Guide API REST Cisco Cisco Secure Firewall Threat Defense

Dernière modification : 2025-04-24

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

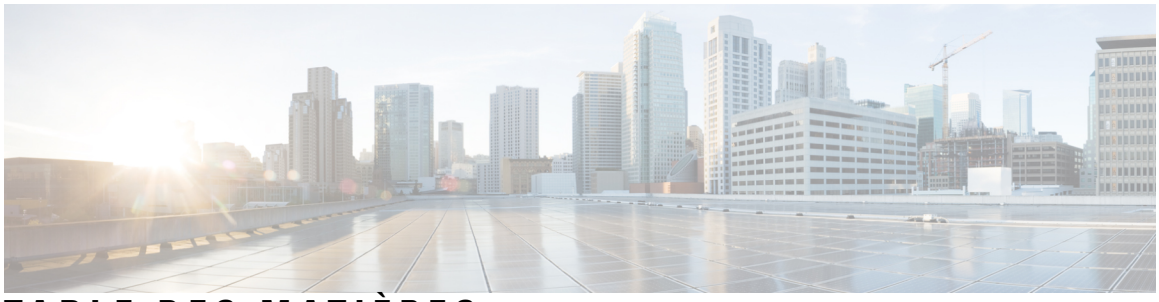
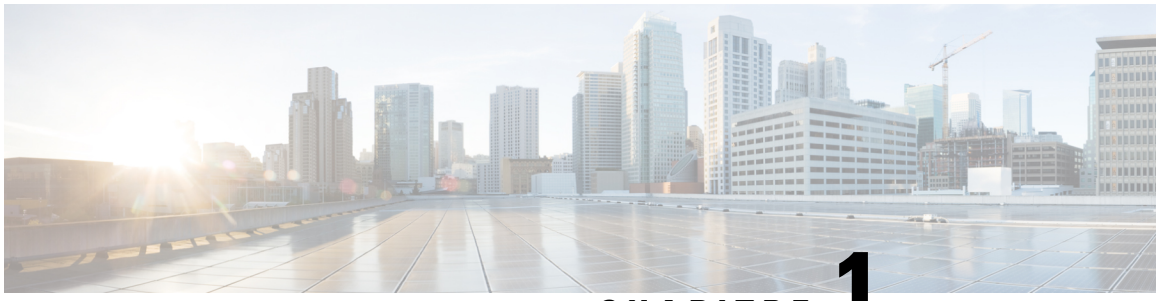


TABLE DES MATIÈRES



CHAPITRE 1

À propos de l'API REST Cisco Secure Firewall Threat Defense

Vous pouvez utiliser l'interface de programmation d'applications (API) de transfert d'état de représentation (REST) Cisco Secure Firewall Threat Defense, par l'entremise du protocole HTTPS, pour interagir avec le dispositif Défense contre les menaces par le biais d'un programme client. L'API REST utilise le format JSON (JavaScript Object Notation) pour représenter les objets.

Cisco Secure Firewall device manager comprend un explorateur d'interface de protocole d'application qui explique toutes les ressources et les objets JSON disponibles à être utilisés en programmation. L'explorateur fournit des informations détaillées sur les paires attribut-valeur dans chaque objet, et vous pouvez tester les différentes méthodes HTTP pour vous assurer de bien comprendre le codage requis pour utiliser chaque ressource. L'explorateur d'interface de protocole d'application fournit également des exemples des URL requises pour chaque ressource.

Vous pouvez également trouver des informations de référence et des exemples en ligne sur le site https://developer.cisco.com/site/ftd-api-reference/.

L'API a son propre numéro de version. Il n'est pas garanti qu'un client conçu pour une version de l'API fonctionnera pour une version future sans erreur ou sans nécessiter de modifications à votre programme.

Public cible de ce guide de programmation

Ce guide a été écrit avec la présomption que vous avez une connaissance générale de la programmation et une compréhension précise des API REST et de JSON. Si vous débutez dans l'utilisation de ces technologies, veuillez d'abord lire un guide général sur les API REST.

Méthodes HTTP prises en charge

Vous ne pouvez utiliser que les méthodes HTTP suivantes : les autres méthodes ne sont pas prises en charge.

- GET : pour lire les données du système.
- POST : pour créer des objets.
- PUT : pour modifier les objets existants. Lorsque vous utilisez PUT, vous devez inclure l'objet JSON entier. Vous ne pouvez pas mettre à jour de manière sélective les attributs individuels d'un objet.
- DELETE : pour supprimer un objet défini par l'utilisateur.

L'URL de base pour l'API

La façon la plus simple de déterminer l'URL de base pour un dispositif Défense contre les menaces donné est d'essayer une méthode GET dans l'explorateur d'interface de protocole d'application et de supprimer simplement la partie objet de l'URL du résultat.

Par exemple, vous pouvez appeler une méthode GET /object/networks et voir quelque chose de similaire à ce qui suit dans la sortie renvoyée sous Request URL (URL de demande) :

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

La partie du nom du serveur de l'URL est le nom d'hôte ou l'adresse IP du dispositif Défense contre les menaces et sera différente pour votre dispositif à la place de « ftd.example.com ». Dans cet exemple, vous devez supprimer /object/networks du chemin pour obtenir l'URL de base :

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

Tous les appels de ressource utilisent cette URL comme base pour l'URL de demande.

Si vous avez modifié le port de données HTTPS, vous devez inclure le port personnalisé dans l'URL. Par exemple, si vous avez changé le port en 4443 : https://ftd.example.com:4443/api/fdm/v1/

Le « v » dans l'URL représente la version de l'API, et cela change généralement avec la version du logiciel. Par exemple, la version de l'API pour Défense contre les menaces, version 6.3.0, est v2, donc l'URL de base serait :

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

Remarque À partir de la version 6.4 de Défense contre les menaces, vous pouvez éviter d'avoir à mettre à jour le chemin dans vos appels d'API en utilisant **latest** (dernier) au lieu de l'élément v dans le chemin. Par exemple, https://ftd.example.com/api/fdm/latest/. L'alias **latest** (dernier) représente la dernière version d'API prise en charge par le dispositif.

Dans l'explorateur d'interface de protocole d'application, si vous faites défiler la page vers le bas de la page, vous pouvez voir des informations sur l'URL de base (sans le nom du serveur) et la version de l'API.

Sécurisation des communications SSL/TLS pour l'API REST

Les appareils Défense contre les menaces sont livrés avec un certificat autosigné pour que vous puissiez l'utiliser pour initier des communications HTTPS. Cependant, comme le certificat n'est pas signé par une autorité de certification (AC) connue, toute tentative d'accès par SSL/TLS fera en sorte que la connexion sera considérée comme non sécurisée.

Au moment de la connexion à un navigateur, vous serez invité à accepter le certificat autosigné, mais une commande comme la commande « curl » rejettera le certificat. Dans le cas de la commande « curl », vous pouvez contourner l'échec de la vérification du certificat en ajoutant le mot-clé **--insecure**. Par exemple :

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

L'une des premières choses que vous devez faire est d'obtenir un certificat de dispositif signé par une autorité de certification pour le dispositif Défense contre les menaces. Ensuite, à l'aide du gestionnaire d'appareil ou de l'API, attribuez ce certificat comme certificat de gestion. La vérification du certificat SSL/TLS ne devrait alors pas échouer et vous n'aurez pas à utiliser des communications non sécurisées dans vos appels d'API.

Procédure

Étape 1 Téléversez le certificat de dispositif signé par une autorité de certification à l'aide de la ressource **POST /objet/internalcertificates**.

Étape 2 Faites de ce certificat le certificat de gestion à l'aide de la ressource **PUT /devicesettings/default/webuicertificates/{objId}**.

Utilisez la ressource **GET /devicesettings/default/webuicertificates** pour déterminer l'ID d'objet du certificat d'interface utilisateur Web.

Étape 3 Déployez les modifications à l'aide de la ressource **POST /operational/deploy**.

Déterminer les versions de l'API prises en charge

Vous pouvez déterminer quelles versions de l'API sont prises en charge sur un dispositif à l'aide de la méthode GET /api/versions (ApiVersions). Cette méthode ne nécessite pas d'authentification et n'inclut pas non plus d'élément indiquant la version dans le chemin. Par exemple :

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

Remplacez « ftd.example.com » par le nom d'hôte ou l'adresse IP du dispositif défense contre les menaces.

Cette méthode renvoie une liste de versions d'API que vous pouvez utiliser. Par exemple :

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

Les chaînes de version sont les mêmes que celles que vous utilisez dans l'URL pour les appels d'API ultérieurs. Si vous utilisez **latest** (dernier) au lieu de l'identifiant de version spécifique, vous pouvez éviter d'avoir à mettre à jour vos appels pour les versions ultérieures. Cependant, utiliser cette technique ne permet pas de tenir compte des modifications des modèles d'objet utilisés dans vos appels, qui pourraient devoir être adaptés d'une version à l'autre.

En règle générale, votre prochaine étape consistera à obtenir un jeton d'accès, comme décrit dans Authentifier votre client API REST à l'aide d'OAuth, à la page 13.

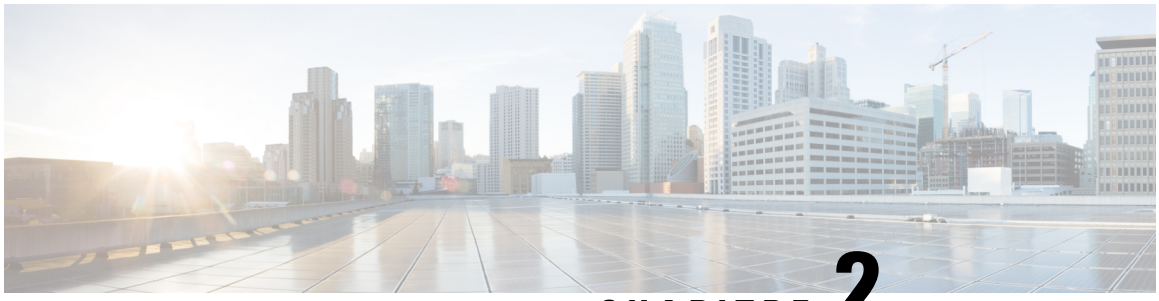
Rétrocompatibilité des versions de l'API

La version de l'API défense contre les menaceschange avec chaque version majeure du logiciel défense contre les menaces. Les nouvelles fonctionnalités ont une incidence sur les appels d'API pour les fonctionnalités en cours d'ajout ou de modification.

Cependant, de nombreuses fonctionnalités ne changent pas d'une version à l'autre. Par exemple, les API liées aux objets réseau et port restent souvent inchangées dans une nouvelle version.

À partir de la version 6.7 de défense contre les menaces, si un modèle de ressource d'API pour une fonctionnalité ne change pas entre les versions, l'API défense contre les menacespeut accepter les appels basés sur l'ancienne version de l'API. Même si le modèle de fonctionnalité a changé, s'il existe un moyen logique de convertir l'ancien modèle au nouveau modèle, l'ancien modèle peut fonctionner. Par exemple, un appel v5 peut être accepté sur un système v6. Si vous utilisez « latest » (dernier) comme numéro de version dans vos appels, ces appels « anciens » sont interprétés comme un appel v6 dans ce scénario. Par conséquent, votre utilisation de la compatibilité ascendante dépend de la façon dont vous structurez vos appels d'API.

