は有効なタイプの 1 つです。これらは常に GET コールです。たとえば、GET /operational/systeminfo/{objId} や GET /operational/featureinfo/{objId} です。

エラーカタログの表示とエラーメッセージの評価

REST API として、存在しないオブジェクトに対して GET を実行した場合は、404 などの標準の HTTP エラーコードが返されます。

また、エラーをより具体的に説明するエラーメッセージがいくつかシステムに組み込まれています。API コールでエラーが発生した場合は、応答本文にこれらの具体的なメッセージが含まれる可能性があります。

たとえば、次のネットワークオブジェクトについて **POST /object/networks** を実行しようとすると、エラーが発生します。この場合、ネットワークを指定しようとしていますが、ネットマスクを含めることを忘れています（つまり、値を 10.10.10.0/24 または 10.10.10.0/255.255.255.0 のいずれかにする必要があります）。

```
{
  "name": "test-network",
  "subType": "NETWORK",
  "value": "10.10.10.0",
  "type": "networkobject"
}
```

結果として、422 の HTTP 応答コードと、特定のエラーメッセージを含む応答本文が表示されます。

```
{
  "error": {
    "severity": "ERROR",
    "key": "Validation",
    "messages": [
      {
        "description": "The type Network requires a netmask. To specify a single host,
either use the type Host, or use {0}/255.255.255.255.",
        "code": "networkWithoutNetmask",
        "location": "value"
      }
    ]
  }
}
```

次の手順では、応答本文で返される可能性のあるエラーメッセージのリストを表示する方法について説明します。

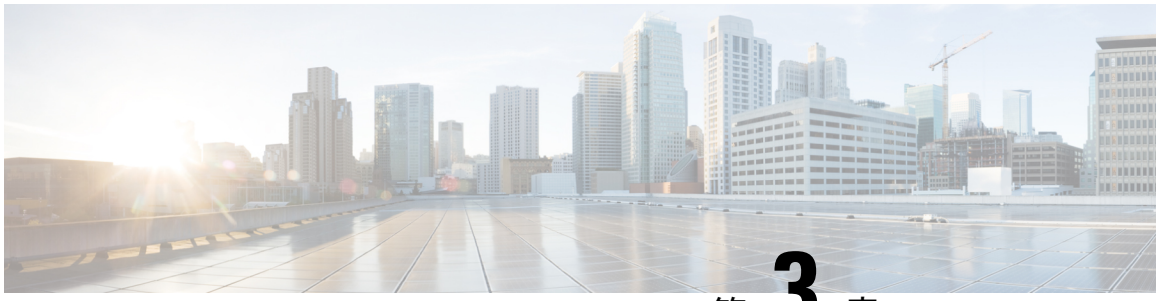
手順

ステップ 1 Device Manager で [詳細オプション（More options）] ボタン（⋮）をクリックし、**[APIエクスプローラ（API Explorer）]** を選択します。

ステップ 2 目次の **[エラーカタログ（Error Catalog）]** をクリックします。

メッセージには次のコンポーネントがあります。

- **[カテゴリ（Category）]**：メッセージの一般タイプ。この値は、エラー応答本文の **key** 属性に表示されます。カテゴリには、検証（Validation）、一般（General）、展開（Deployment）などがあります。
- **[コード（Code）]**：エラーメッセージを識別する固有の文字列。この値は、エラー応答本文の **code** 属性に表示されます。ブラウザの [ページで検索（Find On Page）] 機能を使用して、この値を使用しているカタログ内のメッセージを検索できます。
- **[メッセージ（Message）]**：エラーを説明する具体的なメッセージ。この値は、エラー応答本文の **description** 属性に表示されます。メッセージ内の変数は、{0}、{1} などと示されます。



第 3 章

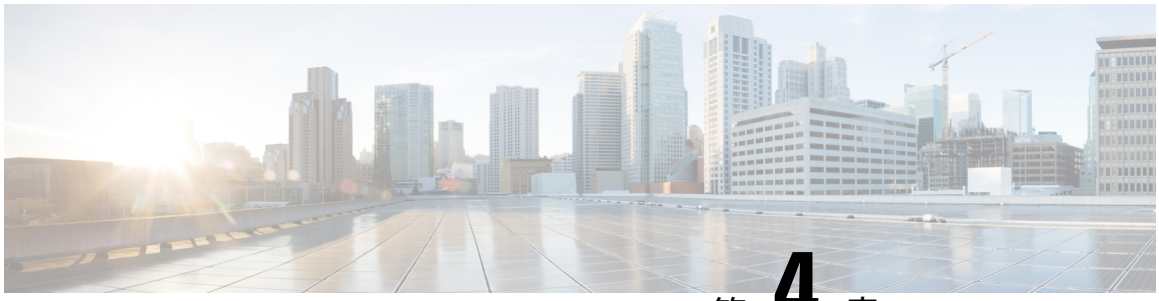
REST API を使用するための一般的なプロセス

REST API を使用するための一般的なプロセス

一般的に、クライアントは次の反復的なプロセスを使用して 脅威に対する防御 デバイスと通信します。

1. アクセス トークンを取得して API 呼び出しを認証します。API クライアント認証プロセスの概要 （13 ページ）を参照してください。
2. 単にデータを読み取る場合を除き、JSON ペイロードをビルドします。
3. リソースのユニバーサル リソース ロケータ（URL）に対する HTTPS 呼び出しを使用して JSON ペイロードを送信します。
4. 返された JSON 応答を使用します。
5. 設定変更を行う場合は、変更を展開します。設定変更の導入 （51 ページ） を参照してください。

REST API を使用するための一般的なプロセス



第 4 章

OAuth を使用した REST API クライアントの認証

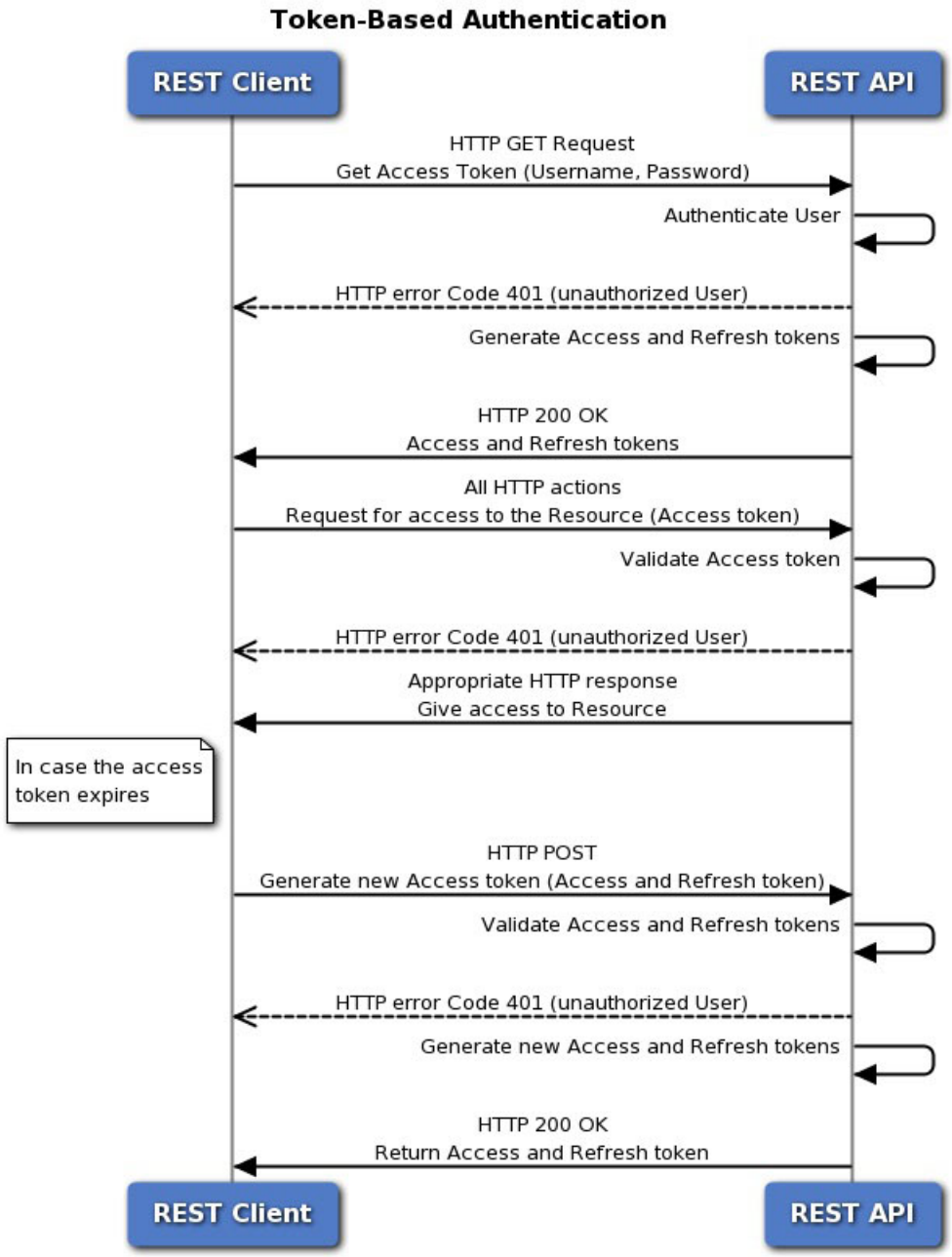
脅威に対する防御 REST API は、Oauth 2.0 を使用して API クライアントからのコールを認証します。OAuth はアクセストークンベースの方法であり、脅威に対する防御 はスキーマに JSON Web トークンを使用します。関連する規格は次のとおりです。

- RFC6749、OAuth 2.0 認証フレームワーク、https://tools.ietf.org/html/rfc6749。
- RFC7519、JSON Web トークン（JWT）、https://tools.ietf.org/html/rfc7519。

ここでは、必要なトークンを取得して使用する方法について説明します。

API クライアント認証プロセスの概要

脅威に対する防御 デバイスを使用して API クライアントを認証する方法のエンドツーエンドビューを以下に示します。



始める前に

各トークンは、HTTPS ログインセッションを表します。これにより、API セッションと Device Manager セッションがカウントされます。最大 5 つのアクティブな HTTPS セッションが可能です。この制限を超えると、最も古いセッション（Device Manager ログインまたは API トークン）が期限切れになり、新しいセッションが許可されます。したがって、必要なトークンのみを取得し、期限切れになるまで各トークンを再利用してから、それらを更新することが重要です。API コールごとに新しいトークンを取得すると、深刻なセッションチャーンが発生し、

ユーザーが Device Manager からロックアウトされる可能性があります。これらの制限は、SSH セッションには適用されません。

手順

ステップ 1 必要な任意の方法を使用して API クライアント ユーザを認証します。

クライアントにはユーザーを認証する責任があるため、クライアントが 脅威に対する防御 デバイスにアクセスして変更する権限を持っていることを確認します。認証権限に基づいた差別化機能を提供する場合は、それをクライアントに構築する必要があります。

たとえば、読み取り専用アクセスを許可する場合は、必要な認証サーバ、ユーザアカウントなどを設定する必要があります。その後、読み取り専用権限を持つユーザがクライアントにログインすると、GET コールのみを発行するようにする必要があります。API v1 では、このタイプの変数アクセスは 脅威に対する防御 デバイス自体では制御できません。API v2以降では、外部ユーザーを使用していて、ユーザー認証に基づいてコールを調整していない場合、ユーザー認証と試行したコール間に不一致があるとエラーが表示されます。

v1 の場合、デバイスと通信するときは、[管理者（admin）] ユーザーアカウントを 脅威に対する防御 デバイスで使用する必要があります。[管理者（admin）] アカウントは、ユーザ設定可能なすべてのオブジェクトに対する完全な読み取り/書き込み権限を持っています。

ステップ 2 [管理者（admin）] アカウントを使用して、ユーザ名/パスワードに基づくパスワードが付与されたアクセス トークンを要求します。

パスワード付与アクセス トークンの要求 （16 ページ）を参照してください。

ステップ 3 必要に応じて、クライアントのカスタム アクセス トークンを要求します。

カスタム トークンを使用すると、有効期間を明示的に要求し、トークンにサブジェクト名を割り当てることができます。カスタム アクセス トークンの要求 （17 ページ）を参照してください。

ステップ 4 [認証：ベアラー（Authorization: Bearer）] ヘッダーで API コールのアクセス トークンを使用します。

API コールでのアクセス トークンの使用 （19 ページ）を参照してください。

ステップ 5 アクセス トークンの有効期限が切れる前にトークンを更新します。

アクセス トークンの更新 （20 ページ）を参照してください。

ステップ 6 完了したら、トークンの有効期限が切れていない場合は無効にします。

アクセス トークンの無効化 （22 ページ） を参照してください。

パスワード付与アクセス トークンの要求

発信者が要求されたアクションを実行する権限を持っていることを確認するための認証トークンが、すべての REST API コールに含まれている必要があります。最初に、[管理者（admin）] のユーザ名およびパスワードを指定してアクセス トークンを取得する必要があります。これはパスワード付与アクセス トークン（grant_type = password）と呼ばれます。

手順

ステップ 1 パスワード付与アクセス トークン付与のための JSON オブジェクトを作成します。

```
{
  "grant_type": "password",
  "username": "string",
  "password": "string"
}
```

[管理者（admin）] のユーザ名および正しいパスワードを指定します。たとえば、次のようになります。

```
{
  "grant_type": "password",
  "username": "admin",
  "password": "Admin123"
}
```

ステップ 2 POST /fdm/token を使用して、アクセストークンを取得します。

たとえば、**curl** コマンドは次のようになります。

```
curl -X POST --header 'Content-Type: application/json' --header
'Accept: application/json' -d '{
   "grant_type": "password",
   "username": "admin",
   "password": "Admin123"
 }' 'https://ftd.example.com/api/fdm/[最新（latest）]/fdm/token'
```

ステップ 3 応答からアクセス トークンと更新トークンを取得します。

正常な応答（ステータス コード 200）は次のようになります。

```
{
  "access_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1MDI4MzI2NjcsInN
1YiI6ImFkbWluIiwianRpIjoiMGM3ZDBmNDgtODIwMS0xMWU3LWE4MWMtMDcwZmYzOW
U3ZjQ0IiwibmJmIjoxNTAyODMyNjY3LCJleHAiOjE1MDI4MzQ0NjcsInJlZnJlc2hU
b2tlbkV4cGlyZXNBdCI6MTUwMjgzNTA2NzQxOSwidG9rZW5UeXBlIjoiSldUX0FjY2
VzcyIsIm9yaWdpbiI6InBhc3N3b3JkIn0.b2hI6fVA_GbmhCOPM-ZUx6IC8SgCk1Ak
HXI-llV0r7s",
  "expires_in": 1800,
  "token_type": "Bearer",
```

```
  "refresh_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1MDI4MzI2NjcsI
nN1YiI6ImFkbWluIiwia nRpIjoiMGM3ZDBmNDgtODIwMS0xMWU3LWE4MWMtMDcwZmY
zOWU3ZjQ0IiwibmJmIjoxNTAyODMyNjY3LCJleHAiOjE1MDI4MzUwNjcsImFjY2Vzc1
Rva2VuRXhwaXJlc0F0IjoxNTAyODM0NDY3NDE5LCJyZWZyZXNoQ291bnQiOi0xLCJ0b2
tlblR5cGUiOiJKV1RfUmVmcmVzaCIsIm9yaWdpbiI6InBhc3N3b3JkIn0.iLNqz1c1Xl
vcq0j9pQYW4gwYsvUCcSyaiDRXGutAz_o",
  "refresh_expires_in": 2400
}
```

説明：

- **access_token** は、API コールに含める必要があるベアラートークンです。API コールでのアクセス トークンの使用 （19 ページ） を参照してください。
- **expires_in** は、アクセストークンが有効である、トークン発行時からの秒数です。
- **refresh_token** は、更新要求で使用するトークンです。アクセス トークンの更新 （20 ページ） を参照してください。
- **refresh_expires**_in は、更新トークンが有効である秒数です。これは、常にアクセス トークンの有効期間より長くなります。

カスタム アクセス トークンの要求

パスワード付与アクセス トークンを使用することができます。カスタム アクセス トークンを要求することもできます。カスタムトークンを使用すると、トークン使用を区別しやすくするために（自分自身のために）サブジェクト名を指定できます。パスワードトークン用に返されたデフォルト値が要件を満たさない場合は、特定の有効期間を要求することもできます。

始める前に

カスタム トークンを取得する前に、パスワード付与アクセス トークンを取得する必要があります。パスワード付与アクセス トークンの要求 （16 ページ） を参照してください。

また、次の点に注意してください。

- ローカルユーザーである場合のみ、カスタムトークンを要求できます。外部ユーザーはカスタムトークンを要求できません。
- 入手したユニットでのみカスタムトークンを使用できます。ハイ アベイラビリティ グループのピア デバイスではトークンを使用できません。

手順

ステップ 1 カスタム アクセス トークン付与のための JSON オブジェクトを作成します。

```
{
  "grant_type": "custom_token",
```

```
  "access_token": "string",
  "desired_expires_in": 0,
  "desired_refresh_expires_in": 0,
  "desired_subject": "string",
  "desired_refresh_count": 0
}
```

説明：

- **access_token** は、有効なパスワード付与アクセストークンです。
- **desired_expires_in** は、カスタムアクセストークンが有効である秒数を表す整数です。比較すると、パスワード付与トークンは 1800 秒間有効です。
- **desired_refresh_expires_in** は、カスタム更新トークンが有効である秒数を表す整数です。更新トークンを取得する場合は、この値が **desired_expires_in** 値より大きいことを確認してください。比較すると、パスワード付与更新トークンは 2400 秒間有効です。**desired_refresh_count** に 0 を指定した場合、このパラメータは不要です。
- **desired_subject** は、カスタムトークンに付ける名前です。
- **desired_refresh_count** は、トークンを更新できる回数です。更新トークンを取得しない場合は、0 を指定します。更新トークンがない場合は、既存のアクセストークンの有効期限が切れるときに新しいアクセス トークンを取得する必要があります。

以下は、2400 秒で有効期限が切れる API クライアントのカスタム トークン、および 3000 秒で有効期限が切れる更新トークンを要求します。トークンは 3 回更新することができます。

```
{{
  "grant_type": "custom_token",
  "access_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1MDI4MzI2NjcsInN
1YiI6ImFkbWluIiwianRpIjoiMGM3ZDBmNDgtODIwMS0xMWU3LWE4MWMtMDcwZmYzOW
U3ZjQ0IiwibmJmIjoxNTAyODMyNjY3LCJleHAiOjE1MDI4MzQ0NjcsInJlZnJlc2hU
b2tlbkV4cGlyZXNBdCI6MTUwMjgzNTA2NzQxOSwidG9rZW5UeXBlIjoiSldUX0FjY2
VzcyIsIm9yaWdpbiI6InBhc3N3b3JkIn0.b2hI6fVA_GbmhCOPM-ZUx6IC8SgCk1Ak
HXI-llV0r7s",
  "desired_expires_in": 2400,
  "desired_refresh_expires_in": 3000,
  "desired_subject": "api-client",
  "desired_refresh_count": 3
}
```

ステップ 2 POST /fdm/token を使用して、アクセストークンを取得します。

たとえば、**curl** コマンドは次のようになります。

```
curl -X POST --header 'Content-Type: application/json' --header
'Accept: application/json' -d '{
   "grant_type": "custom_token",
   "access_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1MDI4MzU5N
jgsInN1YiI6ImFkbWluIiwianRpIjoiYmMyNjM4N2EtODIwOC0xMWU3LWE4MWM
tYzNlYTZkZjJjZThjIiwibmJmIjoxNTAyODM1OTY4LCJleHAiOjE1MDI4Mzc3N
jgsInJlZnJlc2hUb2tlbkV4cGlyZXNBdCI6MTUwMjgzODM2ODYwNiwidG9rZW5
UeXBlIjoiSldUX0FjY2VzcyIsIm9yaWdpbiI6InBhc3N3b3JkIn0.acOE_Y4SE
ds-NE4Qw99fQlUzdoSkhsjInaCh0a9WK38",
   "desired_expires_in": 2400,
```

```
    "desired_refresh_expires_in": 3000,
    "desired_subject": "api-client",
    "desired_refresh_count": 3
}' 'https://ftd.example.com/api/fdm/[最新（latest）]/fdm/token'
```

ステップ 3 応答からアクセス トークンと更新トークンを取得します。

正常な応答（ステータス コード 200）は次のようになります。

```
{
  "access_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1MDI4MzU5O
TEsInN1YiI6ImFwaS1jbGllbnQiLCJqdGkiOiJjOWIxYzdjYi04MjA4LTExZT
ctYTgxYy02YmY0NzY3ZmRmZGUiLCJuYmYiOjE1MDI4MzU5OTEsImV4cCI6MTU
wMjgzODM5MSwicmVmcmVzaFRva2VuRXhwaXJlc0F0IjoxNTAyODM4OTkxMzMx
LCJ0b2tlblR5cGUiOiJKV1RfQWNjZXNzIiwib3JpZ2luIjoiY3VzdG9tIn0.9
IVzLjGffVQffHAWdrNkrYfvuO6TgpJ7Zi_z3RYubN8",
  "expires_in": 2400,
  "token_type": "Bearer",
  "refresh_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1MDI4MzU5
OTEsInN1YiI6ImFwaS1jbGllbnQiLCJqdGkiOiJjOWIxYzdjYi04MjA4LTExZ
TctYTgxYy02YmY0NzY3ZmRmZGUiLCJuYmYiOjE1MDI4MzU5OTEsImV4cCI6MT
UwMjgzODk5MSwiYWNjZXNzVG9rZW5FeHBpcmVzQXQiOjE1MDI4MzgzOTEzMzE
sInJlZnJlc2hDb3VudCI6MywidG9rZW5UeXBlIjoiSldUX1JlZnJlc2giLCJv
cmlnaW4iOiJjdXN0b20ifQ.qseqjg3Uo183YvfN_77iJZELEqwpWw5AbKAqAn
CIcSA",
  "refresh_expires_in": 3000
}
```

説明：

- **access_token** は、API コールに含める必要があるベアラートークンです。API コールでのアクセス トークンの使用 （19 ページ） を参照してください。
- **expires_in** は、アクセストークンが有効である、トークン発行時からの秒数です。
- **refresh_token** は、更新要求で使用するトークンです。アクセス トークンの更新 （20 ページ） を参照してください。
- **refresh_expires_**in は、更新トークンが有効である秒数です。これは、常にアクセス トークンの有効期間より長くなります。

API コールでのアクセス トークンの使用

パスワード付与またはカスタム アクセス トークンを取得した後は、それを HTTPS 要求への [認証：ベアラー（Authorization: Bearer）] ヘッダーの各 API コールに含める必要があります。

たとえば、GET /object/networks を実行するための **curl** コマンドは次のようになります。

```
curl -k -X GET -H 'Accept: application/json'
-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.yJp
YXQiOjE1MDI4MzU5OTEsInN1YiI6ImFwaS1jbGllbnQiLCJqdG
kiOiJjOWIxYzdjYi04MjA4LTExZTctYTgxYy02YmY0NzY3ZmRm
```

```
ZGUiLCJuYmYiOjE1MDI4MzU5OTEsImV4cCI6MTUwMjgzODM5MS
wicmVmcmVzaFRva2VuRXhwaXJlc0F0IjoxNTAyODM4OTkxMzMx
LCJ0b2tlblR5cGUiOiJKV1RfQWNjZXNzIiwib3JpZ2luIjoiY3
VzdG9tIn0.9IVzLjGffVQffHAWdrNkrYfvuO6TgpJ7Zi_z3RYu
bN8'
'https://ftd.example.com/api/fdm/[最新（latest）]/object/networks'
```

（注） APIエクスプローラを使用してメソッドおよびリソースを試す場合、表示される **curl** コマンドには [認証：ベアラー（Authorization: Bearer）] ヘッダーは含まれません。しかし、API クライアントから呼び出しを行う際にこのヘッダーを追加する必要があります。

アクセス トークンの更新

アクセス トークンの有効期限が切れたら、元の付与で提供された更新トークンを使用して更新する必要があります。更新されたアクセス トークンは、実際には元のアクセス トークンとは異なります。「更新」では、実際にアクセス トークンと更新トークンの新しいペアが提供され、古いアクセス トークンの期間が延長されるだけではありません。

手順

ステップ 1 更新トークン付与のための JSON オブジェクトを作成します。

```
{
  "grant_type": "refresh_token",
  "refresh_token": "string"
}
```

refresh_token はパスワード付与、またはカスタムアクセストークン付与から取得できます。

次に例を示します。

```
{
  "grant_type": "refresh_token",
  "refresh_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQ
iOjE1MDI4MzU5OTEsInN1YiI6ImFwaS1jbGllbnQiLCJqdGk
iOiJjOWIxYzdjYi04MjA4LTExZTctYTgxYy02YmY0NzY3ZmR
mZGUiLCJuYmYiOjE1MDI4MzU5OTEsImV4cCI6MTUwMjgzODk
5MSwiYWNjZXNzVG9rZW5FeHBpcmVzQXQiOjE1MDI4MzgzOTE
zMzEsInJlZnJlc2hDb3VudCI6MywidG9rZW5UeXBlIjoiSld
UX1JlZnJlc2giLCJvcmlnaW4iOiJjdXN0b20ifQ.qseqjg3U
o183YvfN_77iJZELEqwpWw5AbKAqAnCIcSA"
}
```

ステップ 2 POST /fdm/token を使用して、更新されたアクセストークンを取得します。

たとえば、**curl** コマンドは次のようになります。

```
curl -X POST --header 'Content-Type: application/json' --header
'Accept: application/json' -d '{
   "grant_type": "refresh_token",
   "refresh_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1M
DI4MzU5OTEsInN1YiI6ImFwaS1jbGllbnQiLCJqdGkiOiJjOWIxYzdj
Yi04MjA4LTExZTctYTgxYy02YmY0NzY3ZmRmZGUiLCJuYmYiOjE1MDI
4MzU5OTEsImV4cCI6MTUwMjgzODk5MSwiYWNjZXNzVG9rZW5FeHBpcm
VzQXQiOjE1MDI4MzgzOTEzMzEsInJlZnJlc2hDb3VudCI6MywidG9rZ
W5UeXBlIjoiSldUX1JlZnJlc2giLCJvcmlnaW4iOiJjdXN0b20ifQ.q
seqjg3Uo183YvfN_77iJZELEqwpWw5AbKAqAnCIcSA"
 }' 'https://ftd.example.com/api/fdm/[最新（latest）]/fdm/token'
```

ステップ 3 応答からアクセス トークンと更新トークンを取得します。

正常な応答（ステータス コード200）は次のようになります。この例では、更新トークンはカスタム トークン用でした。有効期間は、元のカスタム アクセス トークン要求の値に基づいています。

```
{
  "access_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQ
iOjE1MDI4Mzc1MTAsInN1YiI6ImFwaS1jbGllbnQiLCJqdG
kiOiJjOWIxYzdjYi04MjA4LTExZTctYTgxYy02YmY0NzY3Z
mRmZGUiLCJuYmYiOjE1MDI4Mzc1MTAsImV4cCI6MTUwMjgz
OTkxMSwicmVmcmVzaFRva2VuRXhwaXJlc0F0IjoxNTAyODQ
wNTEwNzQxLCJ0b2tlblR5cGUiOiJKV1RfQWNjZXNzIiwib3
JpZ2luIjoiY3VzdG9tIn0.fAAreX0DdnuqnM0Bs0NXYnI-9
jkpyW1pWDMwgwO_h7A",
  "expires_in": 2400,
  "token_type": "Bearer",
  "refresh_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYX
QiOjE1MDI4Mzc1MTAsInN1YiI6ImFwaS1jbGllbnQiLCJqd
GkiOiJjOWIxYzdjYi04MjA4LTExZTctYTgxYy02YmY0NzY3
ZmRmZGUiLCJuYmYiOjE1MDI4Mzc1MTAsImV4cCI6MTUwMjg
0MDUxMCwiYWNjZXNzVG9rZW5FeHBpcmVzQXQiOjE1MDI4Mz
k5MTEwNzIsInJlZnJlc2hDb3VudCI6MiwidG9rZW5UeXBlI
joiSldUX1JlZnJlc2giLCJvcmlnaW4iOiJjdXN0b20ifQ.p
Adc2N0oun7Yyw872qK12pFlix4arAwyMETD1ErKu5c",
  "refresh_expires_in": 3000
}
```

説明：

- **access_token** は、API コールに含める必要があるベアラートークンです。API コールでのアクセス トークンの使用 （19 ページ） を参照してください。
- **expires_in** は、アクセストークンが有効である、トークン発行時からの秒数です。
- **refresh_token** は、更新要求で使用するトークンです。
- **refresh_expires_**in は、更新トークンが有効である秒数です。これは、常にアクセス トークンの有効期間より長くなります。

アクセス トークンの無効化

アクセス トークンは特定の期間有効であるため、ユーザが API クライアントからログアウトするときに、トークンを無効にすることによりクリーンアップする必要があります。これにより、脅威に対する防御デバイスへのバックドアが開いたままにならないことが確認されます。

手順

ステップ 1 無効化トークン付与のための JSON オブジェクトを作成します。

```
{
  "grant_type": "revoke_token",
  "access_token": "string",
  "token_to_revoke": "string",
  "custom_token_id_to_revoke": "string",
  "custom_token_subject_to_revoke": "string"
}
```

説明：

- **access_token** は、パスワード付与アクセストークンにする必要があります。カスタム アクセス トークンを使用してトークンを無効にすることはできません。
- 次のうち 1 つ（1 つのみ）を指定する必要があります。
  - **token_to_revoke** は、無効にするパスワード付与トークンまたはカスタムトークンです。これは **access_token** と同じトークンにすることができるため、パスワード付与トークンを使用してそれ自体を無効にすることができます。
  - (使用不可.)**custom_token_id_to_revoke** は、内部の一意 ID によってカスタムアクセストークンを識別します。ただし、ユーザがこの値を取得する直接的な方法はありません。代わりにその他のオプションを使用します。
  - **custom_token_subject_to_revoke** は、無効にするカスタムアクセストークンの **desired_subject** 値です。

次に例を示します。

```
{
  "grant_type": "revoke_token",
  "access_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQ
iOjE1MDI5MDQzMjQsInN1YiI6ImFkbWluIiwianRpIjoiZT
MzNGIxOWYtODJhNy0xMWU3LWE4MWMtNGQ3NzY2ZTExMzVkI
iwibmJmIjoxNTAyOTA0MzI0LCJleHAiOjE1MDI5MDYxMjQs
InJlZnJlc2hUb2tlbkV4cGlyZXNBdCI6MTUwMjkwNjcyNDE
xMiwidG9rZW5UeXBlIjoiSldUX0FjY2VzcyIsIm9yaWdpbi
I6InBhc3N3b3JkIn0.OVZBT9yVZc4zxZfZiiLH4SZcFclaH
yCPbZJC_Gyd5FE",
  "custom_token_subject_to_revoke": "api-client"
```

```
}
```

ステップ 2 POST /fdm/token を使用して、アクセストークンを無効化します。

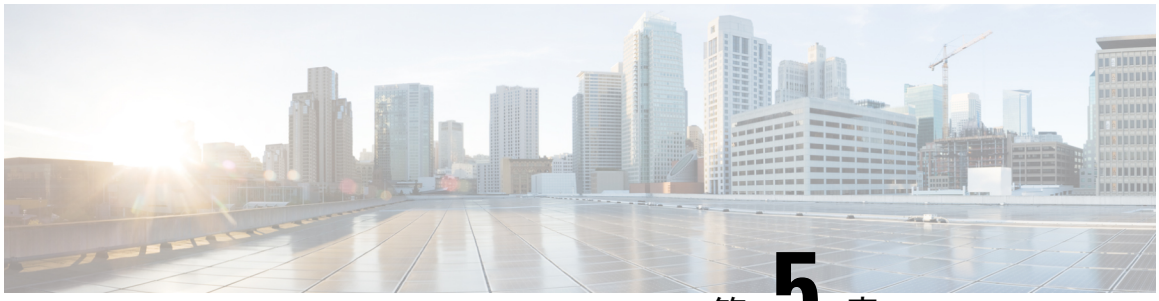
たとえば、**curl** コマンドは次のようになります。

```
curl -X POST --header 'Content-Type: application/json'
--header 'Accept: application/json' -d '{
   "grant_type": "revoke_token",
   "access_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQ
iOjE1MDI5MDQzMjQsInN1YiI6ImFkbWluIiwianRpIjoiZTM
zNGIxOWYtODJhNy0xMWU3LWE4MWMtNGQ3NzY2ZTExMzVkIiw
ibmJmIjoxNTAyOTA0MzI0LCJleHAiOjE1MDI5MDYxMjQsInJ
lZnJlc2hUb2tlbkV4cGlyZXNBdCI6MTUwMjkwNjcyNDExMiw
idG9rZW5UeXBlIjoiSldUX0FjY2VzcyIsIm9yaWdpbiI6InB
hc3N3b3JkIn0.OVZBT9yVZc4zxZfZiiLH4SZcFclaHyCPbZJ
C_Gyd5FE",
   "custom_token_subject_to_revoke": "api-client"
 }' 'https://ftd.example.com/api/fdm/[最新（latest）]/fdm/token'
```