Si un modèle de fonctionnalité a changé entre les versions d'API d'une manière qui empêche la rétrocompatibilité, vous recevrez un message d'erreur et vous devrez vérifier ces erreurs et mettre à jour votre code pour ces appels en particulier.



CHAPITRE 2

L'explorateur d'interface de protocole d'application

Utilisez l'explorateur d'interface de protocole d'application pour en apprendre davantage sur l'API REST. Vous pouvez également tester les différentes méthodes et ressources pour vérifier que vous les configurez correctement. Vous pouvez copier et coller les modèles JSON dans votre code comme point de départ.

Astuces Le but de l'explorateur d'interface de protocole d'application est de vous aider à vous familiariser avec l'API. Pour tester des appels dans l'explorateur d'interface de protocole d'application, il faut créer des verrouillages d'accès qui pourraient interférer avec le fonctionnement normal de l'API. Nous vous recommandons d'utiliser l'explorateur d'interface de protocole d'application sur un dispositif hors production.

Ouvrir l'explorateur d'interface de protocole d'application

L'explorateur d'interface de protocole d'application explique toutes les ressources et les objets JSON disponibles à être utilisés en programmation. L'explorateur fournit des informations détaillées sur les paires attribut-valeur dans chaque objet, et vous pouvez tester les différentes méthodes HTTP pour vous assurer de bien comprendre le codage requis pour utiliser chaque ressource.

Procédure

Étape 1 À l'aide d'un navigateur, ouvrez la page d'accueil du système, par exemple, https://ftd.example.com.

Étape 2 Connectez-vous au gestionnaire d'appareil.

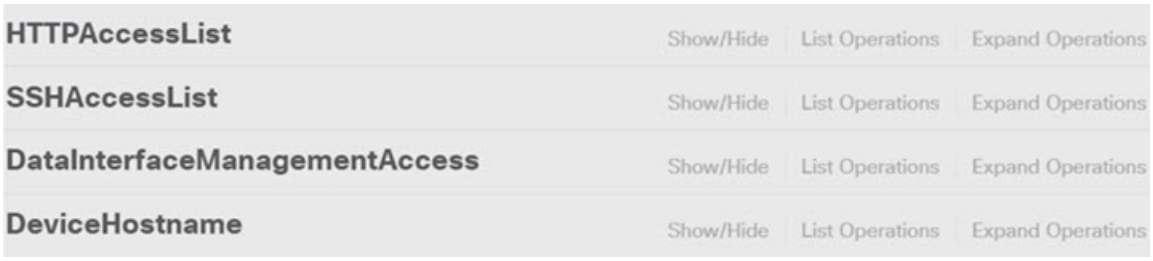
Étape 3 (6.4 et versions antérieures.) Modifiez l'URL pour qu'elle pointe vers /#/api-explorer, par exemple, https://ftd.example.com/#/api-explorer.

Étape 4 (6.5 et versions ultérieures.) Cliquez sur le bouton des autres options (⋮) et choisissez **API Explorer** (Explorateur d'interface de protocole d'application).

Le système ouvre l'explorateur d'interface de protocole d'application dans un onglet ou une fenêtre distincte, en fonction des paramètres de votre navigateur.

S'y retrouver dans l'explorateur d'interface de protocole d'application

Lorsque vous entrez dans l'explorateur d'interface de protocole d'application, une liste de groupes de ressources s'affiche. Ces groupes incluent les ressources disponibles dans l'API. L'illustration suivante montre un petit exemple de la liste.

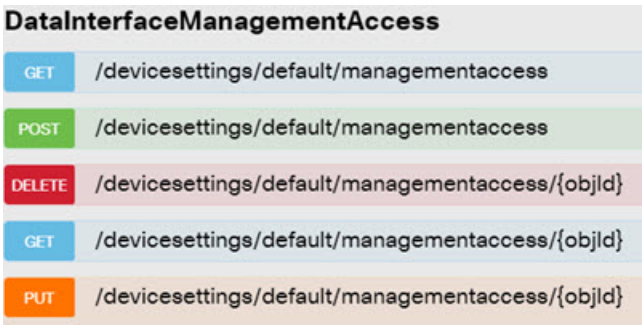


Ces noms de groupes sont des liens : cliquez sur le lien pour ouvrir le groupe et voir les méthodes que vous pouvez utiliser avec les ressources de ce dernier. Chaque groupe comprend également les commandes suivantes à droite :

- **Show/Hide** (Afficher/Masquer) ouvre et ferme le groupe. Cela revient à cliquer sur le nom du groupe. Au départ, cette action ne fait qu'afficher les méthodes (c'est-à-dire le même contenu que **List Operations** [Énumérer les opérations]), mais le système se souviendra du dernier niveau de développement de la liste (avant qu'elle ne soit fermée) et se rouvrira au même degré de développement.
- **List Operations** (Énumérer les opérations) affiche les méthodes HTTP disponibles pour chaque ressource du groupe. Les informations comprennent le chemin relatif pour le modèle de localisateur de ressources universel (URL) pour chaque ressource. Les variables de chemin sont indiquées par la convention standard : {*variable*}. Vous devez remplacer {*variable*}, y compris les accolades, par une valeur appropriée. Vous devez ajouter l'URL de base à ce chemin relatif; voir L'URL de base pour l'API, à la page 2.

  Cliquez sur un modèle d'URL d'opération pour afficher la documentation complète sur cette méthode.
- **Expand Operations** (Développer les opérations) ouvre toutes les méthodes et les ressources HTTP disponibles dans un groupe.

Certains groupes ont de nombreuses ressources enfants. Par exemple, le groupe DataInterface ManagementAccess comprend les opérations GET, POST et DELETE pour /devicesettings/default/managementaccess, ainsi que les opérations GET et PUT pour /devicesettings/default/managementaccess/{objId}.



Consulter la documentation concernant une ressource

Les attributs de chaque ressource sont documentés dans l'explorateur d'interface de protocole d'application.

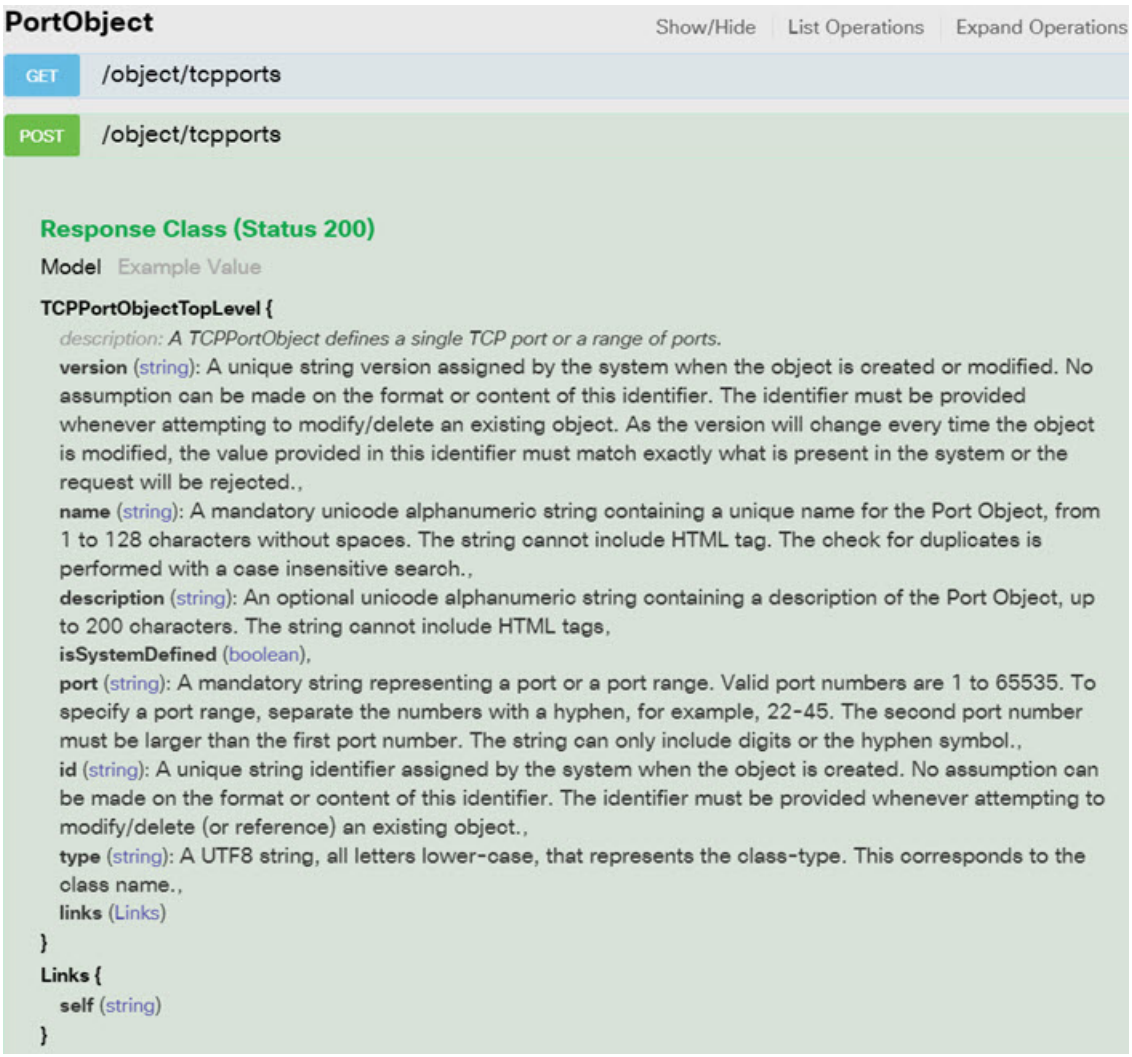
Procédure

Étape 1 Naviguez vers la ressource et la méthode qui vous intéressent.

Étape 2 Dans la section **Response Class** (Classe de réponse), cliquez sur l'onglet **Model** (Modèle).

Le modèle répertorie les attributs avec les descriptions et les types de données. Pour la méthode GET, il existe également des options de pagination qui peuvent être renvoyées. S'il y a plus d'objets que ceux renvoyés dans la réponse, vous obtenez les URL pour le lot d'objets suivant et le lot précédent.

Par exemple, le graphique suivant montre la méthode et la ressource POST /object/tcppports, avec l'onglet **Model** (Modèle) sélectionné. Par défaut, l'onglet **Example Value** (Valeur d'exemple) est sélectionné, vous devez donc toujours cliquer sur **Model** (Modèle) pour voir la documentation.



Trouver l'identifiant de l'objet (objId) et l'identifiant du parent (parentId)

Certaines ressources nécessitent un identifiant d'objet ou un identifiant d'objet parent associé dans l'URL, notamment les éléments suivants :

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

Dans la plupart des cas, vous pouvez obtenir l'identifiant de l'objet ou du parent en utilisant une méthode GET d'un niveau supérieur dans la hiérarchie des ressources. L'identifiant de l'objet ou du parent est l'UUID sur le paramètre **id** pour un objet donné.

Par exemple, GET /object/networks renvoie une liste de tous les objets réseau actuellement définis. Vous devrez peut-être effectuer plusieurs appels pour parcourir la liste et atteindre l'objet souhaité, ou inclure le paramètre de requête **limit** pour augmenter le nombre d'objets renvoyés par appel. Chaque objet a le format suivant; l'ID de l'objet est mis en évidence.

```
{
      "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/dernière version/object/networks/
9bbbc56e-8115-11e7-8cb4-01865c95f930"
      }
```

Remarque Dans certains cas, l'objet {objId} se produit au niveau supérieur d'une hiérarchie. Dans ces cas, vous pouvez parfois saisir n'importe quelle valeur pour l'identifiant de l'objet et obtenir les mêmes résultats. Dans les autres cas, examinez la documentation du modèle d'objet pour obtenir des informations sur les types d'objets valides; l'identifiant est de l'un des types valides. Il s'agit toujours d'appels GET. Par exemple, GET /operational/systeminfo/{objId} et GET /operational/featureinfo/{objId}.

Consulter le catalogue d'erreurs et évaluer les messages d'erreur

Puisque le système est un API REST, il renvoie des codes d'erreur HTTP standard, tels que 404 si vous utilisez la méthode GET pour un objet qui n'existe pas.

En outre, le système comprend un certain nombre de messages d'erreur qui expliquent les erreurs de manière plus précise. Si votre appel d'API génère une erreur, le corps de la réponse peut inclure ces messages plus précis.

Par exemple, si vous essayez d'utiliser la méthode **POST /object/networks** sur l'objet réseau suivant, vous obtiendrez une erreur. Dans ce cas, vous essayez de préciser un réseau, mais vous avez oublié d'inclure un masque réseau (c'est-à-dire que la valeur doit être 10.10.10.0/24 ou 10.10.10.0/255.255.255.0) :

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

Le résultat serait un code réponse HTTP 422 et un corps de réponse qui comprend un message d'erreur spécifique :

```
{
```

```
  "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"
      }
    ]
  }
}
```

La procédure suivante explique comment consulter la liste des messages d'erreur possibles qui peuvent être renvoyés dans un corps de réponse.

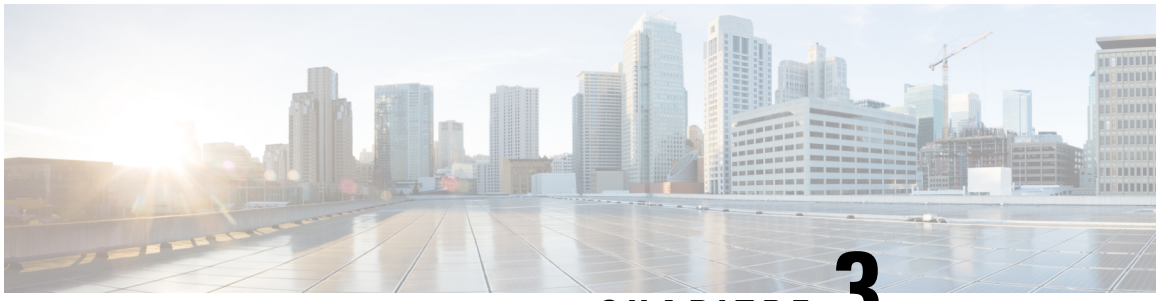
Procédure

Étape 1 Dans le gestionnaire d'appareil, cliquez sur le bouton des autres options (⋮) et choisissez **API Explorer** (Explorateur d'interface de protocole d'application).

Étape 2 Cliquez sur **Error Catalog** (Catalogue d'erreurs) dans la table des matières.

Les messages comprennent les éléments suivants.

- **Category** (Catégorie) : le type général de message. Cette valeur apparaît dans l'attribut **key** (clé) du corps de la réponse d'erreur. Les catégories comprennent Validation (Validation), General (Général) et Deployment (Déploiement).
- **Code** (Code) : une chaîne unique qui identifie le message d'erreur. Cette valeur apparaît dans l'attribut **code** (code) du corps de la réponse d'erreur. Vous pouvez utiliser la fonction Find On Page (Rechercher sur la page) du navigateur pour localiser des messages dans le catalogue avec cette valeur.
- **Message** (Message) : le message précis qui explique l'erreur. Cette valeur apparaît dans l'attribut **description** (description) du corps de la réponse d'erreur. Les variables dans le message sont indiquées comme {0}, {1}, etc.



CHAPITRE 3

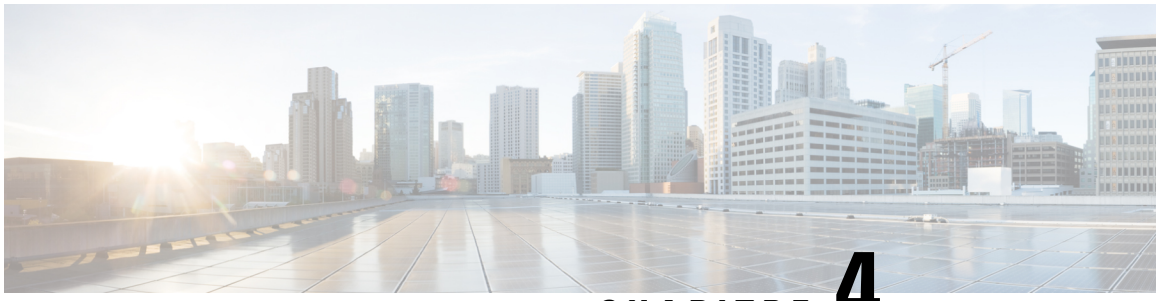
Processus général pour l'utilisation de l'API REST

Processus général pour l'utilisation de l'API REST

En général, votre client interagit avec le dispositif Défense contre les menaces en utilisant le processus itératif suivant :

1. Obtenez un jeton d'accès pour authentifier vos appels d'API. Consultez Survol du processus d'authentification du client API, à la page 13.
2. Créez une charge utile JSON sauf si vous lisez simplement des données.
3. Transmettez la charge utile JSON à l'aide d'un appel HTTPS pour le localisateur uniforme de ressource (URL) pour la ressource.
4. Utilisez la réponse JSON renvoyée.
5. Si vous apportez des modifications à la configuration, déployez-la. Consultez Déploiement des modifications de configuration, à la page 49.

Processus général pour l'utilisation de l'API REST



CHAPITRE 4

Authentifier votre client API REST à l'aide d'OAuth

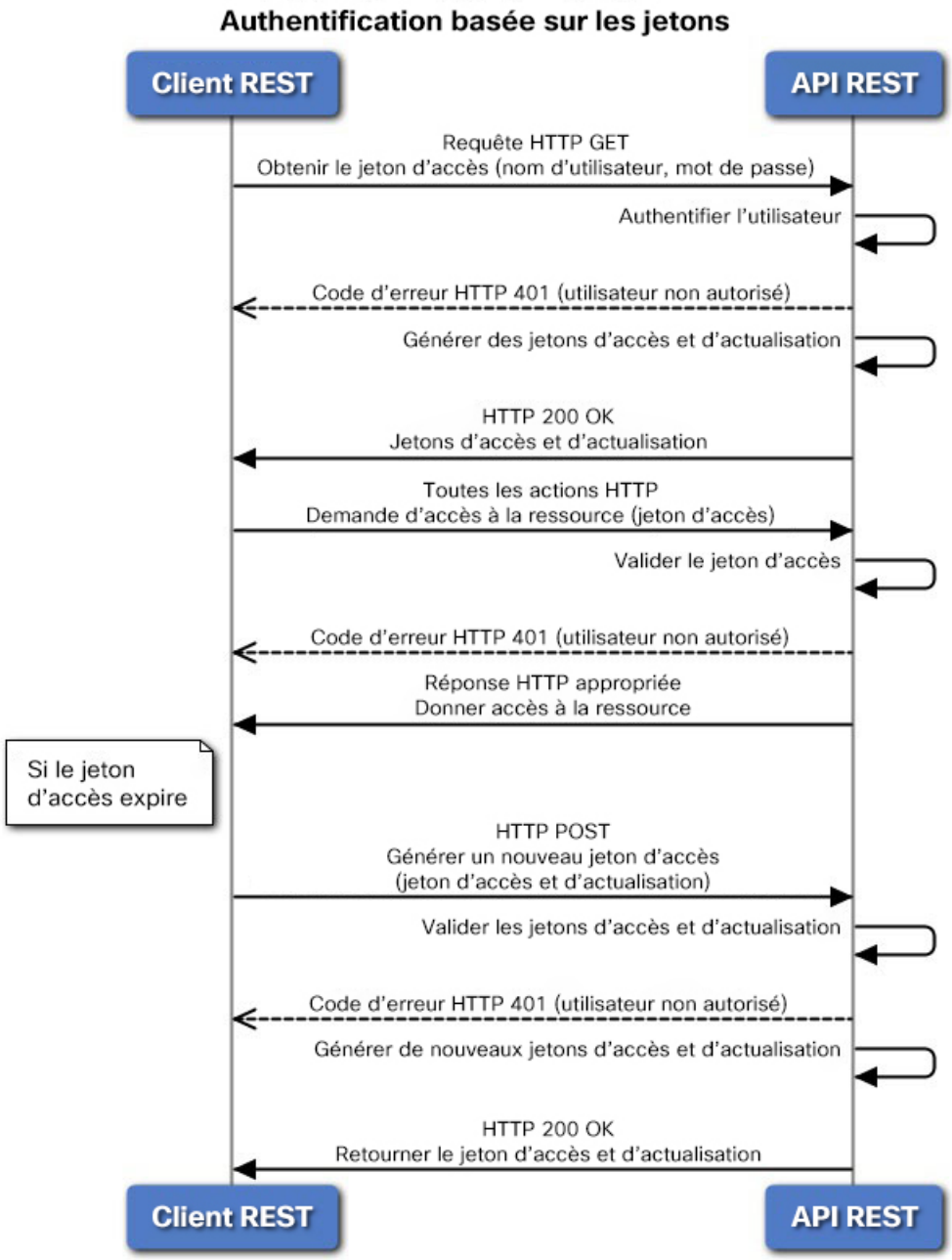
L'API REST Défense contre les menaces utilise OAuth 2.0 pour authentifier les appels provenant de clients API. OAuth est une méthode basée sur les jetons d'accès, et Défense contre les menaces utilise les jetons Web JSON comme modèle. Voici les normes pertinentes :

- RFC6749, le cadre d'autorisation OAuth 2.0, https://tools.ietf.org/html/rfc6749.
- RFC7519, jeton Web JSON (JWT), https://tools.ietf.org/html/rfc7519.

Les rubriques suivantes expliquent les méthodes pour obtenir et utiliser les jetons requis.

Survol du processus d'authentification du client API

Voici la vue de bout en bout de la façon d'authentifier votre client API auprès du dispositif Défense contre les menaces.