ステップ 3 応答を評価して、トークンが無効になったことを確認します。

正常な応答（ステータス コード 200）は次のようになります。

```
{
  "message": "OK",
  "status_code": 200
}
```



第 5 章

API の外部ユーザーの設定

バージョン要件：外部 AAA を使用するには、Threat Defense バージョン 6.3(0) 以降、および Threat Defense REST API v2 以降を実行している必要があります。

外部 RADIUS AAA サーバーを使用して Threat Defense REST API へのユーザーアクセスを認証および認可するようデバイスを設定できます。組み込みのローカル **[管理（admin）]** ユーザーアカウントの代わり、またはこのユーザーアカウントに加えて RADIUS ユーザーアカウントを使用できます。

外部 AAA を使用すると、さまざまな承認レベルを持つアカウントを定義できます。これにより、デバイスに設定変更を加えるユーザーを制限しながら、サポートスタッフに読み取り専用アクセス権を提供することができます。

次の手順では、RADIUS アカウントをセットアップして、認証および認可に外部 AAA を使用するようデバイスを設定するエンドツーエンドのプロセスについて説明します。

始める前に

外部承認を使用する場合、次の操作要因に留意してください。

- デバイスが高可用性向けに設定されている場合、アクティブユニットで外部承認を設定します。その後、承認設定の展開ジョブを実行して、スタンバイデバイスへのユーザーアクセスを許可する必要があります。
- 新規ユーザーがシステムにアクセスするたびに、そのユーザーに対してユーザーリソースが作成されます。設定を展開して、そのユーザーオブジェクトを保存する必要があります。

  （Threat Defense 6.6 より前のバージョン）高可用性（HA）モードで稼働している場合、ユーザーがスタンバイユニットにログインする前に設定を展開する必要があります。管理者または読み取り/書き込みユーザーのみが展開ジョブを開始できるため、最初の読み取り専用ユーザーは、ユーザーオブジェクトを保存するための設定を別のユーザーに展開してもらう必要があります。

  Threat Defense 6.6 以降では、HA の制限が削除されます。外部ユーザーは、最初にアクティブユニットにログインせずに、スタンバイユニットにログインして設定を展開することができます。ユーザーオブジェクトはスタンバイユニットでは作成されませんが、有効

なユーザー名/パスワードが指定されていれば、ユーザーの特性はキャッシュされ、ユーザーにアクセス権が付与されます。

手順

ステップ 1 RADIUS ユーザーアカウントでの承認権限の定義 （26 ページ）。

ステップ 2 RADIUS サーバーの定義 （27 ページ）。

ステップ 3 RADIUS サーバー用 AAA サーバーグループの作成 （28 ページ）。

ステップ 4 HTTPS アクセスの認証ソースとしての AAA サーバーグループの確立 （31 ページ）。

ステップ 5 **[POST/運用/展開（POST/operational/deploy）]** を使用して、展開ジョブを開始します。

curl コマンドは次のようになります。

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json'

'https://ftd.example.com/api/fdm/[最新（latest）]/operational/deploy'
```

変更の展開の詳細については、設定変更の導入 （51 ページ）を参照してください。

ステップ 6 外部ユーザーアクセスの確認 （34 ページ）。

- RADIUS ユーザーアカウントでの承認権限の定義 （26 ページ）
- RADIUS サーバーの定義 （27 ページ）
- RADIUS サーバー用 AAA サーバーグループの作成 （28 ページ）
- HTTPS アクセスの認証ソースとしての AAA サーバーグループの確立 （31 ページ）
- 外部ユーザーアクセスの確認 （34 ページ）

RADIUS ユーザーアカウントでの承認権限の定義

外部 RADIUS サーバーからの Threat Defense REST API へのアクセスを提供できます。RADIUS 認証および認可を有効にすることにより、さまざまなレベルのアクセス権を付与でき、すべてのユーザがローカル**管理者**アカウントを使用してログインする必要がなくなります。

（注） これらの外部ユーザーは、Device Manager についても認証されます。

ロールベースのアクセス制御（RBAC）を提供するには、RADIUS サーバ上のユーザ アカウントを更新して **cisco-av-pair** 属性を定義します。この属性はユーザーアカウントで正しく定義されている必要があります。正しく定義されていないと、ユーザーの REST API へのアクセスが拒否されます。cisco-av-pair 属性でサポートされる値は、次のとおりです。

- **fdm.userrole.authority.admin** はフル管理者アクセスを提供します。これらのユーザは、ローカル**管理者**ユーザが実行できるすべてのアクションを実行できます。

- **fdm.userrole.authority.rw** は読み取り/書き込みアクセスを提供します。これらのユーザは、読み取り専用ユーザが実行できるすべてのアクションを実行でき、設定を編集および展開することもできます。ただし、システムクリティカルなアクションだけは制限されます。これには、アップグレードのインストール、バックアップの作成と復元、監査ログの表示、および他のユーザーのログオフが含まれます。
- **fdm.userrole.authority.ro** は読み取り専用アクセスを提供します。これらのユーザは、ダッシュボードと設定を表示できますが、変更できません。ユーザが変更しようとすると、権限が不足していることを示すエラー メッセージが表示されます。

RADIUS サーバーの定義

適切な認証権限を定義するためのユーザーアカウントをRADIUSサーバーで設定すると、REST APIへのアクセスを認証および認可するためにサーバーが使用するデバイスを設定できます。

POST /object/radiusidentitysources リソースを使用して、定義する各 RADIUS サーバーのオブジェクトを作成します。

手順

ステップ 1 RADIUS サーバーの JSON オブジェクト本文を作成します。

このコールで使用する JSON オブジェクトの例を次に示します。

```
{
  "name": "aaa-server-1",
  "description": "RADIUS server for API access.",
  "host": "172.16.246.220",
  "timeout": 10,
  "serverAuthenticationPort": 1812,
  "serverSecretKey": "secret123",
  "type": "radiusidentitysource"
}
```

その属性は次のとおりです。

- name：オブジェクト名。この名前は、RADIUSサーバーに定義されている名前と一致している必要はありません。
- [Description]：（オプション。） オブジェクトの説明。
- host：RADIUS サーバーの IP アドレスまたは完全修飾ホスト名。
- timeout：（オプション）次のサーバーに要求を送信する前にサーバーからの応答を待機する時間の長さ（1 ～ 300 秒）です。この属性を含めない場合のデフォルトは 10 秒です。
- serverAuthenticationPort：（オプション）RADIUS認証および承認が実行されるポート。この属性を含めない場合のデフォルトは 1812 です。

- serverSecretKey：（オプション）Threat Defense デバイスと RADIUS サーバー間でデータを暗号化するために使用される共有秘密キー。キーは、大文字と小文字が区別される最大 64 文字の英数字文字列です。スペースは使用できません。キーは、英数字または下線で開始する必要があります。特殊文字 $ & - _ . + @ を使用できます。文字列は、RADIUS サーバーで設定された文字列と一致している必要があります。秘密キーを設定していない場合、接続は暗号化されません。

ステップ 2 オブジェクトをポストします。

たとえば、curl コマンドは次のようになります。

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json'
 -d '{
    "name": "aaa-server-1",
   "description": "RADIUS server for API access.",
   "host": "172.16.246.220",
   "timeout": 10,
   "serverAuthenticationPort": 1812,
   "serverSecretKey": "secret123",
   "type": "radiusidentitysource"
 }' 'https://ftd.example.com/api/fdm/[最新（latest）]/object/radiusidentitysources'
```

ステップ 3 応答を確認します。

取得する応答コードは200である必要があります。正常な応答本文は次のようになります。秘密鍵などの機密情報は、応答ではマスク処理されていることに注意してください。

```
{
  "version": "nfamb3cr2jlyi",
  "name": "aaa-server-1",
  "description": "RADIUS server for API access.",
  "host": "172.16.246.220",
  "timeout": 10,
  "serverAuthenticationPort": 1812,
  "serverSecretKey": "*********",
  "capabilities": [
    "AUTHENTICATION",
    "AUTHORIZATION"
  ],
  "id": "1b962e3b-6e56-11e8-bd65-379fa8aaaba1",
  "type": "radiusidentitysource",
  "links": {
    "self": "https://ftd.example.com/api/fdm/[最新（latest）]/object/
radiusidentitysources/1b962e3b-6e56-11e8-bd65-379fa8aaaba1"
  }
}
```

RADIUS サーバー用 AAA サーバーグループの作成

RADIUS サーバー オブジェクトを作成した後、POST /object/radiusidentitysourcegroups リソースを使用して AAA グループを作成し、radiusidentitysource オブジェクトを含めます。

最大 16 台の RADIUS サーバーを RADIUS AAA サーバーグループに追加できます。これらのサーバーは相互にバックアップになる必要があります。つまり、同じユーザーアカウントリストを持つ必要があります。

手順

ステップ 1 RADIUS サーバーグループの JSON オブジェクト本文を作成します。

このコールで使用する JSON オブジェクトの例を次に示します。

```
{
  "name": "radius-group",
  "maxFailedAttempts": 3,
  "deadTime": 10,
  "description": "AAA RADIUS server group.",
  "radiusIdentitySources": [
    {
      "id": "1b962e3b-6e56-11e8-bd65-379fa8aaaba1",
      "type": "radiusidentitysource",
      "version": "nfamb3cr2jlyi",
      "name": "aaa-server-1"
    }
  ],
  "type": "radiusidentitysourcegroup"
}
```

その属性は次のとおりです。

- name：オブジェクト名。メンバー RADIUS サーバーで定義されているものと一致している必要はありません。
- maxFailedAttempts：（オプション）失敗したサーバーは、すべてのサーバーが失敗した後にのみ再アクティブ化されます。デッドタイムは、最後のサーバーが失敗した後にすべてのサーバーを再アクティブ化するまで待機する時間の長さ（0 ～ 1440分）です。この属性が含まれていない場合、デフォルトは 10 分です。
- deadTime：（オプション）次のサーバーを試行する前に、グループ内の RADIUS サーバーに送信された要求の失敗数（応答がなかった要求の数）。1 ～ 5 を指定できます。デフォルトは 3 です。最大失敗試行回数を超えると、システムはそのサーバーを故障としてマークします。

  特定の機能について、ローカルデータベースを使用するフォールバック方式を設定していて、グループ内のすべてのサーバーが応答に失敗した場合、そのグループは非応答と見なされ、フォールバック方式が試行されます。サーバー グループはデッド タイムの間、非応答とマークされたままになるため、その期間内に追加の AAA 要求でサーバー グループへの接続は試行されず、フォールバック方式がすぐに使用されます。
- [Description]：（オプション。） オブジェクトの説明。
- radiusIdentitySources：グループに含める RADIUS サーバーを定義する各 radiusidentitysource オブジェクトを定義する項目のグループ。[ブラケット]内に項目を入れます。各オブジェクトの属性およびシンタックスは次のとおりです。個々のオブジェクトから、id、version、および name 属性の値を取得します。その情報は、オブジェクトを作成する際に応答本文

に含まれます。**GET /object/radiusidentitysources** コールから情報を取得することもできます。type は、 radiusidentitysource である必要があります。

```
{
  "id": "1b962e3b-6e56-11e8-bd65-379fa8aaaba1",
  "type": "radiusidentitysource",
  "version": "nfamb3cr2jlyi",
  "name": "aaa-server-1"
}
```

ステップ 2 オブジェクトをポストします。

たとえば、curl コマンドは次のようになります。

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json'
 -d '{
   "name": "radius-group",
   "maxFailedAttempts": 3,
   "deadTime": 10,
   "description": "AAA RADIUS server group.",
   "radiusIdentitySources": [
     {
       "id": "1b962e3b-6e56-11e8-bd65-379fa8aaaba1",
       "type": "radiusidentitysource",
       "version": "nfamb3cr2jlyi",
       "name": "aaa-server-1"
     }
   ],
   "type": "radiusidentitysourcegroup"
 }' 'https://ftd.example.com/api/fdm/[最新（latest）]/object/radiusidentitysourcegroups'
```

ステップ 3 応答を確認します。

取得する応答コードは 200 である必要があります。正常な応答本文は次のようになります。

```
{
  "version": "7r572novdiyy",
  "name": "radius-group",
  "maxFailedAttempts": 3,
  "deadTime": 10,
  "description": "AAA RADIUS server group.",
  "radiusIdentitySources": [
    {
      "version": "nfamb3cr2jlyi",
      "name": "aaa-server-1",
      "id": "1b962e3b-6e56-11e8-bd65-379fa8aaaba1",
      "type": "radiusidentitysource"
    }
  ],
  "activeDirectoryRealm": null,
  "id": "0a7996ae-6e5b-11e8-bd65-dbab801c44b9",
  "type": "radiusidentitysourcegroup",
  "links": {
    "self": "https://ftd.example.com/api/fdm/[最新（latest）]/object/
radiusidentitysourcegroups/0a7996ae-6e5b-11e8-bd65-dbab801c44b9"
```

```
  }
}
```

HTTPS アクセスの認証ソースとしての AAA サーバーグループの確立

PUT /devicesettings/default/aaasettings/{objId} リソースを使用して、ユーザー認証のアイデンティティ ソースである RADIUS AAA サーバー グループを特定します。

POST メソッドはありません。システム認証に必要なオブジェクトはすでに存在しています。最初に GET を実行して、関連 ID とバージョンの値を確認する必要があります。

手順

ステップ 1 GET /devicesettings/default/aaasettings を使用して、aaasettings オブジェクトの属性を確認します。

curl コマンドは次のようになります。

```
curl -X GET --header 'Accept: application/json'
'https://ftd.example.com/api/fdm/[最新（latest）]/devicesettings/default/aaasettings'
```

たとえば、応答本文は次のようになります。この例は、ローカル アイデンティティ ソースが HTTPS アクセス用に定義されているソースであることを示しています。これは、REST API とは関係がない SSH アクセスでも使用されます。

```
{
  "items": [
    {
      "version": "du52clrtmawlt",
      "name": "HTTPS",
      "identitySourceGroup": {
        "version": "cynutari5ffkl",
        "name": "LocalIdentitySource",
        "id": "e3e74c32-3c03-11e8-983b-95c21a1b6da9",
        "type": "localidentitysource"
      },
      "description": null,
      "protocolType": "HTTPS",
      "useLocal": "NOT_APPLICABLE",
      "id": "00000003-0000-0000-0000-000000000007",
      "type": "aaasetting",
      "links": {
        "self": "https://ftd.example.com/api/fdm/[最新（latest）]/
devicesettings/default/aaasettings/00000003-0000-0000-0000-000000000007"
      }
    },
    {
      "version": "fgkhvu4kwucgv",
      "name": "SSH",
      "identitySourceGroup": {
        "version": "cynutari5ffkl",
```

```
        "name": "LocalIdentitySource",
        "id": "e3e74c32-3c03-11e8-983b-95c21a1b6da9",
        "type": "localidentitysource"
      },
      "description": null,
      "protocolType": "SSH",
      "useLocal": "NOT_APPLICABLE",
      "id": "00000003-0000-0000-0000-000000000008",
      "type": "aaasetting",
      "links": {
        "self": "https://ftd.example.com/api/fdm/[最新（latest）]/
devicesettings/default/aaasettings/00000003-0000-0000-0000-000000000008"
      }
    }
  ],
  "paging": {
    "prev": [],
    "next": [],
    "limit": 10,
    "offset": 0,
    "count": 2,
    "pages": 0
  }
}
```


ステップ 2 （オプション）表示範囲を限定するには、GET /devicesettings/default/aaasettings/{objId} を使用して、HTTPS AAA 設定オブジェクトのコピーを取得します。

PUT コールでは HTTPS オブジェクトのみ更新されます。SSH オブジェクトを更新する必要はありません。

この例では、HTTPS オブジェクトの ID は 00000003-0000-0000-0000-000000000007 であるため、curl コマンドは次のようになります。

```
curl -X GET --header 'Accept: application/json'
'https://ftd.example.com/api/fdm/[最新（latest）]/devicesettings/
default/aaasettings/00000003-0000-0000-0000-000000000007'
```

応答本文は次のようになります。


```
{
  "version": "ha4653ootep7z",
  "name": "HTTPS",
  "identitySourceGroup": {
    "version": "cynutari5ffkl",
    "name": "LocalIdentitySource",
    "id": "e3e74c32-3c03-11e8-983b-95c21a1b6da9",
    "type": "localidentitysource"
  },
  "description": null,
  "protocolType": "HTTPS",
  "useLocal": "NOT_APPLICABLE",
  "id": "00000003-0000-0000-0000-000000000007",
  "type": "aaasetting",
  "links": {
    "self": "https://ftd.example.com/api/fdm/[最新（latest）]/
devicesettings/default/aaasettings/00000003-0000-0000-0000-000000000007"
  }
```


ステップ 3 AAA 管理アクセス用の JSON オブジェクト本文を作成します。

このコールで使用する JSON オブジェクトの例を次に示します。

```
{
  "version": "ha4653ootep7z",
  "name": "HTTPS",
  "identitySourceGroup": {
    "id": "0a7996ae-6e5b-11e8-bd65-dbab801c44b9",
    "type": "radiusidentitysourcegroup",
    "version": "7r572novdiyy",
    "name": "radius-group"
  },
  "description": null,
  "protocolType": "HTTPS",
  "useLocal": "BEFORE",
  "id": "00000003-0000-0000-0000-000000000007",
  "type": "aaasetting"
}
```

その属性は次のとおりです。

- version：HTTPS オブジェクトのバージョン。GET コールの応答本文でこの値を検索します。
- name：オブジェクト名、HTTPS。GET コールの応答本文でこの値を検索します。
- identitySourceGroup：RADIUS サーバー グループを特定します。サーバー グループの作成（または GET /object/radiusidentitysourcegroups コール）時に応答本文から id、version、および name の値を取得します。type は、radiusidentitysource である必要があります。
- [Description]：（オプション。） オブジェクトの説明。
- protocolType：このソースが適用されるプロトコル、HTTPS。
- useLocal：ローカル管理者ユーザー アカウントを含む、ローカル アイデンティティ ソースの使用方法。次のいずれかのオプションを入力します。
  - **[前（Before）]**：最初にローカルソースに照らしてユーザー名とパスワードがチェックされます。
  - **[後（After）]**：外部ソースを利用できない場合、またはユーザーアカウントが外部ソース内で見つからない場合にのみローカルソースがチェックされます。
  - [使用しない（Never）]：（非推奨）ローカル ソースがまったく使用されないため、[管理者（admin）] ユーザーとしてログインできません。

    注意 **[使用しない（Never）]** を選択すると、**[管理（admin）]** アカウントを使用して Device Manager にログインする、または API を使用することができなくなります。RADIUS サーバが使用できなくなった場合または RADIUS サーバのアカウント設定が間違っている場合は、システムがロックされます。

- id：HTTPS オブジェクトの ID 値。GET コールの応答本文でこの値を検索します。
- type：オブジェクト タイプ、aaasetting。

ステップ 4 オブジェクトを PUT します。

たとえば、curl コマンドは次のようになります。URL の {objId} と JSON オブジェクト内の aaasettings オブジェクトの ID は同じである点に注意してください。

```
curl -X PUT --header 'Content-Type: application/json' --header 'Accept: application/json'
 -d '{
   "version": "ha4653ootep7z",
   "name": "HTTPS",
   "identitySourceGroup": {
     "id": "0a7996ae-6e5b-11e8-bd65-dbab801c44b9",
     "type": "radiusidentitysourcegroup",
     "version": "7r572novdiyy",
     "name": "radius-group"
   },
   "description": null,
   "protocolType": "HTTPS",
   "useLocal": "BEFORE",
   "id": "00000003-0000-0000-0000-000000000007",
   "type": "aaasetting"
 }' 'https://ftd.example.com/api/fdm/[最新（latest）]/devicesettings/
default/aaasettings/00000003-0000-0000-0000-000000000007'
```

ステップ 5 応答を確認します。

取得する応答コードは 200 である必要があります。正常な応答本文は次のようになります。

```
{
  "version": "ehxcytq4iccb3",
  "name": "HTTPS",
  "identitySourceGroup": {
    "version": "7r572novdiyy",
    "name": "radius-group",
    "id": "0a7996ae-6e5b-11e8-bd65-dbab801c44b9",
    "type": "radiusidentitysourcegroup"
  },
  "description": null,
  "protocolType": "HTTPS",
  "useLocal": "BEFORE",
  "id": "00000003-0000-0000-0000-000000000007",
  "type": "aaasetting",
  "links": {
    "self": "https://ftd.example.com/api/fdm/[最新（latest）]/devicesettings/
default/aaasettings/00000003-0000-0000-0000-000000000007"
  }
}
```

外部ユーザーアクセスの確認

展開ジョブが完了したら、Device Manager と REST API の両方への外部ユーザーアクセスをテストできます。

手順

ステップ 1 有効な cisco-av-pair 属性を持つ外部ユーザー名を使用して、Device Manager にログインします。

ログインが成功すると、ページの右上にユーザー名と権限レベルが表示されます。

ステップ 2 外部ユーザーの REST API トークンを取得します。

ユーザーがトークンを取得できる場合、そのユーザーは、割り当てられた権限レベルに許可されているリソースとメソッドを使用できます。

a) シンプルなパスワード付与トークンの場合は JSON オブジェクト本文を作成します。

```
{
  "grant_type": "password",
  "username": "radiusreadwriteuser1",
  "password": "Readwrite123!"
}
```

b) **POST /fdm/token** を使用して、トークンを取得します。

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept:
application/json' -d '{
   "grant_type": "password",
   "username": "radiusreadwriteuser1",
   "password": "Readwrite123!"
 }' 'https://ftd.example.com/api/fdm/[最新（latest）]/fdm/token'
```

c) 応答を評価して、トークンが付与されたことを確認します。

取得する応答コードは 200 である必要があります。トークンの取得は、システムがユーザーを認証できたことを意味します。

```
{
  "access_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1Mjg4MjM
3MTAsInN1YiI6InJhZGl1c3JlYWR3cml0ZXVzZXIxIiwianRpIjoiMjk5Zj
Q5YjYtNmU2NC0xMWU4LWJkNjUtNmY0ZmVmYjY1MzI5IiwibmJmIjoxNTI4O
DIzNzEwLCJleHAiOjE1Mjg4MjU1MTAsInJlZnJlc2hUb2tlbkV4cGlyZXNB
dCI6MTUyODgyNjExMDg4OSwidG9rZW5UeXBlIjoiSldUX0FjY2VzcyIsInV
zZXJVdWlkIjoiMjliMjBlNjctNmU2NC0xMWU4LWJkNjUtMzU4MmUwZjU5Yj
Q4IiwidXNlclJvbGUiOiJST0xFX1JFQURfV1JJVEUiLCJvcmlnaW4iOiJwY
XNzd29yZCJ9.dtKsl9IB4ds3RAktEeaSuQy_Zs2SrzLr976UtblBt28",
  "expires_in": 1800,
  "token_type": "Bearer",
  "refresh_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1Mjg4Mj
M3MTAsInN1YiI6InJhZGl1c3JlYWR3cml0ZXVzZXIxIiwianRpIjoiMjk5Z
jQ5YjYtNmU2NC0xMWU4LWJkNjUtNmY0ZmVmYjY1MzI5IiwibmJmIjoxNTI4
ODIzNzEwLCJleHAiOjE1Mjg4MjYxMTAsImFjY2Vzc1Rva2VuRXhwaXJlc0F
0IjoxNTI4ODI1NTEwODg5LCJyZWZyZXNoQ291bnQiOi0xLCJ0b2tlblR5cG
UiOiJKV1RfUmVmcmVzaCIsInVzZXJVdWlkIjoiMjliMjBlNjctNmU2NC0xM
WU4LWJkNjUtMzU4MmUwZjU5YjQ4IiwidXNlclJvbGUiOiJST0xFX1JFQURf
V1JJVEUiLCJvcmlnaW4iOiJwYXNzd29yZCJ9.Lc7MYmieNMMrjx7XoTiW-x
8Z8qFCnzfNM1apgbwLQvo",
  "refresh_expires_in": 2400
}
```

ステップ 3 **GET /object/users** を使用して、各ユーザーに作成されているユーザーオブジェクトを確認します。

Device Manager にログインする、またはアクセストークンを取得する新規ユーザーのユーザーオブジェクトは自動的に作成されます。展開ジョブを実行して、それらのユーザーオブジェクトを保存する必要があります。ハイ アベイラビリティ モードでは、ユーザーがスタンバイ装置にログインする前に展開ジョブを実行する必要があります。

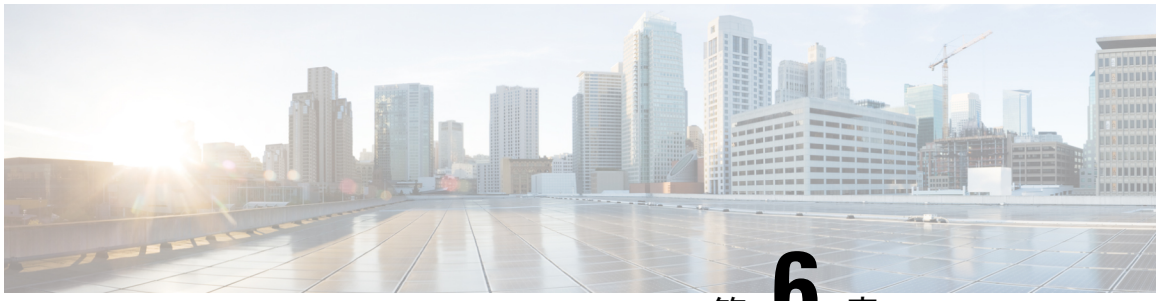
たとえば、curl コマンドは次のようになります。

```
curl -X GET --header 'Accept: application/json'
'https://ftd.example.com/api/fdm/[最新（latest）]/object/users'
```

次の応答本文は、2 人の外部ユーザーがログインしたことを示しています。**userRole** には、2 人のユーザーアカウントに対して、RADIUS サーバーに設定されている cisco-av-pair から取得した権限が表示されています。この情報を使用して、RADIUS ユーザーアカウントを正しく設定していることを確認します。**admin** ユーザーはローカルに定義されたユーザーです。

```
{
  "items": [
    {
      "version": "h2vom4wckm2js",
      "name": "radiusadminuser1",
      "password": null,
      "newPassword": null,
      "userPreferences": {
        "preferredTimeZone": "(UTC+00:00) UTC",
        "colorTheme": "NORMAL_CISCO_IDENTITY",
        "type": "userpreferences"
      },
      "userRole": "ROLE_ADMIN",
      "identitySourceId": "0a7996ae-6e5b-11e8-bd65-dbab801c44b9",
      "userServiceTypes": [
        "MGMT"
      ],
      "id": "150d9754-6e63-11e8-bd65-ed9b20f62114",
      "type": "user",
      "links": {
        "self": "https://ftd.example.com/api/fdm/[最新（latest）]/
object/users/150d9754-6e63-11e8-bd65-ed9b20f62114"
      }
    },
    {
      "version": "p4rgwcjr5colj",
      "name": "admin",
      "password": null,
      "newPassword": null,
      "userPreferences": {
        "preferredTimeZone": "(UTC-07:00) America/Los_Angeles",
        "colorTheme": "NORMAL_CISCO_IDENTITY",
        "type": "userpreferences"
      },
      "userRole": "ROLE_ADMIN",
      "identitySourceId": "e3e74c32-3c03-11e8-983b-95c21a1b6da9",
      "userServiceTypes": [
        "MGMT"
      ],
      "id": "5023d3ab-6dc5-11e8-b9ed-db6dba9bf94c",
      "type": "user",
```

```
      "links": {
        "self": "https://ftd.example.com/api/fdm/[最新（latest）]/
object/users/5023d3ab-6dc5-11e8-b9ed-db6dba9bf94c"
      }
    },
    {
      "version": "ngx7a2dixngoq",
      "name": "radiusreadwriteuser1",
      "password": null,
      "newPassword": null,
      "userPreferences": {
        "preferredTimeZone": "(UTC+00:00) UTC",
        "colorTheme": "NORMAL_CISCO_IDENTITY",
        "type": "userpreferences"
      },
      "userRole": "ROLE_READ_WRITE",
      "identitySourceId": "0a7996ae-6e5b-11e8-bd65-dbab801c44b9",
      "userServiceTypes": [
        "MGMT"
      ],
      "id": "29b20e67-6e64-11e8-bd65-3582e0f59b48",
      "type": "user",
      "links": {
        "self": "https://ftd.example.com/api/fdm/[最新（latest）]/
object/users/29b20e67-6e64-11e8-bd65-3582e0f59b48"
      }
    }
  ],
  "paging": {
    "prev": [],
    "next": [],
    "limit": 10,
    "offset": 0,
    "count": 3,
    "pages": 0
  }
}
```



第 6 章

メソッドとリソースの使用

次のトピックではさまざまなメソッドとリソースを使用する一般的な方法を説明します。

メソッドの試行と結果の解釈

API エクスプローラを使用すると、さまざまなメソッドをテストすることができます。このトピックでは、一般的なプロセスについて、システムが返す応答の説明とともに説明します。メソッドに関連する具体的なテクニックは、各メソッドタイプのトピックを参照してください。

各メソッド/リソースの [試してみよう!（Try It Out!）] ボタンにより、システムと直接対話します。GET は、実際のデータを取得し、POST/PUT は実際のリソースを作成または変更し、DELETE は実際のオブジェクトを削除します。変更はすぐには展開されませんが、システムで実際の設定変更を行っています。変更をアクティブにするには、POST /operational/deploy リソースを使用して展開ジョブを開始します。

メソッド/リソースを開くと、[応答メッセージ（Response Message）] セクションの後に、[試してみよう!（Try It Out!）] ボタンを見つけることができます。メソッド/リソースによっては、テストするにはオブジェクト ID を入力する必要があります。この場合、通常、まず親リソースで GET を実行する必要があります。詳細については、オブジェクト ID（objId）と親 ID の検索 （8 ページ）を参照してください。

POST/PUT でも、JSON モデルに必要な値を入力する必要があります。

[試してみよう!（Try It Out!）] をクリックした後、API エクスプローラは、ボタンの後のページに結果を追加します。この応答には次のセクションが含まれます。

Curl

呼び出しを行うために使用される **curl** コマンド。たとえば、GET /object/networks リソースで **[試してみよう!（Try It Out!）]** をクリックすると、次のようなメッセージが返されます。パスの要素「v」は、新しいバージョンの API によって変わります。

```
curl -X GET --header 'Accept: application/json'
'https://ftd.example.com/api/fdm/[最新（latest）]/object/networks'
```

（注） これには [認証：ベアラー（Authorization: Bearer）] ヘッダーは含まれません。これはクライアントからの API 呼び出しで必要になります。

要求 URL

要求を行うクライアントから発行する URL です。たとえば、GET /object/networks の場合：

```
https://ftd.example.com/api/fdm/[最新（latest）]/object/networks
```

レスポンス ボディ

システムがクライアントに返すオブジェクト。リソースに複数のオブジェクト（/object/network など）を含めることができる場合、GET 要求でアイテムのリストを取得します。POST/PUT/DELETE の応答は、ほとんど単一のオブジェクトです。

返される特定のコンテンツは、リソース モデルに基づきます。たとえば、GET /object/networks は次のような各オブジェクトを含む、オブジェクトのリストを返します（最初のアイテムのリストも表示されます）。リンク/自己値は、このオブジェクトを参照するために使用する URL を示すことに注意してください。オブジェクト ID は URL に含まれます。

```
{
  "items": [
    {
      "version": "900f8558-7d19-11e7-bf7b-3dcaf0c58345",
      "name": "AIM_SERVERS-205.188.1.132",
      "description": null,
      "subType": "HOST",
      "value": "205.188.1.132",
      "isSystemDefined": true,
      "id": "900fac69-7d19-11e7-bf7b-d9417b20e59e",
      "type": "networkobject",
      "links": {
        "self": "https://ftd.example.com/api/fdm/[最新（latest）]/
object/networks/900fac69-7d19-11e7-bf7b-d9417b20e59e"
      }
    },
```