Avant de commencer

Chaque jeton représente une session de connexion HTTPS, qui compte pour les sessions API et les sessions gestionnaire d'appareil. Il peut y avoir un maximum de cinq sessions HTTPS actives. Si vous dépassez cette limite, la session la plus ancienne, la connexion au gestionnaire d'appareil ou le jeton API, expirera pour permettre la nouvelle session. Par conséquent, il est important que vous n'obteniez que les jetons dont vous avez besoin et que vous réutilisiez chaque jeton jusqu'à son expiration, pour ensuite les renouveler. L'obtention d'un nouveau jeton pour chaque appel d'API entraînera une perte de session importante et pourrait bloquer l'accès des utilisateurs au gestionnaire d'appareil. Ces limites ne s'appliquent pas aux sessions SSH.

Procédure

Étape 1 Authentifiez l'utilisateur du client API en utilisant la méthode dont vous avez besoin.

Votre client est tenu d'authentifier les utilisateurs et de s'assurer qu'ils sont autorisés à accéder au dispositif Défense contre les menaces. Si vous souhaitez fournir des capacités différentielles en fonction des droits d'autorisation, vous devez intégrer cette fonction à votre client.

Par exemple, si vous souhaitez autoriser l'accès en lecture seule, vous devez configurer le serveur d'authentification, les comptes d'utilisateurs et autres éléments requis. Ensuite, lorsqu'un utilisateur avec des droits en lecture seule se connecte à votre client, vous devez vous assurer d'émettre uniquement des appels GET. Dans la version v1 de l'API, ce type d'accès à une variable ne peut pas être contrôlé par le dispositif Défense contre les menaces lui-même. À partir de la version v2 de l'API, si vous utilisez des utilisateurs externes et que vous ne filtrez pas les appels en fonction de l'autorisation des utilisateurs, vous obtiendrez des erreurs en cas de non-concordance entre l'autorisation d'un utilisateur et les appels que vous tentez.

Pour la version v1, lorsque vous communiquez avec le dispositif, vous devez utiliser le compte d'utilisateur **admin** sur le dispositif Défense contre les menaces. Le compte **admin** dispose d'une autorisation complète de lecture/écriture pour tous les objets configurables par l'utilisateur.

Étape 2 Demandez un jeton d'accès attribué par mot de passe en fonction du nom d'utilisateur et du mot de passe à l'aide du compte **admin**.

Consultez Demander un jeton d'accès attribué par mot de passe, à la page 15.

Étape 3 Vous pouvez également demander un jeton d'accès personnalisé pour votre client.

Avec un jeton personnalisé, vous pouvez demander explicitement une période de validité et attribuer un nom de sujet au jeton. Consultez Demander un jeton d'accès personnalisé, à la page 17.

Étape 4 Utilisez le jeton d'accès sur les appels d'API dans l'en-tête Authorization: Bearer (Autorisation : porteur).

Consultez Utiliser un jeton d'accès sur les appels d'API, à la page 19.

Étape 5 Avant que le jeton d'accès expire, actualisez le jeton.

Consultez Renouveler un jeton d'accès, à la page 20.

Étape 6 Lorsque vous avez terminé, révoquez le jeton s'il n'a pas encore expiré.

Consultez Révoquer un jeton d'accès, à la page 21.

Demander un jeton d'accès attribué par mot de passe

Chaque appel de l'API REST doit inclure un jeton d'authentification pour vérifier que l'appelant est autorisé à effectuer l'action demandée. Initialement, vous devez obtenir un jeton d'accès en fournissant le nom d'utilisateur/mot de passe **admin**. Cela s'appelle un jeton d'accès accordé par mot de passe, c'est-à-dire que grant_type = password (type_d'octroi = mot de passe).

Procédure

Étape 1 Créez l'objet JSON pour l'octroi du jeton d'accès accordé par mot de passe.

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

Précisez le nom d'utilisateur **admin** et le bon mot de passe, par exemple :

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

Étape 2 Utilisez POST /fdm/token pour obtenir le jeton d'accès.

Par exemple, la commande **curl** ressemblerait à ce qui suit :

```
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/dernière version/fdm/token'
```

Étape 3 Récupérez les jetons d'accès et de renouvellement de la réponse.

Une bonne réponse (code d'état 200) ressemble à ce qui suit :

```
{
  "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
}
```

Lieu :

- **access_token** est le jeton-porteur que vous devez inclure dans les appels d'API. Consultez Utiliser un jeton d'accès sur les appels d'API, à la page 19.

- **expires_in** est le nombre de secondes pendant lesquelles le jeton d'accès est valide, à partir de l'émission de ce dernier.
- **refresh_token** est le jeton que vous utiliseriez pour une demande de renouvellement. Consultez Renouveler un jeton d'accès, à la page 20.
- **refresh_expires_in** est le nombre de secondes pendant lesquelles le jeton de renouvellement est valide. Ce nombre est toujours plus élevé que le nombre de secondes de la période de validité du jeton d'accès.

Demander un jeton d'accès personnalisé

Vous pouvez utiliser le jeton d'accès attribué par mot de passe. Cependant, vous pouvez également demander un jeton d'accès personnalisé. Avec un jeton personnalisé, vous pouvez fournir un nom de sujet pour différencier l'utilisation des jetons (pour votre référence ultérieure). Vous pouvez également demander des périodes de validité spécifiques si les valeurs par défaut renvoyées pour les jetons de mot de passe ne satisfont pas à vos exigences.

Avant de commencer

Vous devez d'abord obtenir un jeton d'accès attribué par mot de passe avant d'obtenir un jeton personnalisé. Consultez Demander un jeton d'accès attribué par mot de passe, à la page 15.

De plus :

- Vous pouvez demander un jeton personnalisé uniquement si vous êtes un utilisateur local. Les utilisateurs externes ne peuvent pas demander de jetons personnalisés.
- Vous pouvez utiliser un jeton personnalisé sur l'unité pour laquelle vous l'obtenez uniquement. Vous ne pouvez pas utiliser le jeton sur l'appareil homologue dans un groupe à haute disponibilité.

Procédure

Étape 1 Créez l'objet JSON pour l'octroi du jeton d'accès personnalisé.

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

Lieu :

- **access_token** est un jeton d'accès attribué par mot de passe valide.

- **desired_expires_in** est un entier représentant le nombre de secondes pendant lesquelles le jeton d'accès personnalisé sera valide. En comparaison, les jetons attribués par mot de passe sont valides pendant 1800 secondes.
- **desired_refresh_expires_in** est un entier représentant le nombre de secondes pendant lesquelles le jeton d'actualisation personnalisé sera valide. Si vous obtenez un jeton d'actualisation, assurez-vous que cette valeur est supérieure à la valeur **desired_expires_in**. En comparaison, les jetons d'actualisation attribués par mot de passe sont valides pendant 2400 secondes. Ce paramètre n'est pas obligatoire si vous spécifiez 0 pour **desired_refresh_count**.
- **desired_subject** est un nom que vous attribuez au jeton personnalisé.
- **desired_refresh_count** est le nombre de fois où vous souhaitez pouvoir actualiser le jeton. Précisez 0 si vous ne souhaitez pas obtenir de jeton d'actualisation. Si vous n'avez pas de jeton d'actualisation, vous devez obtenir un nouveau jeton d'accès lorsque le jeton existant expire.

Par exemple, l'option suivante demande un jeton personnalisé pour api-client qui expire après 2400 secondes, avec un jeton d'actualisation qui expire après 3000 secondes. Le jeton peut être actualisé trois fois.

```
{{
  "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
}
```

Étape 2 Utilisez POST /fdm/token pour obtenir le jeton d'accès.

Par exemple, la commande **curl** ressemblerait à ce qui suit :

```
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/dernière version/fdm/token'
```

Étape 3 Récupérez les jetons d'accès et de renouvellement de la réponse.

Une bonne réponse (code d'état 200) ressemble à ce qui suit :

```
{
```

```
  "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
}
```


Lieu :

- **access_token** est le jeton-porteur que vous devez inclure dans les appels d'API. Consultez Utiliser un jeton d'accès sur les appels d'API, à la page 19.
- **expires_in** est le nombre de secondes pendant lesquelles le jeton d'accès est valide, à partir de l'émission de ce dernier.
- **refresh_token** est le jeton que vous utiliseriez pour une demande de renouvellement. Consultez Renouveler un jeton d'accès, à la page 20.
- **refresh_expires_in** est le nombre de secondes pendant lesquelles le jeton de renouvellement est valide. Ce nombre est toujours plus élevé que le nombre de secondes de la période de validité du jeton d'accès.

Utiliser un jeton d'accès sur les appels d'API

Après avoir obtenu un jeton d'accès personnalisé ou par mot de passe, vous devez l'inclure sur chaque appel d'API dans l'en-tête **Authorization: Bearer** (Autorisation : porteur) de la demande HTTPS.

Par exemple, une commande **curl** pour lancer une méthode GET /object/networks pourrait ressembler à ce qui suit :


```
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/dernière version/object/networks'
```

Remarque Lorsque vous utilisez l'explorateur d'interface de protocole d'application pour essayer des méthodes et des ressources, la commande **curl** affichée n'inclut pas l'en-tête **Authorization: Bearer** (Autorisation : porteur). Cependant, vous devez ajouter cet en-tête lorsque vous passez des appels à partir de votre client API.

Renouveler un jeton d'accès

Après l'expiration d'un jeton d'accès, vous devez le renouveler à l'aide du jeton de renouvellement qui a été fourni lors de l'octroi d'origine. Le jeton d'accès renouvelé est en fait différent du jeton d'accès d'origine. Lors du renouvellement, une nouvelle paire de jetons d'accès et de jetons de renouvellement est fournie. Ce processus ne fait pas que prolonger tout simplement la durée de vie de l'ancien jeton d'accès.

Procédure

Étape 1 Créez l'objet JSON pour l'octroi de jeton de renouvellement.

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

Le **refresh_token** peut provenir d'un octroi par mot de passe ou par jeton d'accès.

Par exemple :

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

Étape 2 Utilisez POST /fdm/token pour obtenir le jeton d'accès renouvelé.

Par exemple, la commande **curl** ressemblerait à ce qui suit :

```
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/dernière version/fdm/token'
```

Étape 3 Récupérez les jetons d'accès et de renouvellement de la réponse.

Une bonne réponse (code d'état 200) ressemble à ce qui suit. Dans cet exemple, le jeton de renouvellement était pour un jeton personnalisé. Les périodes d'expiration sont basées sur les valeurs de la demande de jeton d'accès personnalisé initiale.

```
{
  "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
}
```

Lieu :

- **access_token** est le jeton-porteur que vous devez inclure dans les appels d'API. Consultez Utiliser un jeton d'accès sur les appels d'API, à la page 19.
- **expires_in** est le nombre de secondes pendant lesquelles le jeton d'accès est valide, à partir de l'émission de ce dernier.
- **refresh_token** est le jeton que vous utiliseriez pour une demande de renouvellement.
- **refresh_expires_in** est le nombre de secondes pendant lesquelles le jeton de renouvellement est valide. Ce nombre est toujours plus élevé que le nombre de secondes de la période de validité du jeton d'accès.

Révoquer un jeton d'accès

Comme les jetons d'accès sont valides pour une durée donnée, vous devez nettoyer votre système en révoquant un jeton lorsque l'utilisateur se déconnecte de votre client API. Ainsi, vous vous assurez qu'aucune porte d'entrée vulnérable n'est laissée ouverte pour le dispositif Défense contre les menaces.

Procédure

Étape 1 Créez l'objet JSON pour l'octroi de jeton de révocation.

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

Lieu :

- **access_token** doit être un jeton d'accès attribué par mot de passe. Vous ne pouvez pas révoquer un jeton en utilisant un jeton d'accès personnalisé.
- Vous devez spécifier une et une seule des options suivantes :
  - **token_to_revoke** est un jeton attribué par mot de passe ou un jeton personnalisé que vous souhaitez révoquer. Il peut s'agir du même jeton que **access_token**, vous pouvez donc utiliser un jeton attribué par mot de passe pour le révoquer.
  - (Ne pas utiliser.) **custom_token_id_to_revoke** identifie le jeton d'accès personnalisé par son identifiant unique interne. Cependant, il n'y a aucun moyen direct par lequel vous pouvez obtenir cette valeur. Utilisez plutôt les autres options.
  - **custom_token_subject_to_revoke** est la valeur **desired_subject** du jeton d'accès personnalisé que vous souhaitez révoquer.

Par exemple :

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

Étape 2 Utilisez POST /fdm/token pour révoquer le jeton d'accès.

Par exemple, la commande **curl** ressemblerait à ce qui suit :

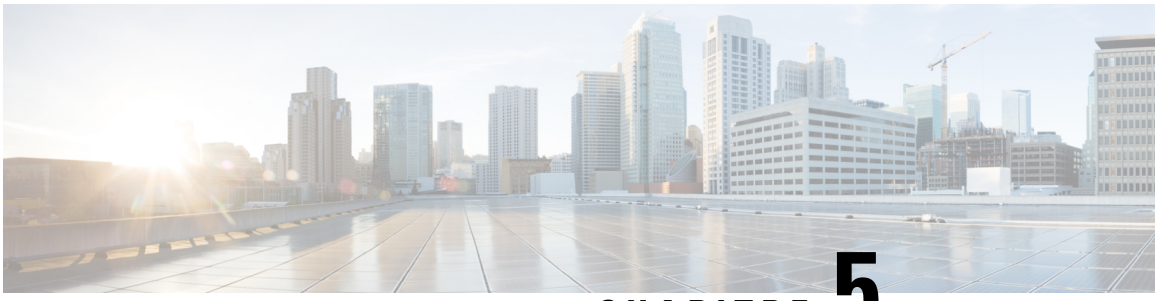
```
curl -X POST --header 'Content-Type: application/json'
--header 'Accept: application/json' -d '{
   "grant_type": "revoke_token",
   "access_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQ
iOjE1MDI5MDQzMjQsInN1YiI6ImFkbWluIiwianRpIjoiZTM
```

```
zNGIxOWYtODJhNy0xMWU3LWE4MWMtNGQ3NzY2ZTExMzVkIiw
ibmJmIjoxNTAyOTA0MzI0LCJleHAiOjE1MDI5MDYxMjQsInJ
lZnJlc2hUb2tlbkV4cGlyZXNBdCI6MTUwMjkwNjcyNDExMiw
idG9rZW5UeXBlIjoiSldUX0FjY2VzcyIsIm9yaWdpbiI6InB
hc3N3b3JkIn0.OVZBT9yVZc4zxZfZiiLH4SZcFclaHyCPbZJ
C_Gyd5FE",
   "custom_token_subject_to_revoke": "api-client"
 }' 'https://ftd.example.com/api/fdm/dernière version/fdm/token'
```

Étape 3 Évaluez la réponse pour vérifier que le jeton a été révoqué.

Une bonne réponse (code d'état 200) ressemble à ce qui suit.

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



CHAPITRE 5

Configurer des utilisateurs externes pour l'API

Exigence relative à la version : pour utiliser AAA externe, vous devez exécuter la version 6.3(0) ou supérieure de défense contre les menaces, et la version v2 ou supérieure de l'API REST défense contre les menaces.

Vous pouvez configurer le dispositif pour qu'il utilise un serveur RADIUS AAA externe pour authentifier et autoriser l'accès des utilisateurs à l'API REST défense contre les menaces. Vous pouvez utiliser les comptes d'utilisateur RADIUS à la place ou en plus du compte d'utilisateur **admin** local intégré.

Lorsque vous utilisez un serveur AAA externe, vous pouvez définir différents niveaux d'autorisation pour différents comptes. Cela vous permet de limiter les entités qui peuvent apporter des modifications à la configuration du dispositif tout en fournissant un accès en lecture seule au personnel de soutien.

La procédure suivante explique le processus de bout en bout de configuration des comptes RADIUS et de configuration du dispositif pour qu'il utilise un protocole AAA externe pour l'authentification et l'autorisation.

Avant de commencer

Gardez les facteurs opérationnels suivants à l'esprit lorsque vous utilisez une autorisation externe.

- Si le dispositif est configuré pour fonctionner en mode haute disponibilité, configurez l'autorisation externe sur l'unité active. Vous devez ensuite exécuter une tâche de déploiement pour les paramètres d'autorisation afin de permettre l'accès utilisateur au dispositif de secours.
- Chaque fois qu'un nouvel utilisateur accède au système, une ressource d'utilisateur est créée pour ce dernier. Vous devez déployer la configuration pour enregistrer cet objet utilisateur.

  (Versions antérieures à 6.6. de défense contre les menaces) Si vous travaillez en mode haute disponibilité (HA), vous devez déployer la configuration avant que cet utilisateur puisse se connecter à l'unité de secours. Étant donné que seuls les utilisateurs administrateurs ou en lecture-écriture peuvent démarrer une tâche de déploiement, un nouvel utilisateur en lecture seule doit demander à une autre personne de déployer la configuration pour enregistrer l'objet utilisateur.

  À partir de la version 6.6 de défense contre les menaces, les restrictions du mode HA sont supprimées. Un utilisateur externe peut se connecter à l'unité en veille sans d'abord se connecter à l'unité active et déployer la configuration. L'objet utilisateur ne sera pas créé sur l'unité en veille, mais les caractéristiques de l'utilisateur seront mises en mémoire cache et l'accès sera accordé à l'utilisateur, en supposant qu'un nom d'utilisateur et un mot de passe valides sont fournis.

Procédure

Étape 1 Définir les droits d'autorisation dans les comptes d'utilisateurs RADIUS, à la page 26.

Étape 2 Définir les serveurs RADIUS, à la page 27.

Étape 3 Créer un groupe de serveurs AAA pour les serveurs RADIUS, à la page 28.

Étape 4 Désigner le groupe de serveurs AAA comme source d'authentification pour l'accès HTTPS, à la page 31.

Étape 5 Utilisez **POST /operational/deploy** pour démarrer une tâche de déploiement.

La commande **curl** ressemblerait à ce qui suit :

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

'https://ftd.example.com/api/fdm/dernière version/operational/deploy'
```

Pour de plus amples renseignements, consultez Déploiement des modifications de configuration, à la page 49.

Étape 6 Vérifier l'accès d'utilisateurs externes, à la page 34.

Définir les droits d'autorisation dans les comptes d'utilisateurs RADIUS

Vous pouvez fournir un accès SSH à l'API REST de défense contre les menacesà partir d'un serveur RADIUS externe. En activant l'authentification et l'autorisation RADIUS, vous pouvez fournir différents niveaux de droits d'accès et ne pas demander à chaque utilisateur de se connecter avec le compte **admin** local.

Remarque Ces utilisateurs externes sont également autorisés pour gestionnaire d'appareil.

Pour fournir un contrôle d'accès basé sur les rôles (RBAC), mettez à jour les comptes d'utilisateur sur votre serveur RADIUS pour définir l'attribut **cisco-av-pair**. Cet attribut doit être défini correctement sur un compte utilisateur, sinon l'utilisateur se voit refuser l'accès à l'API REST. Voici les valeurs suivantes prises en charge pour l'attribut cisco-av-pair :

- **fdm.userrole.authority.admin** fournit un accès administrateur complet. Ces utilisateurs peuvent effectuer toutes les actions que l'utilisateur **admin** local peut effectuer.
- **fdm.userrole.authority.rw** fournit un accès en lecture-écriture. Ces utilisateurs peuvent faire tout ce qu'un utilisateur en lecture seule peut faire, ainsi que modifier et déployer la configuration. Les seules restrictions concernent les actions critiques pour le système, qui comprennent l'installation des mises à

niveau, la création et la restauration de sauvegardes, l'affichage du journal d'audit et la déconnexion d'autres utilisateurs.

- **fdm.userrole.authority.ro** fournit un accès en lecture seule. Ces utilisateurs peuvent afficher les tableaux de bord et la configuration, mais ne peuvent apporter aucune modification. Si l'utilisateur tente d'apporter une modification, le message d'erreur explique que cela est causé par un manque d'autorisation.

Définir les serveurs RADIUS

Après avoir configuré les comptes d'utilisateur dans le serveur RADIUS pour définir les droits d'autorisation appropriés, vous pouvez configurer le dispositif pour qu'il utilise le serveur pour authentifier et autoriser l'accès à l'API REST.

Utilisez la ressource **POST /object/radiusidentitysources** pour créer un objet pour chaque serveur RADIUS que vous souhaitez définir.

Procédure

Étape 1 Créez le corps de l'objet JSON pour le serveur RADIUS.

Voici un exemple d'objet JSON à utiliser avec cet appel.

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

Les attributs sont :

- **name** (nom) : le nom de l'objet. Cela ne doit pas absolument correspondre à tout ce qui est défini dans le serveur RADIUS.
- **description** (description) : (facultatif.) Une description de l'objet.
- **host** (domaine) : l'adresse IP ou le nom de domaine complet du serveur RADIUS.
- **timeout** (délai d'expiration) : (facultatif.) La durée, de 1 à 300 secondes, pendant laquelle le système attend une réponse du serveur avant d'envoyer la demande au serveur suivant. Si vous n'incluez pas cet attribut, la valeur par défaut est de 10 secondes.
- **serverAuthenticationPort** (portD'AuthentificationDuServeur) : (facultatif.) Le port sur lequel l'authentification et l'autorisation RADIUS sont effectuées. Si vous n'incluez pas cet attribut, la valeur par défaut sera 1812.
- **serverSecretKey** (cléSecrèteDuServeur) : (facultatif.) le code secret partagé qui est utilisé pour chiffrer les données entre le dispositif défense contre les menaceset le serveur RADIUS. La clé est une chaîne de caractères alphanumériques sensible à la casse et composé de jusqu'à 64 caractères, espaces exclus. La clé doit commencer par un caractère alphanumérique ou un trait de soulignement et peut contenir les

caractères spéciaux : $ & - _ . + @. Cette chaîne de caractères doit correspondre à la clé configurée sur le serveur RADIUS. Si vous ne configurez pas de clé secrète, la connexion ne sera pas chiffrée.

Étape 2 Postez l'objet.

Par exemple, la commande **curl** ressemblerait à ce qui suit :

```
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/dernière version/object/radiusidentitysources'
```

Étape 3 Vérifiez la réponse.

Vous devriez obtenir un code réponse de 200. Un corps de réponse qui fonctionne ressemblera à ce qui suit : Notez que les informations sensibles, telles que la clé secrète, sont masquées dans la réponse.