GET 要求もページングのセクションに含まれます。これはGET：システムからのデータの取得 （41 ページ）で説明されています。

応答コード

HTTP 呼び出しの数値 HTTP ステータス コードが返されます。これらは、標準の HTTP ステータス コードで、RFC やウィキペディア（https://en.wikipedia.org/wiki/List_of_HTTP_status_codes など）で検索できます。たとえば、200 (OK) は、GET/PUT/POST 呼び出しの成功を示し、204 は DELETE 呼び出しの成功を示します。

レスポンス ヘッダー

HTTP 応答のパケット ヘッダーです。たとえば、GET /object/networks には、次のようなヘッダーがあります。

```
{
  "date": "Thu, 10 Aug 2017 19:19:16 GMT",
  "content-encoding": "gzip",
  "x-content-type-options": "nosniff",
  "transfer-encoding": "chunked",
  "connection": "Keep-Alive",
  "vary": "Accept-Encoding",
  "x-xss-protection": "1; mode=block",
  "pragma": "no-cache",
  "server": "Apache",
  "x-frame-options": "SAMEORIGIN",
  "strict-transport-security": "max-age=31536000 ; includeSubDomains",
  "content-type": "application/json;charset=UTF-8",
  "cache-control": "no-cache, no-store, max-age=0, must-revalidate",
  "accept-ranges": "bytes",
  "keep-alive": "timeout=5, max=99",
  "expires": "0"
}
```

GET：システムからのデータの取得

デバイスから情報を読み取るには、GET メソッドを使用します。

リソースに複数のオブジェクトが含まれている可能性がある場合は、応答でオブジェクトのリストを取得します。URL にクエリ パラメータを含めて、返されるオブジェクトの数を制御することができます。デフォルトでは、オブジェクト リストの先頭から 10 個のオブジェクトが返されます。

次の手順では、API エクスプローラで GET コールを使用する一般的なアプローチを説明します。API クライアントのコード例を使用します。

手順

ステップ 1 APIエクスプローラでGETメソッドを開きます（最初に、グループを開いてメソッドとリソースを参照します）。

ステップ 2 使用するメソッドがURL内にオブジェクトまたは親のIDを必要とする場合は、親メソッドを使用して必要なIDを取得します。

たとえば、GET /objects/networks/{objId} は特定のオブジェクトの ID を必要とします。GET /objects/networks メソッドを使用してネットワーク オブジェクトのリストを取得し、調べるオブジェクトの id 値を探します。この場合、GET /object/networks コールで返される情報は GET /objects/network/{objId} の場合と同じになることに注意してください。オブジェクト ID（objId）と親 ID の検索 （8 ページ）を参照してください。

ステップ 3 **Parameters** セクションで、以下のオプションを設定します。

- **objId**：オブジェクト ID は、URL 内で必要な場合は常に必要です。たとえば、900fac69-7d19-11e7-bf7b-d9417b20e59e です。
- **parentId**：親 ID はオブジェクト ID と同等ですが、単に階層内で高い位置にある親に対するものです。たとえば、GET /policy/intrusion は侵入ポリシーのリストを返しますが、GET /policy/intrusion/{parentId}/intrusionrules はそれらのポリシーの 1 つで定義されているルールを返します。親 ID は GET /policy/intrusion から取得します。
- **offset**：複数のオブジェクトをサポートするリソースの場合、オブジェクトを返すのを開始するリスト内の位置。デフォルトは 0 で、リストの先頭を示します。
- **limit**：応答で返されるオブジェクトの最大数。デフォルトは 10 です。最大値は 1000 です。無効な値を入力すると、自動的に 1000 に変更されます。
- **sort**：応答で返されるオブジェクトを並べ替える方法。デフォルトの並べ替え順は、**name** 値でアルファベット順です。並べ替え順を変更するには、並べ替えに使用するリソース内の属性の名前を入力します。たとえば、ネットワーク オブジェクトで **sort=value** を使用すると、value 属性（つまり IP アドレス）で並べ替えることができます。逆順に並べ替えるには、マイナス記号を含めます（例：**sort=-name**）。
- **filter**（一部のリソースでは使用不可）：フィルタ条件に一致する項目のみを返します。フィルタ値の形式は {キー}{演算子}{値} で、キーは属性名、値はフィルタに使用する文字列です。項目間にスペースはありません。フィルタ処理できるフィールドは、API エクスプローラの **filter** パラメータの説明に一覧表示されています。フィールドはオブジェクトごとに異なります。複数フィールドのフィルタリングをサポートしているオブジェクトの場合、複数の値をセミコロン（;）で区切って **filter** パラメータに含めることができます。たとえば、GET /policy/intrusionpolicies/{parentId}/intrusionrules の gid:1;sid:105 をフィルタ処理できます。使用可能な演算子は次のとおりです。
  - **:**：等号。たとえば、**filter=name!Canada** です。
  - **!**：不等号。たとえば、**filter=name!Canada** です。
  - **~**：類似。たとえば、**filter=name~United** です。
- **filter=fts~***string*（一部のリソースでは使用不可）：フィルタに一致する項目のみを返します。**Fts~** オプションはフルテキスト検索を指定します。オブジェクト内の全属性で文字列が検索されます。オプションで部分文字列を使用できます。アスタリスク（*）をワイルドカードとして使用し、1 文字以上と一致させます。次の文字を含めないでください。検索文字列の一部としてサポートされていません。?~!{}<>:%。次の文字は無視されます。;#&。

たとえば、GET /object/networks?filter=fts~10 を使用して、最初のオクテットとして 10 を持つすべてのネットワークオブジェクトを見つけることができます。新しく作成された、または更新されたオブジェクトのインデックス化には 3 ～ 5 秒かかるため、それらのオブジェクトのフルテキスト検索を実行する前に少し待つ必要があります。

- **filter=fetchZeroHitCount:{true | false}**（アクセスルールのみで使用可能）：**includeHitCounts=true** を指定した場合、このフィルタオプションを使用して、ヒットしていない（つまりヒットカウントがゼロである）ルールを含める（**true**）または除外（**false**）することができます。デフォルトは **true** です。
- **includeHitCounts**（アクセスルールのみで使用可能）：ポリシーにルールのヒットカウント情報を含めるかどうか。ヒットカウントを取得するには、**includeHitCounts=true** を指定します。**false**（デフォルト）を指定してヒットカウントを除外します。ヒットカウント情報は、返されたオブジェクトの hitCount 属性で返されます。
- **time_duration**（トレンド レポートの場合にのみ使用可能）：過去何秒間までのレポートを含めるか。たとえば、1800 は過去 30 分のレポートを返します。

（注） {objId} と {parentId} は URL パスの一部ですが、**offset**、**limit**、**sort**、**filter**、**includeHitCounts**、および **time_duration** の各パラメータは URL の末尾に **?** 文字に続けて追加します。

ステップ 4 [試してみよう!（Try It Out!）] ボタンをクリックし、応答を調べます。

コールが成功すると（戻りコード 200）、応答本文にはコールの内容に応じて 1 つのオブジェクトまたはオブジェクトのリストが含まれています。応答の一般的な構造と内容については、メソッドの試行と結果の解釈 （39 ページ）を参照してください。

GET 要求にはページング セクションが含まれています。コールで返されたオブジェクトより多くのオブジェクトが存在する場合、**prev** および **next** の値はその前または後のオブジェクトセットを取得する方法を示しています。**count** 値はオブジェクト全体の数を示しています。**limit** 値は応答で返される項目数を示しています。**offset** 値は返されるオブジェクトの開始位置を示しており、0 はリストの先頭を示しています。

```
"paging": {
    "prev": [],
    "next": [
     "https://ftd.example.com/api/fdm/[最新（latest）]/object/networks?limit=10&offset=10"

    ],
    "limit": 10,
    "offset": 0,
    "count": 22,
    "pages": 0
  }
```

POST：新しいオブジェクトの作成

あるリソースタイプの新しいオブジェクトを作成するには、POSTメソッドを使用します。たとえば、POSTを使用して新しいネットワーク オブジェクトを作成します。

次の手順では、APIエクスプローラでPOSTコールを使用する一般的なアプローチを説明します。APIクライアントのコード例を使用します。

手順

ステップ1 APIエクスプローラでPOSTメソッドを開きます（最初に、グループを開いてメソッドとリソースを参照します）。

ステップ2 [応答クラス（Response Class）] 見出しで [モデル（Model）] をクリックし、リソース属性のデータ型および値について読みます。

ステップ3 **[パラメータ（Parameters）]** 見出しの下で、以下のオプションを設定します（設定可能な場合）。

- **[parentId]**：このオブジェクトを含む親オブジェクトのID。たとえば、SSLルールを追加する場合は、SSL復号ポリシーのID。
- **[at]**：番号付きリストでオブジェクトを編成する親に属するオブジェクト（SSL復号ポリシーなど）の場合は、その場所をオブジェクトに挿入します。整数を使用して場所を示します（リストの先頭は0）。デフォルトでは、新しいオブジェクトはリストの最後に追加されます。

（注） {objId} と {parentId} はURLパスの一部ですが、**at**パラメータをURLの末尾の**?**文字に続けて追加します。

ステップ4 また、**[パラメータ（Parameters）]** 見出しの下で、**body**パラメータの**[データ型（Data Type）]**>**[サンプル値（Example Value）]** 列に表示されるJSONモデルをクリックします。

このボックスをクリックすると、JSONモデルが**body**パラメータの [値（Value）] 列にロードされます。たとえば、POST /object/networksリソースのボックスをクリックすると、次の本文がロードされます。

```
{
  "name": "string",
  "description": "string",
  "subType": "HOST",
  "value": "string",
  "type": "networkobject"
}
```

ステップ5 **body** JSONオブジェクト属性の必須値を入力します。

列挙値の場合は、必ず**[応答クラス（Response Class）]**>**[モデル（Model）]**で許可される値を確認してください。たとえば、値を入力して[subType]のデフォルト値を変更することにより、サブネット（ホストではない）アドレスのネットワーク オブジェクトを作成できます。

```
{
  "name": "new_network_object",
  "description": "A subnet object created using the REST API.",
  "subType": "NETWORK",
  "value": "10.100.10.0/24",
  "type": "networkobject"
}
```

ステップ 6 [試してみよう!（Try It Out!）] ボタンをクリックし、応答を調べます。

システムの更新に使用される**curl**コマンドを調べます。追加のヘッダーに注意してください。APIクライアントを作成するときは、これらのヘッダーフィールドと値も含める必要があります。たとえば、サンプルオブジェクトを作成するための**curl**コマンドは次のとおりです。Content-Type ヘッダーと Accept ヘッダーに注意してください。

```
curl -X POST --header 'Content-Type: application/json' \
   --header 'Accept: application/json' -d '{ \
   "name": "new_network_object", \
   "description": "A subnet object created using the REST API.", \
   "subType": "NETWORK", \
   "value": "10.100.10.0/24", \
   "type": "networkobject" \
 }' 'https://ftd.example.com/api/fdm/[最新（latest）]/object/networks'
```

コールが成功すると（戻りコード200）、応答本文には作成した完全なオブジェクトが含まれていて、**version**や**id**などのその他のシステム生成値も含まれています。バージョンおよびIDの値は、その後オブジェクトを変更するためにPUTを使用する場合に必要になるため、特に重要です。応答の一般的な構造と内容については、メソッドの試行と結果の解釈（39ページ）を参照してください。

応答本文には、作成したオブジェクトのURLである**links/self**値も含まれています。たとえば、サンプル オブジェクトの応答本文は次のとおりです。

```
{
  "version": "f6d8da48-7ed5-11e7-9bfd-d96183b5f5f1",
  "name": "new_network_object",
  "description": "A subnet object created using the REST API.",
  "subType": "NETWORK",
  "value": "10.100.10.0/24",
  "isSystemDefined": false,
  "id": "f6d8da49-7ed5-11e7-9bfd-27136f5686ad",
  "type": "networkobject",
  "links": {
    "self": "https://ftd.example.com/api/fdm/[最新（latest）]/object/networks/
f6d8da49-7ed5-11e7-9bfd-27136f5686ad"
  }
}
```

PUT：既存のオブジェクトの変更

既存のオブジェクトの属性を変更するには、PUT メソッドを使用します。たとえば、PUT を使用して、既存のネットワーク オブジェクトに含まれているアドレスを、オブジェクトの ID を変更せずに変更します。

PUT メソッドはオブジェクト全体を置き換えます。単に 1 つの属性を変更することはできません。したがって、保存する古い値が JSON オブジェクトに含まれていることを確認する必要があります。

次の手順では、API エクスプローラで PUT コールを使用する一般的なアプローチを説明します。API クライアントのコード例を使用します。

始める前に

オブジェクトの既存の状態のコピーを取得するには、親リソースに対して GET メソッドを使用します（GET：システムからのデータの取得 （41 ページ）を参照）。

少なくとも以下のパラメータに対する正しい値、および変更しないユーザ指定値が必要です。

- **version**
- **id**

手順

ステップ 1 API エクスプローラで PUT メソッドを開きます（最初に、グループを開いてメソッドとリソースを参照します）。

ステップ 2 [パラメータ（Parameters）] 見出しの下で、以下のオプションを設定します。

- [objId]：オブジェクトの **id** 値。たとえば、900fac69-7d19-11e7-bf7b-d9417b20e59e です。
- **[parentId]**：別のオブジェクト内にあるオブジェクトの場合、このオブジェクトを含む親オブジェクトの ID。たとえば、SSL ルールを変更する場合は、SSL 復号ポリシーの ID。
- **[at]**：番号付きリストでオブジェクトを編成する親に属するオブジェクト（SSL 復号ポリシーなど）の場合は、その場所をオブジェクトに挿入します。整数を使用して場所を示します（リストの先頭は 0）。デフォルトでは、オブジェクトはリストの最後に挿入されます。

（注） {objId} と {parentId} は URL パスの一部ですが、**at** パラメータを URL の末尾の **?** 文字に続けて追加します。

ステップ 3 また、**[パラメータ（Parameters）]** 見出しの下で、**body** パラメータの **[データ型（Data Type）]** > **[サンプル値（Example Value）]** 列に表示される JSON モデルをクリックします。

このボックスをクリックすると、JSON モデルが **body** パラメータの [値（Value）] 列にロードされます。たとえば、PUT /object/networks リソースのボックスをクリックすると、次の本文が

ロードされます。これは、同じリソースに対するPOSTバージョンとは少し異なっていることに注意してください。PUT本文には **version** 属性が含まれています。

```
{
  "version": "string",
  "name": "string",
  "description": "string",
  "subType": "HOST",
  "value": "string",
  "type": "networkobject"
}
```

ステップ4 **body** JSONオブジェクト属性の必須値を入力します。

変更しない古い値を必ず複製してください。

列挙値の場合は、必ず **[応答クラス（Response Class）]** > **[モデル（Model）]** で許可される値を確認してください。オブジェクトを別のサブタイプに変更する場合を除き、古い値を繰り返し使用します。たとえば、ネットワークオブジェクトのデフォルトのPUTモデルでは **subType** はHOSTですが、サブネットオブジェクトを変更する場合は必ず **subType** をNETWORKに変更します。

たとえば、ネットワークオブジェクト内のサブネットIPアドレスを更新するには、**value** を除くすべての属性の古い値をすべて繰り返し使用します。新しいサブネットアドレスを **value** に入力します。

```
{
  "version": "f6d8da48-7ed5-11e7-9bfd-d96183b5f5f1",
  "name": "new_network_object",
  "description": "A subnet object created using the REST API.",
  "subType": "NETWORK",
  "value": "10.100.11.0/24",
  "type": "networkobject",
}
```