```
{
  "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/dernière version/object/
radiusidentitysources/1b962e3b-6e56-11e8-bd65-379fa8aaaba1"
  }
}
```

Créer un groupe de serveurs AAA pour les serveurs RADIUS

Après avoir créé les objets de serveur RADIUS, utilisez la ressource **POST /object/radiusidentitysourcegroups** pour créer un groupe AAA contenant les objets radiusidentitysource.

Vous pouvez ajouter jusqu'à 16 serveurs RADIUS à un groupe de serveurs RADIUS AAA. Ces serveurs doivent être des copies de sauvegarde les uns des autres, c'est-à-dire qu'ils doivent avoir la même liste de comptes d'utilisateurs.

Procédure

Étape 1 Créez le corps de l'objet JSON pour le groupe de serveurs RADIUS.

Voici un exemple d'objet JSON à utiliser avec cet appel.

```
{
  "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"
}
```

Les attributs sont :

- **name** (nom) : le nom de l'objet. Cela ne doit pas obligatoirement correspondre à tout ce qui est défini dans les serveurs RADIUS du membre.
- **maxFailedAttempts** (nombreMaximumDeTentativesÉchouées) : (facultatif.) Les serveurs défaillants ne sont réactivés que lorsque tous les serveurs sont tombés en panne. Le temps mort est le temps d'attente, de 0 à 1440 minutes, après l'échec du dernier serveur avant la réactivation de tous les serveurs. Si vous n'incluez pas cet attribut, la valeur par défaut sera 10 minutes.
- **deadTime** (tempsMort) : (facultatif.) Le nombre de demandes ayant échoué (c'est-à-dire de demandes qui ne reçoivent pas de réponse) envoyées à un serveur RADIUS du groupe avant d'essayer le serveur suivant. Vous pouvez spécifier de 1 à 5, et la valeur par défaut est 3. Lorsque le nombre maximal de tentatives ayant échoué est dépassé, le système marque le serveur comme ayant échoué.

  Pour une fonctionnalité donnée, si vous avez configuré une méthode de secours à l'aide de la base de données locale et qu'aucun serveur du groupe ne répond, le groupe est considéré comme ne répondant pas et la méthode de secours est essayée. Le groupe de serveurs reste marqué comme ne répondant pas pendant la durée du temps mort, de sorte que les demandes AAA supplémentaires effectuées dans cette période ne résultent pas en une tentative d'entrer en contact avec le groupe de serveurs et que la méthode de secours est utilisée immédiatement.
- **description** (description) : (facultatif.) Une description de l'objet.
- **radiusIdentitySources** (sourceD'IdentitéRadius) : il s'agit d'un groupe d'éléments qui définit chaque objet radiusidentitysource (sourced'identitéradius) qui définit un serveur RADIUS à inclure dans le groupe. Incluez les éléments entre les [crochets]. Voici les attributs et la syntaxe de chaque objet. Vous obtenez la valeur des attributs **id** (identifiant), **version** (version) et **name** (nom) des objets individuels; les informations se trouvent dans le corps de la réponse lorsque vous créez les objets. Vous pouvez également obtenir les informations à partir d'un appel **GET /object/radiusidentitysources**. Le **type** doit être **radiusidentitysource**.

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

Étape 2 Postez l'objet.

Par exemple, la commande **curl** ressemblerait à ce qui suit :

```
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/dernière version/object/radiusidentitysourcegroups'
```

Étape 3 Vérifiez la réponse.

Vous devriez obtenir un code réponse de 200. Un corps de réponse qui fonctionne ressemblera à ce qui suit :

```
{
  "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/dernière version/object/
radiusidentitysourcegroups/0a7996ae-6e5b-11e8-bd65-dbab801c44b9"
  }
}
```

Désigner le groupe de serveurs AAA comme source d'authentification pour l'accès HTTPS

Utilisez la ressource **PUT /devicesettings/default/aaasettings/{objId}** pour identifier le groupe de serveurs RADIUS AAA comme source d'identité pour l'autorisation de l'utilisateur.

Il n'y a pas de méthode POST : les objets nécessaires à l'authentification du système existent déjà. Vous devez d'abord faire appel à une méthode GET pour déterminer les valeurs d'identifiant et de version pertinentes.

Procédure

Étape 1 Utilisez **GET /devicesettings/default/aaasettings** pour déterminer les attributs des objets aaasettings.

La commande **curl** ressemblerait à ce qui suit :

```
curl -X GET --header 'Accept: application/json'
'https://ftd.example.com/api/fdm/dernière version/devicesettings/default/aaasettings'
```

Par exemple, le corps de la réponse devrait ressembler à ce qui suit. Cet exemple montre que la source d'identité locale est celle définie pour l'accès HTTPS. Il est également utilisé pour l'accès SSH, qui n'est pas pertinent pour l'API REST.

```
{
  "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/dernière version/
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/dernière version/
devicesettings/default/aaasettings/00000003-0000-0000-0000-000000000008"
      }
    }
  ],
  "paging": {
    "prev": [],
    "next": [],
    "limit": 10,
    "offset": 0,
    "count": 2,
    "pages": 0
  }
}
```

Étape 2 (Facultatif) Utilisez **GET /devicesettings/default/aaasettings/{objId}** pour obtenir une copie de l'objet de paramètre HTTPS AAA afin de restreindre votre vue.

Votre appel PUT mettra à jour l'objet HTTPS uniquement. Vous n'avez pas besoin de mettre à jour l'objet SSH.

Dans cet exemple, l'identifiant de l'objet HTTPS est 00000003-0000-0000-0000-000000000007. La commande **curl** ressemblerait donc à ce qui suit :

```
curl -X GET --header 'Accept: application/json'
'https://ftd.example.com/api/fdm/dernière version/devicesettings/
default/aaasettings/00000003-0000-0000-0000-000000000007'
```

Le corps de la réponse ressemblerait à ce qui suit.

```
{
  "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/dernière version/
devicesettings/default/aaasettings/00000003-0000-0000-0000-000000000007"
  }
```

Étape 3 Créez le corps d'objet JSON pour l'accès à la gestion AAA.

Voici un exemple d'objet JSON à utiliser avec cet appel.

```
{
  "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"
}
```

Les attributs sont :

- **version** (version) : la version de l'objet HTTPS. Vous trouverez cette valeur dans le corps de la réponse pour l'appel de la méthode GET.
- **name** (nom) : le nom de l'objet, **HTTPS**. Vous trouverez cette valeur dans le corps de la réponse pour l'appel de la méthode GET.
- **identitySourceGroup** (groupeSourceDeL'Identité) : cette valeur identifie le groupe de serveurs RADIUS. Obtenez les valeurs **id** (identifiant), **version** (version) et **name** (nom) dans le corps de la réponse une fois le groupe de serveurs (ou un appel **GET /object/radiusidentitysourcegroups**) créé. Le type doit être **radiusidentitysourcegroup**.
- **description** (description) : (facultatif.) Une description de l'objet.
- **protocolType** (typeDeProtocole) : le protocole auquel cette source s'applique, **HTTPS**.
- **useLocal** (utiliserLocal) : comment utiliser la source d'identité locale, qui contient le compte d'utilisateur administrateur local. Saisissez l'une des options suivantes :
  - **Before** (Avant) : le système vérifie d'abord le nom d'utilisateur et le mot de passe par rapport à la source locale.
  - **After** (Après) : la source locale est vérifiée uniquement si la source externe n'est pas disponible ou si le compte d'utilisateur n'a pas été trouvé dans la source externe.
  - **Never** (Jamais) : (non recommandé.) La source locale n'est jamais utilisée, vous ne pouvez donc pas vous connecter en tant qu'utilisateur **admin**.

    Mise en garde
    Si vous sélectionnez **Never** (Jamais), vous ne pourrez pas vous connecter au gestionnaire d'appareil ni utiliser l'API en utilisant le compte **admin**. Vous serez verrouillé du système si le serveur RADIUS n'est plus disponible ou si vous ne configurez pas les comptes correctement dans le serveur RADIUS.

- **id** : la valeur d'identifiant de l'objet HTTPS. Vous trouverez cette valeur dans le corps de la réponse pour l'appel de la méthode GET.
- **type** (type) : le type d'objet, **aaasetting**.

Étape 4 Placez l'objet.

Par exemple, la commande **curl** ressemblerait à ce qui suit. Notez que l'objet {objId} dans l'URL et l'identifiant de l'objet aaasettings dans l'objet JSON sont identiques.

```
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/dernière version/devicesettings/
default/aaasettings/00000003-0000-0000-0000-000000000007'
```

Étape 5 Vérifiez la réponse.

Vous devriez obtenir un code réponse de 200. Un corps de réponse qui fonctionne ressemblera à ce qui suit :

```
{
  "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/dernière version/devicesettings/
default/aaasettings/00000003-0000-0000-0000-000000000007"
  }
}
```

Vérifier l'accès d'utilisateurs externes

Une fois la tâche de déploiement terminée, vous pouvez tester l'accès d'un utilisateur externe au gestionnaire d'appareil et à l'API REST.

Procédure

Étape 1 Connectez-vous au gestionnaire d'appareil en utilisant un nom d'utilisateur externe doté d'un attribut cisco-av-pair valide.

La connexion devrait réussir, et le coin supérieur droit de la page devrait afficher le nom d'utilisateur et le niveau de privilège.

Étape 2 Obtenez un jeton API REST pour un utilisateur externe.

Si l'utilisateur peut obtenir un jeton, il peut utiliser les ressources et les méthodes autorisées pour le niveau de privilège qui lui a été attribué.

a) Créez le corps de l'objet JSON pour un jeton simple attribué par mot de passe.

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

b) Utilisez **POST /fdm/token** pour obtenir le jeton.

```
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/dernière version/fdm/token'
```

c) Évaluez la réponse pour vérifier qu'un jeton a été accordé.

Vous devriez obtenir un code réponse de 200. Lorsque vous obtenez un jeton, cela signifie que le système a pu authentifier l'utilisateur.