ステップ5 [試してみよう!（Try It Out!）] ボタンをクリックし、応答を調べます。

システムの更新に使用される **curl** コマンドを調べます。追加のヘッダーに注意してください。APIクライアントを作成するときは、これらのヘッダーフィールドと値も含める必要があります。たとえば、サンプルオブジェクトを更新するための **curl** コマンドは次のとおりです。Content-Typeヘッダーと Acceptヘッダーに注意してください。

```
curl -X PUT --header 'Content-Type: application/json' \
--header 'Accept: application/json' -d '{ \
   "version": "f6d8da48-7ed5-11e7-9bfd-d96183b5f5f1", \
   "name": "new_network_object", \
   "description": "A subnet object created using the REST API.", \
   "subType": "NETWORK", \
   "value": "10.100.11.0/24", \
   "type": "networkobject" \
 }' 'https://ftd.example.com/api/fdm/[最新（latest）]/object/
networks/f6d8da49-7ed5-11e7-9bfd-27136f5686ad'
```

コールが成功すると（戻りコード 200）、応答本文には更新した完全なオブジェクトが含まれています。バージョン値は変更されますが、オブジェクト ID は（したがって link/self も）同じままです。オブジェクトを変更するたびに、バージョンが変更されます。応答の一般的な構造と内容については、メソッドの試行と結果の解釈 （39 ページ）を参照してください。

（注） オブジェクトに何も変更を加えなかった場合、つまり、更新されるオブジェクトが以前のバージョンと同じである場合、要求は処理されず、そのリソースに対して何も変更されていないことを伝える 204 コードが返されます。

たとえば、サンプル オブジェクト更新の応答本文は次のとおりです。

```
{
  "version": "96f5f3cc-7ede-11e7-9bfd-9b7d8a92863f",
  "name": "new_network_object",
  "description": "A subnet object created using the REST API.",
  "subType": "NETWORK",
  "value": "10.100.11.0/24",
  "isSystemDefined": false,
  "id": "f6d8da49-7ed5-11e7-9bfd-27136f5686ad",
  "type": "networkobject",
  "links": {
    "self": "https://ftd.example.com/api/fdm/[最新（latest）]/object/networks/
f6d8da49-7ed5-11e7-9bfd-27136f5686ad"
  }
}
```

DELETE：ユーザが作成したオブジェクトの削除

自分または別のユーザが作成したオブジェクトを削除するには、DELETE メソッドを使用します。たとえば、不要になったネットワーク オブジェクトを削除するには、DELETE を使用します。

システム定義オブジェクトや存在する必要があるオブジェクトは削除できません。

また、アクセス ルールで使用されているネットワーク オブジェクトなど、別のオブジェクトによって現在使用されているオブジェクトも削除できません。使用中のオブジェクトについては、そのオブジェクトを使用しているすべてのオブジェクトを変更してからそのオブジェクトを削除します。

次の手順では、API エクスプローラで DELETE コールを使用する一般的なアプローチを説明します。API クライアントのコード例を使用します。

始める前に

オブジェクトの既存の状態のコピーを取得するには、親リソースに対して GET メソッドを使用します（GET：システムからのデータの取得 （41 ページ）を参照）。

オブジェクトを削除するには、オブジェクト ID（**id** 値）が必要です。

手順

ステップ 1 APIエクスプローラでDELETEメソッドを開きます（最初に、グループを開いてメソッドとリソースを参照します）。

ステップ 2 [パラメータ（Parameters）] 見出しの下で、オブジェクトの [id] 値を [objId] フィールドに入力します。たとえば、f6d8da49-7ed5-11e7-9bfd-27136f5686ad です。

該当オブジェクトがコンテナ内に存在する場合は、親オブジェクトの ID も **[parentId]** フィールドに入力する必要があります。

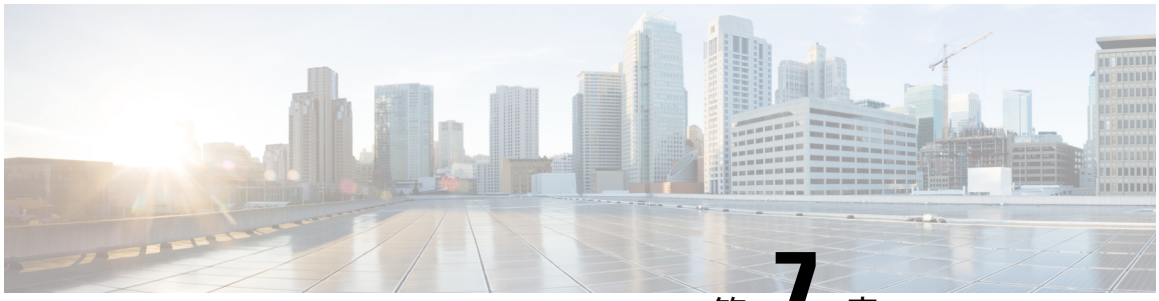
ステップ 3 [試してみよう!（Try It Out!）] ボタンをクリックし、応答を調べます。

システムからオブジェクトを削除するために使用する **curl** コマンドを調べます。追加のヘッダーに注意してください。APIクライアントを作成するときは、これらのヘッダーフィールドと値も含める必要があります。たとえば、サンプルオブジェクトを削除するための **curl** コマンドは以下のとおりです。Accept ヘッダーに注意してください。

```
curl -X DELETE --header 'Accept: application/json'
'https://ftd.example.com/api/fdm/[最新（latest）]/object/
networks/f6d8da49-7ed5-11e7-9bfd-27136f5686ad'
```

コールが成功すると（戻りコード204「コンテンツがありません」）、空の応答本文を受け取ります。これは予期されている結果です。

DELETE：ユーザが作成したオブジェクトの削除



第 7 章

設定変更の導入

設定変更の導入

POST、PUT、およびDELETE の各コールは 脅威に対する防御 デバイスを直接更新しますが、デバイスはすぐにはアクティブになりません。デバイスがトラフィック処理時に新しい設定を使用するためには、設定変更を展開する必要があります。

手順

ステップ 1 Deployment グループで POST /operational/deploy リソースを使用して、展開を開始します。

たとえば、**curl** コマンドは次のようになります。

```
curl -X POST --header 'Content-Type: application/json'
--header 'Accept: application/json'
'https://ftd.example.com/api/fdm/[最新（latest）]/operational/deploy'
```

ステップ 2 応答を評価して、展開ジョブがキューに入ったことを確認します。

正常な応答（ステータス コード 200）は次のようになります。状態に注意してください。

```
{
  "id": "62bf405f-796c-11e8-8640-a9156b92ec49",
  "statusMessage": null,
  "statusMessages": null,
  "modifiedObjects": {},
  "cliErrorMessage": null,
  "queuedTime": 1530036705491,
  "startTime": -1,
  "endTime": -1,
  "state": "QUEUED",
  "name": "User (admin) Triggered Deployment",
  "links": {
    "self": "https://ftd.example.com/api/fdm/[最新（latest）]/operational/deploy/
62bf405f-796c-11e8-8640-a9156b92ec49"
```

```
  }
}
```

（注） **cliErrorMessage** および **name** は API v2 で追加された属性で、v1 の応答には含まれていません。

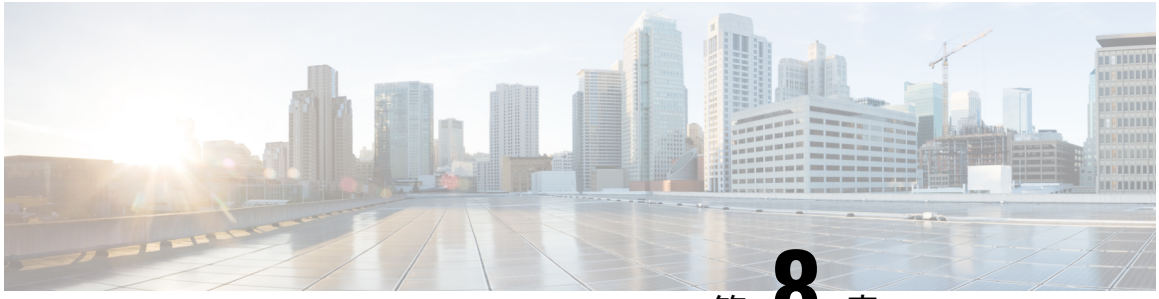
ステップ 3 GET /operational/deploy/{objId} リソースを使用して、ジョブのステータスを確認します。

たとえば、**curl** コマンドは次のようになります。

```
curl -X GET --header 'Accept: application/json'
'https://ftd.example.com/api/fdm/[最新（latest）]/operational/deploy/
a7a227fb-82ab-11e7-8186-0dc471ff0672'
```

応答は次のようになります。状態 DEPLOYED が、ジョブが正常に完了したことを示していることに注意してください。modifiedObjects パラメータは、展開ジョブで変更されたオブジェクトを表示します。この場合、new-network という名前のネットワーク オブジェクトに対する 1 つの変更があります。

```
{
  "id": "62bf405f-796c-11e8-8640-a9156b92ec49",
  "statusMessage": "Deployed Successfully",
  "statusMessages": [
    "Deployed Successfully"
  ],
  "modifiedObjects": {
    "NetworkObject": [
      "new-network"
    ]
  },
  "cliErrorMessage": null,
  "queuedTime": 1530036705491,
  "startTime": 1530036705924,
  "endTime": 1530036822612,
  "state": "DEPLOYED",
  "name": "User (admin) Triggered Deployment",
  "links": {
    "self": "https://ftd.example.com/api/fdm/[最新（latest）]/operational/deploy/
62bf405f-796c-11e8-8640-a9156b92ec49"
  }
}
```



第 8 章

コンフィギュレーションのインポート/エクスポート

バージョンの要件：構成のインポート/エクスポートを使用するには、脅威に対する防御 バージョン 6.5(0) 以降、および 脅威に対する防御 REST API v4 以降を実行している必要があります。

Device Manager で管理されているデバイスから構成をエクスポートし、同じデバイスに、または別の互換性のあるデバイスにインポートすることができます。たとえば、構成のインポート/エクスポートを使用して、類似する複数のデバイスにベースラインの構成を複製し、各デバイスで Device Manager を使用してデバイスごとに固有の特性を設定することができます。

コンフィギュレーションのインポート/エクスポートについて

Device Manager または CDO を使用して 脅威に対する防御 デバイスをローカルで管理する場合は、脅威に対する防御 API を使用してデバイスの構成をエクスポートできます。このメソッドは、Secure Firewall Management Center が管理するデバイスでは機能しません。

構成をエクスポートすると、zip ファイルが作成されます。作成された zip ファイルはワークステーションにダウンロードできます。構成自体は、JSON 形式のテキストファイルで属性と値のペアを使用して定義されたオブジェクトとして表されます。ファイルを編集した後、同じデバイスまたは別のデバイスに再びインポートできます。

そのため、エクスポートファイルを使用してテンプレートを作成し、ネットワーク内の他のデバイスに展開できます。

オブジェクトをインポートする際、構成ファイルでオブジェクトを定義するのではなく、import コマンドで直接オブジェクトを定義することもできます。ただし、オブジェクトを直接定義するのは、少数の変更をインポートする場合に限定してください。

ここでは、構成のインポート/エクスポートについて詳しく説明します。

エクスポートファイルに含まれるもの

エクスポートを実行する場合は、どの構成をエクスポートファイルに含めるかを指定します。完全なエクスポートには、エクスポート zip ファイル内のすべてのものが含まれます。何をエクスポートするかに基づいて、エクスポート zip ファイルには次のものを含める場合があります。

- 設定された各オブジェクトを定義する属性と値のペア。Device Manager で「オブジェクト」と呼ばれるものだけに限らず、設定可能な項目はすべて、オブジェクトとしてモデル化されます。
- リモートアクセス VPN を設定した場合は、AnyConnect パッケージおよびその他の参照ファイル（クライアントプロファイル XML ファイル、DAP XML ファイル、Hostscan パッケージなど）。
- カスタムファイルポリシーを設定した場合は、すべての参照済みクリーンリストまたはカスタム検出リスト。

インポート/エクスポートとバックアップ/復元の比較

構成のインポート/エクスポートは、バックアップ/復元と同じではありません。

- バックアップ/復元は、ディザスタリカバリを目的としています。デバイスにバックアップを復元できるのは、デバイスが同じモデルであり、バックアップの取得元デバイスと同じソフトウェアバージョンを実行している場合のみです。これは主として、「最後に良好だった」構成を同じデバイスに回復すること、または構成を交換用デバイスに復元することを目的としています。
- インポート/エクスポートは、構成の全部または一部を保持することを目的としています。エクスポートファイルを使用して、デバイスのイメージを再作成した後で、構成をデバイスに復元することができます。または、エクスポートファイルをテンプレートとして使用し、その内容を編集してから別のデバイスにインポートすることもできます。インポート/エクスポートを使用すると、新しいデバイスを特定のベースラインの構成にまで迅速に設定できるため、デバイスをネットワークに迅速に導入できます。制限の範囲内で、別のデバイスモデル（たとえば、Firepower 2120 から 2130）にファイルをインポートすることもできます。インポートファイルに、すべてのデバイスモデルでサポートされているオブジェクトのみが含まれている場合、インポートに関する制限はほとんどありません。1 つ制約として、デバイスではエクスポートファイルに使用したものと同じ API バージョンを使用する必要があります。

インポート/エクスポートの戦略

次に、インポート/エクスポートを使用する方法をいくつか示します。

- **新しいデバイス用のテンプレートを作成します。**モデルデバイスを必要な基準設定にしてから、完全な構成をエクスポートします。その後、その構成を新しいデバイスにインポートしてから、Device Manager または 脅威に対する防御 API を使用して必要な変更を加えることができます。また、インポートの前にテンプレートを編集して、各インターフェイスのIPアドレスなどの変更を加えることもできます。完全なエクスポートにはManagementIPオブジェクト（type = managementip）が含まれていることに注意してください。ターゲットデバイスに管理アドレスおよびゲートウェイがすでに設定されている場合は、新しいデバイス用のテンプレートを作成するときに、エクスポートファイルからこのオブジェクトを削除する必要があります。そうしないと、管理アドレッシング情報が上書きされます。
- **あるデバイスから他の同様のデバイスに構成の変更を展開します。**たとえば、デバイスAの構成を編集するときに、いくつかの新しいネットワークオブジェクトとアクセス制御ルールを作成します。次に、保留中の変更をエクスポートし、それらの変更をデバイスBにインポートできます。両方のデバイスに構成を展開すると、同じ新しいルールが実行されます。
- **システムの再イメージ化後に構成を再適用します。**デバイスを再イメージ化すると、構成が消去されます。最初に完全な構成をエクスポートしておけば、再イメージ化の完了後にインポートすることができます。
- **ターゲットの構成を適用します。**エクスポートファイルは編集ができ、手動で作成することもできるため、別のデバイスにインポートするオブジェクトを除くすべてのオブジェクトを削除することができます。たとえば、一連のネットワークオブジェクトを含む構成ファイルを作成し、それを使用して、同じネットワークオブジェクトのグループをすべての 脅威に対する防御 デバイスにインポートすることができます。

設定のインポート/エクスポートのガイドライン

- エクスポートジョブの間は、構成データベースで書き込みロックが保持されます。ジョブが完了するまでは、APIまたはDevice Manager を使用して構成を変更することはできません。ただし、エクスポートジョブ中に、Device Manager で設定を表示したり、API で GET コールを使用したりすることは可能です。
- インポートジョブの間は、構成データベースで読み取りと書き込みの両方のロックが保持されます。ジョブが完了するまでは、APIまたはDevice Manager を使用して構成を表示したり変更を加えたりすることはできません。
- インポートされた構成は、既存の構成に追加されます。デバイスの構成を消去して、インポートした構成に置き換えることはできません。インポートの前にデバイスの構成をリセットする必要がある場合は、デバイスの CLI に移動して、**configure manager delete** コマンドを発行し、その後 **configure manager local** コマンドを発行できます。管理インターフェイスの構成のみが維持されます。
- デバイスにファイルをインポートできるのは、ファイルに含まれているメタデータオブジェクト内の apiVersion 属性で定義されているものと同じ API バージョンをデバイスで実行している場合のみです。

- エクスポートするデバイスとインポートするデバイスの SRU バージョンを同じにする必要があります。異なる場合、インポートは失敗します。

設定のインポートおよびエクスポート

インポート/エクスポートプロセスでは、まずローカル管理対象デバイスから構成がエクスポートされます。その後、エクスポートファイルをダウンロードし、必要に応じて編集を加え、同じデバイスまたは互換性のあるデバイスにアップロードできます。以降のトピックでは、それぞれのステップについて説明します。

設定のエクスポート

設定のエクスポートジョブを作成して開始するには、POST /action/configexport メソッドを使用します。

手順

ステップ 1 エクスポートジョブ用の JSON オブジェクト本体を作成します。

このコールで使用する JSON オブジェクトの例を次に示します。

```
{
  "diskFileName": "string",
  "encryptionKey": "*********",
  "doNotEncrypt": false,
  "configExportType": "FULL_EXPORT",
  "deployedObjectsOnly": true,
  "entityIds": [
    "string"
  ],
  "jobName": "string",
  "type": "scheduleconfigexport"
}
```

その属性は次のとおりです。

- **diskFileName**：（任意）。エクスポート zip ファイルの名前。名前を指定しない場合は、システムによって名前が生成されます。名前を指定した場合でも、一意性を確保するために名前に文字が付加される場合があります。名前の最大長は 60 文字です。
- **encryptionKey**：（任意）。zip ファイルの暗号化キー。ファイルを暗号化しない場合は、このフィールドを省略して、代わりに "doNotEncrypt": true を指定します。キーを指定する場合は、キーをワークステーションにダウンロードした後に、キーを使用して zip ファイルを開く必要があります。エクスポートされた構成ファイルでは、秘密鍵、パスワード、およびその他の機密データがクリアテキストで公開されることに注意してください（他の方法ではインポートできないため）。そのため、場合によっては暗号化キーを適用して機密データを保護する必要があります。システムでは AES 256 暗号化が使用されます。

- **doNotEncrypt**：（任意）。エクスポートファイルを暗号化するか（false）または暗号化しないか（true）。デフォルトは false です。つまり、空でない encryptionKey 属性を指定する必要があります。true を指定した場合、encryptionKey 属性は無視されます。
- **configExportType**：次のいずれかの enum 値。
  - **FULL_EXPORT**：エクスポートファイルに構成全体を含めます。これがデフォルトです。
  - **PARTIAL_EXPORT**：entityIds リストで識別されるオブジェクトとその子孫オブジェクトのみを含めます。エクスポート不可能なオブジェクトは、ID を指定しても含まれません。ユーザー定義オブジェクトはすべてエクスポート可能です。
  - **PENDING_CHANGE_EXPORT**：まだ展開されていない（つまり、保留中の変更を含む）オブジェクトのみを含めます。
- **deployedObjectsOnly**：（任意）。オブジェクトが展開されている場合にのみ、それらのオブジェクトをエクスポートファイルに含めるかどうか。つまり、保留中の変更は含めません。これらのジョブには未展開のオブジェクトしか含まれないため、PENDING_CHANGE_EXPORT ジョブではこの属性は無視されます。デフォルトは false です。これは、すべての保留中の変更がエクスポートに含まれることを意味します。保留中の変更を除外するには、true を指定します。
- **entityIds**：[ブラケット] で囲まれた、一連の開始ポイントオブジェクトの ID をカンマで区切ったリスト。このリストは PARTIAL_EXPORT ジョブでは必須です。このリスト内の各項目には、UUID 値か、または "**id=***uuid-value*"、"**type=***object-type*"、"**name=***object-name*" などのパターンと一致する属性と値のペアのいずれかを指定できます。たとえば、**"type=networkobject"** などを指定できます。

  type は、networkobject などのリーフエンティティ、または一連のリーフタイプのエイリアスのいずれかになります。通常の type エイリアスの例としては、network（NetworkObject と NetworkObjectGroup）、port（すべての TCP/UDP/ICMP ポート、プロトコル、およびグループタイプ）、url（URL オブジェクトおよびグループ）、ikepolicy（IKE V1/V2 ポリシー）、ikeproposal（Ike V1/V2 プロポーザル）、identitysource（すべてのアイデンティティソース）、certificate（すべての証明書タイプ）、object（Device Manager の [オブジェクト（Objects）] ページにリストされるすべてのオブジェクト/グループタイプ）、interface（すべてのネットワーク インターフェイス）、s2svpn（すべてのサイト間 VPN 関連タイプ）、ravpn（すべての RA VPN 関連タイプ）、vpn（s2svpn と ravpn の両方）などがあります。

  これらのオブジェクトとそれらから送出される参照子孫はすべて、PARTIAL_EXPORT の出力ファイルに含まれます。エクスポート不可能なオブジェクトはすべて、ID を指定した場合でも、出力から除外されます。適切なリソースタイプに対して GET メソッドを使用し、ターゲットオブジェクトの UUID、タイプ、または名前を取得します。

  たとえば、すべてのネットワークオブジェクトと、myaccessrule という名前のアクセスルール、および、UUID で識別される 2 つのオブジェクトをエクスポートする場合、次のように指定できます。

```
  "entityIds": [
     "type=networkobject",
     "id=bab3e3cd-8c70-11e9-930a-1f12ee87d473",
     "name=myaccessrule",
     "acc2e3cd-8c70-11e9-930a-1f12ee87b286"
     ]",
```

- **jobName**：（任意）。エクスポートジョブの名前。ジョブの名前を指定すると、ジョブのステータスを取得するときにジョブを見つけやすくなります。
- **type**：ジョブのタイプ。常に **scheduleconfigexport** です。

例：

次の例では、export-config-1 への完全なエクスポートを実行し、他のすべての属性に対してデフォルトを受け入れます。

```
{
  "diskFileName": "export-config-1",
  "doNotEncrypt": true
  "configExportType": "FULL_EXPORT",
  "type": "scheduleconfigexport"
}
```

ステップ 2 オブジェクトをポストします。

たとえば、curl コマンドは次のようになります。

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json'
 -d '{ \
   "configExportType": "FULL_EXPORT", \
   "type": "scheduleconfigexport" \
 }' 'https://10.89.5.38/api/fdm/[最新（latest）]/action/configexport'
```

ステップ 3 応答を確認します。

取得する応答コードは 200 である必要があります。最低限の JSON オブジェクトをポストした場合の正常な応答本文は次のようになります。暗号化キーを指定した場合、暗号化キーは応答でマスクされます。

```
{
  "version": null,
  "scheduleType": "IMMEDIATE",
  "user": "admin",
  "forceOperation": false,
  "jobHistoryUuid": "c7a8ba61-629a-11e9-8b8d-0fcc3c9d6d0b",
  "ipAddress": "10.24.5.177",
  "diskFileName": "export-config-1",
  "encryptionKey": null,
  "doNotEncrypt": true
  "configExportType": "FULL_EXPORT",
  "deployedObjectsOnly": false,
  "entityIds": null,
  "jobName": "Config Export",
  "id": "c79be920-629a-11e9-8b8d-85231be77de0",
  "type": "scheduleconfigexport",
  "links": {
    "self": "https://10.89.5.38/api/fdm/[最新（latest）]
```

```
/action/configexport/c79be920-629a-11e9-8b8d-85231be77de0"
  }
}
```

エクスポートジョブのステータスの確認

エクスポートジョブの完了には多少時間がかかります。構成が大きいほど、ジョブに必要な時間が長くなります。ジョブのステータスをチェックし、ファイルをダウンロードする前に正常に完了していることを確認します。

ステータスを取得するには、GET /jobs/configexportstatus を使用する方法が最も簡単です。たとえば、curl コマンドは次のようになります。

```
curl -X GET --header 'Accept: application/json'
'https://10.89.5.38/api/fdm/[最新（latest）]/jobs/configexportstatus'
```

正常に完了したジョブは次のようなステータスを返します。

```
{
  "version": "hdy62yf5xp3vf",
  "jobName": "Config Export",
  "jobDescription": null,
  "user": "admin",
  "startDateTime": "2019-04-19 13:14:54Z",
  "endDateTime": "2019-04-19 13:14:56Z",
  "status": "SUCCESS",
  "statusMessage": "The configuration was exported successfully",
  "scheduleUuid": "1ef502ad-62a5-11e9-8b8d-074ebc750708",
  "diskFileName": "export-config-1.zip",
  "messages": [],
  "configExportType": "FULL_EXPORT",
  "deployedObjectsOnly": false,
  "entityIds": null,
  "id": "1f0aad8e-62a5-11e9-8b8d-bb1ebb4d1300",
  "type": "configexportjobstatus",
  "links": {
    "self": "https://10.89.5.38/api/fdm/[最新（latest）]
/jobs/configexportstatus/1f0aad8e-62a5-11e9-8b8d-bb1ebb4d1300"
  }
}
```

または、GET /jobs/configexportstatus/{objId} メソッドを使用して特定のジョブのステータスを取得することもできます。応答オブジェクトの **id** フィールドからオブジェクト ID を取得します。

エクスポートファイルのダウンロード

エクスポートジョブが完了すると、エクスポートファイルがシステムディスクに書き込まれ、構成ファイルと呼ばれます。このエクスポートファイルは、GET /action/downloadconfigfile/{objId} メソッドを使用してワークステーションにダウンロードできます。使用可能なファイルのリストを取得するには、GET /action/configfiles メソッドを使用します。

（注） GET /action/downloadconfigfile/{objId} では、通常はオブジェクト ID としてファイル名を指定します。または、ファイルに関連付けられている ConfigExportStatus オブジェクトの ID を指定することもできます。

手順

ステップ 1 ディスク上の構成ファイルのリストを取得します。

構成ファイルのリストには、エクスポートファイルと、インポート用にアップロードしたファイルが含まれます。

curl コマンドは次のようになります。

```
curl -X GET --header 'Accept: application/json'
'https://10.89.5.38/api/fdm/[最新（latest）]/action/configfiles'
```

応答には項目のリストが表示され、これらはそれぞれが構成ファイルです。たとえば、次のリストは 2 つのファイルを示しています。すべてのファイルの **id** が default であることに注意してください。ID を無視し、代わりに **diskFileName** を使用します。

```
{
  "items": [
    {
      "diskFileName": "export-config-2.zip",
      "dateModified": "2019-04-19 13:32:28Z",
      "sizeBytes": 10182,
      "id": "default",
      "type": "configimportexportfileinfo",
      "links": {
        "self": "https://10.89.5.38/api/fdm/[最新（latest）]/action/configfiles/default"
      }
    },
    {
      "diskFileName": "export-config-1.zip",
      "dateModified": "2019-04-19 13:14:56Z",
      "sizeBytes": 10083,
      "id": "default",
      "type": "configimportexportfileinfo",
      "links": {
        "self": "https://10.89.5.38/api/fdm/[最新（latest）]/action/configfiles/default"
      }
    }
  ],
```

ステップ 2 diskFileName をオブジェクト ID として使用し、ファイルをダウンロードします。

curl コマンドは次のようになります。

```
curl -X GET --header 'Accept: application/octet-stream'
'https://10.89.5.38/api/fdm/[最新（latest）]/action/downloadconfigfile/export-config-2.zip'
```

ファイルは、デフォルトのダウンロードフォルダにダウンロードされます。APIエクスプローラからGETメソッドを発行していて、ダウンロード場所を指定するよう求めるようにブラウザが設定されている場合は、ファイルを保存するよう求められます。

正常にダウンロードされると、戻りコードが200となり、応答本文はなくなります。

エクスポートした構成ファイルの編集

構成ファイルをダウンロードした後、ファイルを解凍して、オブジェクトが含まれているテキストファイルを開くことができます。ワードパッドはメモ帳よりも読みやすい形式で内容を表示します。インストールしている他のテキストエディタを使用することもできます。独自の構成ファイルを最初から作成することもできますが、ファイル構造を理解するために構成をエクスポートする必要があります。

次のトピックでは、テキストファイルの要件について説明します。

構成ファイルの最低要件

構成ファイルには、次の最低限の要素が含まれている必要があります。

- ファイル内のオブジェクトは[ブラケット]で囲みます。ファイル全体がオブジェクトの配列になっていて、標準のJSON表記が使用されています。
- 各オブジェクトは{波カッコ}で囲みます。
- 構成ファイル内のオブジェクトを区切るには、カンマを使用します。つまり、最後のオブジェクトを除いて、オブジェクトの閉じ波カッコの後にはカンマを付ける必要があります。
- ファイル内の最初のオブジェクトはメタデータオブジェクトである必要があります。正しいオブジェクト属性を取得するには、目的のモデルのデバイスから構成をエクスポートするのが最も簡単です。たとえば、次に示すのはSecure Firewall Threat Defense Virtualデバイスから取得したメタデータオブジェクトです。デバイスをインポートする前に、構成タイプとエクスポートタイプを編集できます。必要に応じて、generatedOn属性を削除できます。

```
{"hardwareModel":"Cisco Firepower Threat Defense for VMWare",
 "type":"metadata",
 "configType":"FULL_CONFIG",
 "apiVersion":"[最新（latest）]",
 "generatedOn":"Fri Apr 19 13:32:28 UTC 2019",
 "exportType":"FULL_EXPORT",
 "softwareVersion":"6.5.0-10480"}
```

- メタデータオブジェクトでは、正しい構成タイプ（configType）値を指定する必要があります。
  - FULL_CONFIG：このテキストファイルにはデバイスの完全な構成が含まれています。

- DELTA_CONFIG：このテキストファイルには部分的な構成が含まれています。数個のオブジェクトしか含まれていない場合もあります。

- exportType は、FULL_EXPORT、PARTIAL_EXPORT、PENDING_CHANGE_EXPORT のいずれかです。
- 完全な構成をインポートする場合、メタデータオブジェクトでは、hardwareModel、softwareVersion、apiVersion の各属性を指定する必要があります。
- オブジェクトは1行または複数行に記述できますが、オブジェクト内の属性間には空の行やコメント行を入れないでください。ファイル内ではコメントは使用できません。
- オブジェクトは依存関係の順序でエクスポートされますが（別のオブジェクトによって参照されるオブジェクトが先に定義されます）、インポートの構成ファイルでその順序を維持する必要はありません。オブジェクト間の関係は、オブジェクトの名前と ID が依存する側のオブジェクト間で正しく解決されれば、インポート時に自動的に解決されます。

アイデンティティ ラッパー オブジェクトの基本構造

構成ファイルでは、アイデンティティ ラッパー オブジェクトを使用して、エクスポートまたはインポートが可能な ConfigEntity または ManagementEntity オブジェクトを定義します。次に、アイデンティティ ラッパー オブジェクトの基本構造を示します。

```
{
    "type" : "identitywrapper",
    "data" : {},
    "parentName" : "container-name",
    "oldName" : "old-object-name",
    "action" : "EDIT", //Enum values: CREATE, EDIT or DELETE
    "index" : integer,
  }
```

オブジェクトには次の属性が含まれています。

- **type**：これは常に **identitywrapper** です。
- **data**これは、ネットワークオブジェクトやアクセス制御ルールなど、構成から取得したオブジェクトを定義する属性と値のペアの集合です。このコレクションに必要な属性は、特定のオブジェクトタイプに対応するモデルと、実行しようとするアクションによって異なります。属性と値のペアは {波カッコ} で囲みます。データ配列内の属性はカンマで区切ります。
- **parentName**：（必要な場合）。限られた数のオブジェクトが ContainedObjects です。ContainedObject は、自身を含むオブジェクトとの関係を持ちます。たとえば、アクセスルール、手動 NAT ルール、サブインターフェイスなどがあります。これらの項目について、parentName は、その項目が含まれているオブジェクト（親）の名前を指定します。この属性は、含まれている側のオブジェクトに対して指定します。含まれている側のオブジェクト以外のオブジェクトには指定しないでください。また、場合によってはこれらのオブジェクトの指数を指定する必要もあります。

親が AccessPolicy などの単一のオブジェクトである（つまり、複数のオブジェクトを作成できない）場合は、実際にこの属性を省略でき、参照はシステムによって解決されます。

- **oldName**：（必要な場合）。既存のオブジェクトの名前を変更する場合は、この属性で古い名前を指定し、data 属性の **name** 属性に新しい名前を指定できます。この属性を使用するには、action を EDIT にする必要があります。
- **action**：定義されたオブジェクトに関連して実行するアクション。完全なエクスポートでは、アクションは常に **CREATE** になります。保留中の変更または部分的なエクスポートの場合は、他のアクションが **EDIT** または **DELETE** になる場合があります。

  インポート用のファイルを編集する場合は、目的のアクションを指定します。CREATE を指定したもののオブジェクトがすでに存在する場合は、アクションが EDIT に変更されます。オブジェクトが存在しない場合は、EDIT が CREATE に変更されます。DELETE アクションは変更されません。オブジェクト参照は、オブジェクトのタイプと名前、またはオブジェクトのタイプと古い名前、あるいはオブジェクトのタイプと親の名前に基づいて解決されます。

  - CREATE：これは新しいオブジェクトです。オブジェクトを POST するときに必要なデータ属性を指定する必要があります。**name** が、指定したタイプの既存のオブジェクトと一致する場合、アクションは自動的に EDIT に変更されます。

    新しいオブジェクトを作成して他のオブジェクトからそのオブジェクトを参照する場合（ネットワークオブジェクトを定義してからアクセスルールで使用する場合など）、オブジェクトの **name** が正確に参照されている必要があります。
  - EDIT：オブジェクトを更新しようとしています。オブジェクトを PUT するときに必要なデータ属性（バージョンと ID を除きます）を指定する必要があります。名前とオブジェクトタイプは更新するオブジェクトの決定に使用され、バージョン属性は常に無視されます。
  - DELETE：オブジェクトを削除しようとしています。**type** および **name** 属性をオブジェクトデータで指定する必要があります。

- **index**：（任意。整数）。アクセス制御ルールや手動 NAT ルールなどの順序付きリストの一部となっているオブジェクトの場合における、ポリシー内のオブジェクトの位置。新しいルールを作成する場合、指数値を指定しなければ、ルールは最後のルールとしてポリシーの末尾に追加されます。ルールを編集する場合は、ルールの既存の位置が保持されます。

例：別のデバイスにインポートするためのネットワークオブジェクトの編集

各オブジェクトの構成は次のようになっています。この例は、syslog サーバーの IP アドレスを定義するネットワーク ホスト オブジェクトです。