```
{
  "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
}
```

Étape 3 Utilisez **GET /objet/users** pour vérifier que les objets utilisateur ont été créés pour chaque utilisateur.

Les objets utilisateur sont automatiquement créés pour chaque nouvel utilisateur qui se connecte au gestionnaire d'appareil ou qui obtient un jeton d'accès. Vous devez exécuter une tâche de déploiement pour enregistrer ces objets utilisateur. En mode haute disponibilité, la tâche de déploiement est requise avant que l'utilisateur puisse se connecter à l'unité en veille.

Par exemple, la commande **curl** ressemblerait à ce qui suit :

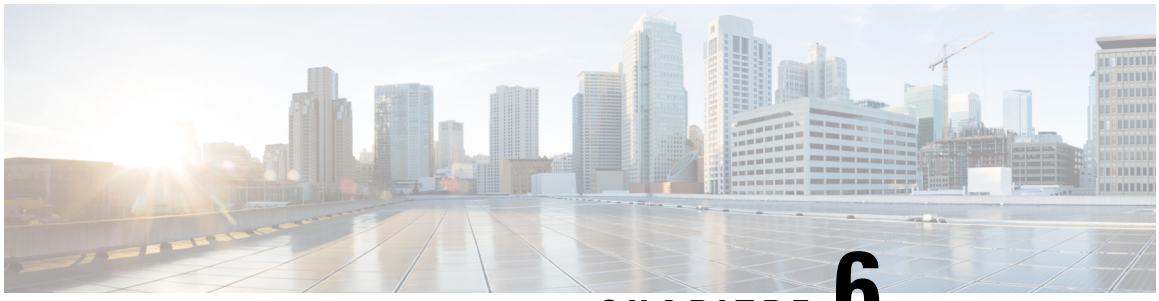
```
curl -X GET --header 'Accept: application/json'
'https://ftd.example.com/api/fdm/dernière version/object/users'
```

Le corps de réponse suivant indique que deux utilisateurs externes se sont connectés. Notez que la valeur **userRule** affiche les droits concédés par la commande cisco-av-pair et configurés dans le serveur RADIUS

pour ces comptes d'utilisateurs. Utilisez ces informations pour vérifier que vous avez configuré les comptes d'utilisateur du serveur RADIUS correctement. L'utilisateur **admin** est l'utilisateur défini localement.


```
{
  "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/dernière version/
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/dernière version/
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/dernière version/
object/users/29b20e67-6e64-11e8-bd65-3582e0f59b48"
      }
    }
  ],
  "paging": {
    "prev": [],
    "next": [],
    "limit": 10,
    "offset": 0,
    "count": 3,
    "pages": 0
  }
}
```



CHAPITRE 6

Utiliser les méthodes et les ressources

Les rubriques suivantes expliquent en général comment utiliser les différentes méthodes et ressources.

Tester une méthode et en interpréter les résultats

Vous pouvez utiliser l'explorateur d'interface de protocole d'application pour tester les différentes méthodes. Cette rubrique explique le processus général et offre une explication de la réponse renvoyée par le système. Consultez les rubriques sur chaque type de méthode pour connaître les techniques spécifiques des différentes méthodes.

Le bouton **Try It Out!** (Essayez!) pour chaque méthode ou ressource interagit directement avec le système. La méthode GET récupère les données réelles, les méthodes POST/PUT créent ou modifient des ressources réelles, et la méthode DELETE supprime des objets réels. Vous apportez de réelles modifications de configuration dans le système, bien que les modifications ne soient pas déployées immédiatement. Pour que les modifications entrent en vigueur, utilisez la ressource POST /operational/deploy pour démarrer une tâche de déploiement.

Vous pouvez trouver le bouton **Try It Out!** (Essayez!) après la section **Response Message** (Message de réponse) lorsque vous ouvrez une méthode ou une ressource. Pour certaines méthodes ou ressources, vous devez saisir un identifiant d'objet pour les tester. Dans ce cas, vous devez généralement d'abord utiliser une méthode GET sur la ressource parente. Pour en savoir plus, consultez Trouver l'identifiant de l'objet (objId) et l'identifiant du parent (parentId), à la page 8.

Pour les méthodes POST/PUT, vous devez également définir les valeurs requises dans le modèle JSON.

Après avoir cliqué sur le bouton **Try It Out!** (Essayez!), l'explorateur d'interface de protocole d'application ajoutera les résultats à la page après le bouton. La réponse comprendra les sections suivantes :

Curl

La commande **curl** utilisée pour passer l'appel. Par exemple, en cliquant sur le bouton **Try It Out!** (Essayez!) pour la ressource GET /object/networks, vous recevrez une réponse similaire à ce qui suit. L'élément « v » du chemin change à chaque nouvelle version de l'API.

```
curl -X GET --header 'Accept: application/json'
'https://ftd.example.com/api/fdm/dernière version/object/networks'
```

Remarque Cela ne comprend pas l'en-tête **Authorization: Bearer** (Autorisation : porteur), qui serait requis dans un appel API de votre client.

URL de la demande

L'URL à faire émettre par votre client pour faire la demande. Par exemple, pour la ressource GET /objet/networks :

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

Corps de la réponse

L'objet que le système renvoie à votre client. Si une ressource peut comprendre plusieurs objets (comme /objet/network), vous obtiendrez une liste d'éléments si vous effectuez une requête GET. Les réponses pour les méthodes POST/PUT/DELETE ne renverront qu'un seul objet.

Le contenu spécifique retourné sera déterminé en fonction du modèle de ressource. Par exemple, la méthode GET /object/networks renverra une liste d'objets, chaque objet ressemblant à ce qui suit (l'indication initiale d'une liste d'éléments sera également affichée). Notez que la valeur links/self indique l'URL que vous utilisez pour faire référence à cet objet; l'identifiant de l'objet est inclus dans l'URL.

```
{
  "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/dernière version/
object/networks/900fac69-7d19-11e7-bf7b-d9417b20e59e"
      }
    },
```

Les requêtes GET comprennent également une section de pagination, qui est expliquée dans GET : obtenir des données du système, à la page 41.

Code réponse

Le code d'état HTTP numérique renvoyé pour l'appel HTTP. Il s'agit des codes d'état HTTP standard, que vous pouvez trouver dans les RFC ou dans Wikipédia (à l'adresse https://fr.wikipedia.org/wiki/Liste_des_codes_HTTP par exemple) Par exemple, 200 (OK) indique un appel des méthodes GET/PUT/POST réussi, et 204 un appel DELETE réussi.

En-têtes de réponse

Voici les en-têtes de paquet dans la réponse HTTP. Par exemple, GET /object/networks peut avoir des en-têtes tels que les suivants :


```
{
  "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 : obtenir des données du système

Utilisez la méthode GET pour lire les informations du dispositif.

Si une ressource peut contenir plusieurs objets, vous obtiendrez une liste d'objets dans la réponse. Vous pouvez inclure des paramètres de requête dans l'URL pour contrôler le nombre d'objets renvoyés. La valeur par défaut consiste à retourner 10 objets à partir du début de la liste d'objets.

La procédure suivante explique l'approche générale pour effectuer des appels GET dans l'explorateur d'interface de protocole d'application. Utilisez l'exemple de code pour votre client API.

Procédure

Étape 1 Dans l'explorateur d'interface de protocole d'application, ouvrez une méthode GET (ouvrez d'abord le groupe pour voir les méthodes et les ressources).

Étape 2 Si la méthode que vous souhaitez utiliser nécessite un identifiant d'objet ou parent dans l'URL, utilisez une méthode parente pour obtenir l'identifiant nécessaire.

Par exemple, la méthode GET /objects/networks/{objId} nécessitera l'identifiant d'un objet spécifique. Utilisez la méthode GET /objects/networks pour obtenir une liste des objets réseau, puis recherchez la valeur id (identifiant) de l'objet que vous souhaitez examiner. Notez que dans ce cas, les informations renvoyées dans l'appel GET /object/networks seront les mêmes que celles que vous voyez dans l'appel GET /objects/network/{objId}. Consultez Trouver l'identifiant de l'objet (objId) et l'identifiant du parent (parentId), à la page 8.

Étape 3 Dans la section **Parameters** (Paramètres), configurez les options suivantes :

- **objId** (idObj) : l'identifiant d'objet est toujours requis s'il est nécessaire dans l'URL. Par exemple, 900fac69-7d19-11e7-bf7b-d9417b20e59e.

- **parentId** (idParent) : l'identifiant parent est équivalent à l'identifiant de l'objet, mais concerne simplement un parent qui est plus élevé dans la hiérarchie. Par exemple, GET /policy/intrusion renvoie une liste de politiques de prévention des intrusions, tandis que GET /policy/intrusion/{parentId}/intrusionrules renvoie les règles définies dans l'une de ces politiques. Vous obtiendrez l'identifiant parent en utilisant GET /policy/intrusion.
- **offset** (décalage) : pour les ressources qui prennent en charge plusieurs objets, à partir de quel index dans la liste commence à retourner les objets. La valeur par défaut, 0, indique le début de la liste.
- **limit** (limite) : le nombre maximal d'objets à retourner dans la réponse. La valeur par défaut est 10. La limite maximale est de 1000; si vous saisissez une valeur non valide, elle est automatiquement remplacée par 1000.
- **sort** (tri) : comment trier les objets renvoyés dans la réponse. La méthode de tri par défaut est le tri par ordre alphabétique en fonction de la valeur **name** (nom). Pour modifier la méthode de tri, saisissez le nom de l'attribut dans la ressource à utiliser pour le tri. Par exemple, vous pouvez utiliser **sort=value** (tri=valeur) dans les objets réseau pour effectuer le tri en fonction de l'attribut value (valeur) (c'est-à-dire l'adresse IP). Pour trier en ordre inverse, ajoutez le signe moins, par exemple, **sort=-name** (tri=-nom).
- **filter** (filtre) (non disponible pour toutes les ressources) : retournez les éléments qui correspondent aux critères de filtre uniquement. Le format de la valeur de filtre est {clé}{opérateur}{valeur}, où la clé est un nom d'attribut et la valeur est la chaîne sur laquelle effectuer le tri. Il n'y a pas d'espaces entre les éléments. Les champs sur lesquels vous pouvez appliquer les filtres sont répertoriés dans la description du paramètre **filter** (filtre) dans l'explorateur d'interface de protocole d'application; les champs varient d'un objet à l'autre. Si un objet prend en charge le filtrage sur plusieurs champs, vous pouvez inclure plusieurs valeurs sur le paramètre **filter** (filtre), séparées par un point-virgule (;). Par exemple, vous pouvez filtrer sur gid:1;sid:105 pour GET /policy/intrusionpolicies/{parentId}/intrusionrules. Les opérateurs autorisés sont les suivants :
  - **:** pour égal à. Par exemple, **filter=name:Canada**.
  - **!** pour n'est pas égal à. Par exemple, **filter=name!Canada**.
  - **~** pour similaire à. Par exemple, **Filter=name~United**.
- **filtre=fts~***chaîne* (non disponible pour toutes les ressources) : retourne uniquement les éléments qui correspondent au filtre. L'option **fts~** représente la recherche en texte intégral (full-text search). Tous les attributs de l'objet sont inclus dans la recherche avec la chaîne spécifiée. Vous pouvez inclure une chaîne partielle; utiliser l'astérisque * comme caractère générique pour correspondre à un ou plusieurs caractères est facultatif. N'incluez pas les caractères suivants, ils ne sont pas pris en charge dans la chaîne de recherche : ?~!{}<>:%. Les caractères suivants sont ignorés : ;#&.