```
{"type":"identitywrapper",
 "action":"CREATE",
 "data":{
    "version":"lfxdbtbyg4ex6",
    "name":"syslog-host",
```

```
    "subType":"HOST",
    "value":"10.100.10.10",
    "isSystemDefined":false,
    "dnsResolution":"IPV4_AND_IPV6",
    "id":"2cd0ea03-62a7-11e9-8b8d-dbf377c781d8",
    "type":"networkobject"}}
```

デバイスからこのオブジェクトをエクスポートし、そのオブジェクトを別のデバイスにインポートすることを考えます。ただし、新しいデバイスでは別のアドレス（192.168.5.15）にあるsyslogサーバーを使用する必要があるとします。新しいオブジェクトを作成しようとしているので、data属性から**version**属性と**id**属性を削除します。また、**isSystemDefined**（デフォルトはfalse）と**dnsResolution**（FQDNオブジェクトの場合にのみ該当）を削除することもできます。結果として得られる新しいオブジェクトは次のようになります。

```
{"type":"identitywrapper",
 "action":"CREATE",
 "data":{
    "name":"syslog-host",
    "subType":"HOST",
    "value":"192.168.5.15",
    "type":"networkobject"}}
```

ファイルの上部で、メタデータオブジェクトを保持（または追加）する必要があります。また、行の戻り値を追加して、ファイルの内容をスキャンして確認しやすくすることもできます。以上により、完成した構成ファイルは次のようになります。

```
[
{"hardwareModel":"Cisco Firepower Threat Defense for VMWare",
 "type":"metadata",
 "configType":"DELTA_CONFIG",
 "apiVersion":"[最新（latest）]",
 "exportType":"PARTIAL_EXPORT",
 "softwareVersion":"6.5.0-10465"}
,
{"type":"identitywrapper",
 "action":"CREATE",
 "data":{
    "name":"syslog-host",
    "subType":"HOST",
    "value":"192.168.5.15",
    "type":"networkobject"}}
]
```

インポートファイルのアップロード

構成ファイルをデバイスにインポートするには、最初にそのファイルをデバイスにアップロードしておく必要があります。zipファイルまたはテキストファイルのいずれかをアップロードできます。zipファイルを使用する場合は、AnyConnectパッケージとクライアントプロファイルを含めることができます。

ファイルをアップロードするにはPOST /action/uploadconfigfileリソースを使用します。名前の最大長は60文字です。

- API エクスプローラからこのメソッドを使用する場合は、**fileToUpload** 属性の横にある **[ファイルの選択（Choose File）]** ボタンをクリックし、ワークステーションドライブからファイルを選択します。
- 独自のプログラムからメソッドを使用する場合、要求ペイロードには、file-name フィールドを含む単一の file-item を含める必要があります。ファイル名拡張子は .txt または .zip のいずれかにする必要があり、実際のファイル内容の形式とファイル拡張子に合わせる必要があります。

curl コマンドは次のようになります。

```
curl -F 'fileToUpload=@./import-1.txt'
'https://10.89.5.38/api/fdm/[最新（latest）]/action/uploadconfigfile'
```

正常に転送されると、戻りコード 200 が返され、応答本文は次のようになります。この例には、インポートジョブに必要となる 脅威に対する防御 システムのファイル名（**diskfilename**）を示しています。

```
{
  "diskFileName": "import-1.txt",
  "dateModified": "2019-04-22 10:18:12Z",
  "sizeBytes": 267,
  "id": "default",
  "type": "configimportexportfileinfo",
  "links": {
    "self": "https://10.89.5.38/api/fdm/[最新（latest）]/action/uploadconfigfile/default"

  }
}
```

設定のインポートとジョブステータスの確認

構成ファイルを 脅威に対する防御 システムにアップロードした後、構成ファイルで定義されているオブジェクトを 脅威に対する防御 の構成にインポートできます。POST /action/configimport メソッドを使用します。

オブジェクトをインポートする際、構成ファイルでオブジェクトを定義するのではなく、import コマンドで直接オブジェクトを定義することもできます。ただし、オブジェクトを直接定義するのは、1 ～ 2 個のネットワークオブジェクトの場合など、少数の変更をインポートする場合に限定してください。

手順

ステップ 1 インポートジョブ用の JSON オブジェクト本体を作成します。

このコールで使用する JSON オブジェクトの例を次に示します。

```
{
  "diskFileName": "string",
  "encryptionKey": "*********",
```

```
  "preserveConfigFile": true,
  "autoDeploy": true,
  "allowPendingChange": true,
  "excludeEntities": [
     "string"
     ]",
  "inputEntities": [
    {
      "action": "CREATE",
      "oldName": "string",
      "parentId": "string",
      "parentName": "string",
      "index": 0,
      "data": {
        "version": "string",
        "id": "string",
        "type": "identity"
      },
      "id": "string",
      "type": "IdEntityWrapper"
    }
  ],
  "jobName": "string",
  "type": "scheduleconfigimport"
}
```

その属性は次のとおりです。

- **diskFileName**：インポートする構成の zip または txt ファイルの名前。
- **encryptionKey**：zip ファイルの暗号化に使用するキー（存在する場合）。構成ファイルが暗号化されていない場合は、キーを指定しないでください。
- **preserveConfigFile**：（任意）。インポートジョブが正常に完了した後に、脅威に対する防御ディスクにインポートされた構成ファイルのコピーを保持するかどうか。ファイルを保持する場合は true を指定し、ファイルを 脅威に対する防御 ディスクから削除する場合は false を指定します。デフォルトは False です。
- **autoDeploy**：（任意）。インポートが成功した場合に展開ジョブを自動的に開始するかどうか。インポートされたオブジェクトは保留中の変更であり、変更を正常に展開するまではアクティブになりません。展開ジョブを自動的に開始するには、true を指定します。false を指定した場合は、手動で変更を展開する必要があります。デフォルトは False です。
- **allowPendingChange**：（任意）。既存の保留中の変更がある場合にインポートジョブの開始を許可するかどうかを指定します。この属性を true に設定し、autoDeploy を true に設定した場合、自動展開ジョブには、既存とインポート済みの両方の変更がすべて含まれます。この属性を false に設定すると、保留中の変更がある場合にインポートジョブは実行されません。デフォルトは False です。
- **excludeEntities**：（任意）。インポートしないオブジェクトを識別する文字列に一致するオブジェクトのリスト。インポートする必要がない項目がインポートファイルに含まれている場合（つまり、アップロードしたファイルからそれらのオブジェクトを削除しない場合）にのみ、この属性を指定する必要があります。このリスト内の各項目は、"**id**=*uuid-value*"、"**type**=*object-type*"、"**name**=*object-name*" などのパターンを持ちます。これらのパターンのいずれかに一致する入力オブジェクトが、インポートから除外されます。

type は、networkobject などのリーフエンティティ、または一連のリーフタイプのエイリアスのいずれかになります。通常の type エイリアスの例としては、network（NetworkObject と NetworkObjectGroup）、port（すべての TCP/UDP/ICMP ポート、プロトコル、およびグループタイプ）、url（URL オブジェクトおよびグループ）、ikepolicy（IKE V1/V2 ポリシー）、ikeproposal（Ike V1/V2 プロポーザル）、identitysource（すべてのアイデンティティソース）、certificate（すべての証明書タイプ）、object（Device Manager の [オブジェクト（Objects）] ページにリストされるすべてのオブジェクト/グループタイプ）、interface（すべてのネットワーク インターフェイス）、s2svpn（すべてのサイト間 VPN 関連タイプ）、ravpn（すべての RA VPN 関連タイプ）、vpn（s2svpn と ravpn の両方）などがあります。

たとえば、すべてのネットワークオブジェクト、および、名前 myobj と UUID で識別される他の 2 つのオブジェクトをインポートから除外するには、次のように指定します。

```
"excludeEntities": [
   "type=networkobject",
   "name=myobj",
   "id=acc2e3cd-8c70-11e9-930a-1f12ee87b286"
   ]",
```

- **inputEntities**：インポートするオブジェクトの数が少ない場合は、それらを構成ファイルで定義するのではなく inputEntities オブジェクトリストで定義できます。この属性を使用する場合は、diskFileName 属性を含めることはできません。または、この属性を null に設定する必要があります。
- **jobName**：（任意）。エクスポートジョブの名前。ジョブの名前を指定すると、ジョブのステータスを取得するときにジョブを見つけやすくなります。
- **type**：ジョブのタイプ。常に **scheduleconfigimport** です。

例：

次に、import-1.txt という名前の構成ファイルをインポートする例を示します。

```
{
  "diskFileName": "import-2.txt",
  "preserveConfigFile": true,
  "autoDeploy": true,
  "allowPendingChange": true,
  "type": "scheduleconfigimport"
}
```

ステップ 2 オブジェクトをポストします。

たとえば、curl コマンドは次のようになります。

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json'
 -d '{ \
   "diskFileName": "import-2.txt", \
   "preserveConfigFile": true, \
   "autoDeploy": true, \
   "allowPendingChange": true, \
   "type": "scheduleconfigimport" \
 }' 'https://10.89.5.38/api/fdm/[最新（latest）]/action/configimport'
```

ステップ 3 応答を確認します。

取得する応答コードは 200 である必要があります。最低限の JSON オブジェクトをポストした場合の正常な応答本文は次のようになります。暗号化キーを指定した場合、暗号化キーは応答でマスクされます。

```
{
  "version": null,
  "scheduleType": "IMMEDIATE",
  "user": "admin",
  "forceOperation": false,
  "jobHistoryUuid": "7e360139-6725-11e9-abb5-078014531401",
  "ipAddress": "10.24.127.37",
  "diskFileName": "import-2.txt",
  "encryptionKey": null,
  "preserveConfigFile": true,
  "autoDeploy": true,
  "allowPendingChange": true,
  "jobName": "Config Import",
  "id": "7e2b52d8-6725-11e9-abb5-5dec35337506",
  "type": "scheduleconfigimport",
  "links": {
    "self": "https://10.89.5.38/api/fdm/[最新（latest）]
/action/configimport/7e2b52d8-6725-11e9-abb5-5dec35337506"
  }
}
```

ステップ 4 インポートジョブのステータスを確認するには、GET /jobs/configimportstatus を使用します。

または、GET /jobs/configimportstatus/{objId} を使用して、1 つのインポートジョブのステータスを取得することもできます。objId については、応答本文の jobHistoryUuid 値を POST /action/configimport コールで使用します。

curl コマンドは次のようになります。

```
curl -X GET --header 'Accept: application/json'
'https://10.89.5.38/api/fdm/[最新（latest）]/jobs/configimportstatus'
```

インポートに成功すると、応答本文は次のようになります。インポートに失敗した場合は、必要に応じてファイルを編集して形式や内容のエラーを修正してからやり直す必要があります。

```
{
      "version": "pcgccfnk4hmiz",
      "jobName": "Config Import",
      "jobDescription": null,
      "user": "admin",
      "startDateTime": "2019-04-25 06:43:54Z",
      "endDateTime": "2019-04-25 06:44:01Z",
      "status": "SUCCESS",
      "statusMessage": "The configuration was imported successfully",
      "scheduleUuid": "7e2b52d8-6725-11e9-abb5-5dec35337506",
      "diskFileName": "import-2.txt",
      "messages": [],
      "preserveConfigFile": true,
      "autoDeploy": true,
      "allowPendingChange": true,
      "id": "7e360139-6725-11e9-abb5-078014531401",
      "type": "configimportjobstatus",
      "links": {
```

```
        "self": "https://10.89.5.38/api/fdm/[最新（latest）]
/jobs/configimportstatus/7e360139-6725-11e9-abb5-078014531401"
  }
}
```

次のタスク

autoDeploy を false に設定した場合は、インポートした変更を組み込むために展開ジョブを実行する必要があります。POST /operational/deploy メソッドを使用します。true に設定する場合は、すでに構成が正常に展開されている必要があります。Device Manager または API（GET /operational/auditevents）で、監査ログを確認できます。展開ジョブは「Post Configuration Import Deployment」という名前です。

（注） 機能によっては特定のライセンスが必要です。たとえば、デバイスにはすべてのリモートアクセス VPN 機能用のライセンスが必要です。ただし、インポートプロセスではライセンスは検証されません。そのため、ライセンスにより制御される機能のオブジェクトを、必要なライセンスを持たないデバイスにインポートすると、展開ジョブは失敗します。この問題が発生した場合は、必要なライセンスをデバイスに割り当てるか、オブジェクトを削除してください。

不要なインポート/エクスポートファイルの削除

エクスポートジョブによって作成された構成ファイルや、構成のインポート用にアップロードした構成ファイルが不要になった場合は、ファイルを削除できます。

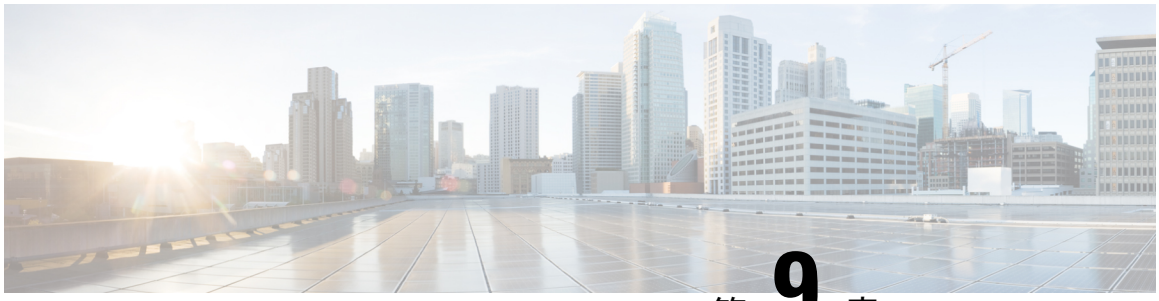
DELETE /action/configfiles/{objId} を使用し、objId 値としてファイル名を指定します。

たとえば、export-config-2.zip という名前のファイルを削除する curl コマンドは次のようになります。

```
curl -X DELETE --header 'Accept: application/json'
'https://10.89.5.38/api/fdm/[最新（latest）]/action/configfiles/export-config-2.zip'
```

成功すると、戻りコードが 204 となり、応答本文はありません。

GET /action/configfiles を使用して、ファイルが削除されたことを確認できます。



第 9 章

詳細および例について

詳細および例について

API の使用方法に関する追加情報は、次のサイトから入手できます。

- https://developer.cisco.com/site/ftd-api-reference/

  このサイトには、Bash コールや Python コードの例などのリソースの参照情報が含まれています。使用している API のバージョンを選択するためのメニューがあります。正しい参照情報を表示するには、適切なバージョンを選択してください。すべての一意のエラーコードと、API を使用すると表示されるメッセージのリストもあります。

- https://developer.cisco.com/docs/firepower/threat-defense/

  このサイトには、高可用性などの選択機能を設定するためのエンドツーエンドの例（コードの例を含む）があります。

- https://developer.cisco.com/firepower/threat-defense/

  このサイトには、API の使用方法の学習に役立つビデオ、学習モジュール、およびラボが含まれています。

詳細および例について

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。