  Par exemple, vous pourriez trouver tous les objets réseau qui ont 10. comme premier octet à l'aide de la méthode GET /object/networks?filter=fts~10. Notez qu'il faut de 3 à 5 secondes pour indexer les objets nouvellement créés ou mis à jour, vous devez donc faire une pause avant d'effectuer immédiatement une recherche en texte intégral sur les objets nouveaux ou modifiés.
- **filter=fetchZeroHitcount:**{**true** | **false**} (disponible pour les règles d'accès uniquement) : si vous spécifiez **includeHitcounts=true**, vous pouvez utiliser cette option de filtre pour inclure (**true**) ou exclure (**false**) les règles qui n'ont pas été détectées, c'est-à-dire dont le nombre de résultats est de zéro. La valeur par défaut est **true** (vrai).
- **includeHitCounts** (inclureLeNombreDeRésultats) (disponible pour les règles d'accès uniquement) : sert à indiquer si les renseignements relatifs au nombre de résultats pour les règles dans la politique. Précisez **includeHitcounts=true** (inclureLeNombreDeRésultats=vrai) pour obtenir le nombre de résultats.

Précisez la valeur **false** (faux) (valeur par défaut) pour exclure le nombre de résultats. Les renseignements sur le nombre de résultats sont renvoyés dans l'attribut hitCount (nombreDeRésultats) de l'objet renvoyé.

- **time_duration** (durée) (disponible uniquement pour les rapports sur les tendances) : nombre de secondes dans le passé que le rapport doit inclure. Par exemple, 1800 renvoie un rapport pour les 30 dernières minutes.

Remarque
Considérant que {objId} and {parentId} font partie du chemin de l'URL, ajoutez les paramètres **offset** (décalage), **limit** (limite), **sort** (tri), **filter** (filtre), **includeHitCounts** (inclureLeNombreDeRésultats) et **time_duration** (durée) après un **?** à la fin de l'URL.

Étape 4 Cliquez sur le bouton **Try It Out!** (Essayez!) et examinez la réponse.

Pour les appels réussis (code de retour 200), le corps de la réponse comprend un objet ou une liste d'objets, selon l'appel que vous avez effectué. Pour en savoir plus sur la structure générale et le contenu de la réponse, consultez Tester une méthode et en interpréter les résultats, à la page 39.

Les requêtes GET comprennent une section de pagination. S'il y a plus d'objets que ceux retournés pour l'appel, les valeurs **prev** et **next** indiquent comment obtenir l'ensemble d'objets précédent ou suivant. La valeur **count** indique le nombre total d'objets. La valeur **limit** indique le nombre d'éléments retournés dans la réponse. La valeur **offset** indique la position de début des objets retournés, 0 indiquant le début de la liste.

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

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

POST : créer un objet

Utilisez la méthode POST pour créer un objet pour un type de ressource. Par exemple, utilisez POST pour créer un nouvel objet réseau.

La procédure suivante explique l'approche générale pour le passage d'appels POST dans l'explorateur d'interface de protocole d'application. Utilisez l'exemple de code pour votre client API.

Procédure

Étape 1 Dans l'explorateur d'interface de protocole d'application, ouvrez une méthode POST (ouvrez d'abord le groupe pour afficher les méthodes et les ressources).

Étape 2 Cliquez sur **Model** (Modèle) sous l'en-tête **Response Class** (Classe de réponse) et lisez ce qui concerne les types de données et les valeurs des attributs de ressource.

Étape 3 Sous l'en-tête **Parameters** (Paramètres), configurez les options suivantes, si elles sont disponibles :

- **parentId** (idParent) : identifiant de l'objet parent qui contiendra cet objet. Par exemple, lors de l'ajout d'une règle SSL, l'identifiant de la politique de déchiffrement SSL.
- **at** (à) : pour les objets qui sont contenus dans un objet parent qui organise les objets dans une liste ordonnée, comme les politiques de déchiffrement SSL, l'emplacement auquel insérer l'objet. Utilisez un nombre entier pour indiquer l'emplacement, 0 représentant le début de la liste. La valeur par défaut consiste à insérer le nouvel objet à la fin de la liste.

Remarque
Considérant que {objId} et {parentId} font partie du chemin d'URL, ajoutez le paramètre **at** (à) après un **?** à la fin de l'URL.

Étape 4 Également sous l'en-tête **Parameters** (Paramètres), cliquez sur le modèle JSON affiché dans la colonne **Data Type (Type de données)** > **Example Value (Valeur de l'exemple)** pour le paramètre **body** (corps).

En cliquant dans cette zone, vous chargerez le modèle JSON dans la colonne Value (Valeur) pour le paramètre **body** (corps). Par exemple, en cliquant sur la case de la ressource POST /object/networks, vous chargerez le corps suivant :

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

Étape 5 Saisissez les valeurs requises pour les attributs de l'objet JSON **body** (corps).

Pour les valeurs enum, assurez-vous de lire les valeurs autorisées sous **Response Class (Classe de réponse)** > **Model (Modèle)**. Par exemple, vous pouvez créer un objet réseau pour une adresse de sous-réseau (plutôt qu'une adresse de nom de domaine) en remplissant les valeurs et en modifiant la valeur par défaut de **subType** (sousType) :

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

Étape 6 Cliquez sur le bouton **Try It Out!** (Essayez!) et examinez la réponse.

Examinez la commande **curl** utilisée pour mettre à jour le système. Prenez note des en-têtes supplémentaires. Lorsque vous créez votre client API, vous devez également inclure ces champs d'en-tête et ces valeurs. Par exemple, la commande **curl** pour créer l'exemple d'objet est la suivante. Attardez-vous aux en-têtes Content-Type et Accept.

```
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/dernière version/object/networks'
```

Pour les appels réussis (code de retour 200), le corps de la réponse comprend l'objet complet que vous avez créé, y compris d'autres valeurs générées par le système telles que **version** et **id**. Les valeurs version (version) et ID (identifiant) sont particulièrement importantes, car vous en aurez besoin si vous utilisez ultérieurement la méthode PUT pour modifier l'objet. Pour en savoir plus sur la structure générale et le contenu de la réponse, consultez Tester une méthode et en interpréter les résultats, à la page 39.

Le corps de la réponse comprend également la valeur **links/self**, qui est l'URL de l'objet que vous avez créé. Par exemple, voici le corps de la réponse pour l'objet d'exemple.

```
{
  "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/dernière version/object/networks/
f6d8da49-7ed5-11e7-9bfd-27136f5686ad"
  }
}
```

PUT : modifier un objet existant

Utilisez la méthode PUT pour modifier les attributs d'un objet existant. Par exemple, utilisez la méthode PUT pour modifier l'adresse contenue dans un objet réseau existant sans en changer l'identifiant.

La méthode PUT remplace l'objet entier. Elle ne vous permet pas de ne modifier qu'un seul attribut. C'est pourquoi vous devez vous assurer que votre objet JSON comprend les anciennes valeurs que vous souhaitez conserver.

La procédure suivante explique l'approche générale pour l'exécution d'appels PUT dans l'explorateur d'interface de protocole d'application. Utilisez l'exemple de code pour votre client API.

Avant de commencer

Utilisez la méthode GET pour la ressource parente afin d'obtenir une copie de l'état existant de l'objet, comme décrit dans GET : obtenir des données du système, à la page 41.

Vous devez avoir les valeurs correctes pour les paramètres suivants au moins, ainsi que toutes les valeurs fournies par l'utilisateur que vous ne souhaitez pas modifier.

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

Procédure

Étape 1 Dans l'explorateur d'interface de protocole d'application, ouvrez une méthode PUT (ouvrez d'abord le groupe pour afficher les méthodes et les ressources).

Étape 2 Sous l'en-tête **Parameters** (Paramètres), configurez les options suivantes :

- **objId** (idObj) : la valeur **id** de l'objet. Par exemple, 900fac69-7d19-11e7-bf7b-d9417b20e59e.
- **parentId** (idParent) : pour les objets qui sont contenus dans un autre objet, l'identifiant de l'objet parent qui contiendra cet objet. Par exemple, lors de la modification d'une règle SSL, l'identifiant de la politique de déchiffrement SSL.
- **at** (à) : pour les objets qui sont contenus dans un objet parent qui organise les objets dans une liste ordonnée, comme les politiques de déchiffrement SSL, l'emplacement auquel insérer l'objet. Utilisez un nombre entier pour indiquer l'emplacement, 0 représentant le début de la liste. La valeur par défaut consiste à insérer l'objet à la fin de la liste.

Remarque
Considérant que {objId} et {parentId} font partie du chemin d'URL, ajoutez le paramètre **at** (à) après un **?** à la fin de l'URL.

Étape 3 Également sous l'en-tête **Parameters** (Paramètres), cliquez sur le modèle JSON affiché dans la colonne **Data Type (Type de données)** > **Example Value (Valeur de l'exemple)** pour le paramètre **body** (corps).

En cliquant dans cette zone, vous chargerez le modèle JSON dans la colonne Value (Valeur) pour le paramètre **body** (corps). Par exemple, en cliquant sur la case de la ressource PUT /object/networks, vous chargerez le corps suivant. Notez que cela est légèrement différent de la version de la méthode POST pour la même ressource : le corps de la méthode PUT comprend l'attribut **version**.

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

Étape 4 Saisissez les valeurs requises pour les attributs de l'objet JSON **body** (corps).

Assurez-vous de répliquer les anciennes valeurs que vous ne souhaitez pas modifier.

Pour les valeurs enum, assurez-vous de lire les valeurs autorisées sous **Response Class (Classe de réponse)** > **Model (Modèle)**. Reportez l'ancienne valeur, sauf si vous remplacez l'objet par un sous-type différent. Par exemple, le modèle PUT par défaut pour un objet de réseau a HOST (HÔTE) comme **subType**, mais si vous modifiez un objet de sous-réseau, assurez-vous de remplacer **subType** par NETWORK (RÉSEAU).

Par exemple, pour mettre à jour l'adresse IP de sous-réseau dans un objet réseau, répétez toutes les anciennes valeurs pour tous les attributs à l'exception de **value**. Saisissez la nouvelle adresse de sous-réseau dans **value**.

```
{
  "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",
}
```

Étape 5 Cliquez sur le bouton **Try It Out!** (Essayez!) et examinez la réponse.

Examinez la commande **curl** utilisée pour mettre à jour le système. Prenez note des en-têtes supplémentaires. Lorsque vous créez votre client API, vous devez également inclure ces champs d'en-tête et ces valeurs. Par exemple, la commande **curl** pour mettre à jour l'exemple d'objet est la suivante. Attardez-vous aux en-têtes Content-Type et Accept.

```
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/dernière version/object/
networks/f6d8da49-7ed5-11e7-9bfd-27136f5686ad'
```

Pour les appels réussis (code de retour 200), le corps de la réponse comprend l'objet complet que vous avez mis à jour. Notez que la valeur de la version change, mais l'ID d'objet (et donc le lien ou l'attribue « self ») reste le même. La version change chaque fois que vous modifiez un objet. Pour en savoir plus sur la structure générale et le contenu de la réponse, consultez Tester une méthode et en interpréter les résultats, à la page 39.

Remarque
Si vous n'avez pas apporté de modification à l'objet, c'est-à-dire que l'objet mis à jour est identique à sa version précédente, le système ne traitera pas la demande et renverra un code 204 indiquant que rien n'a été modifié pour cette ressource.

Par exemple, voici le corps de la réponse pour la mise à jour de l'objet exemple.

```
{
  "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/dernière version/object/networks/
f6d8da49-7ed5-11e7-9bfd-27136f5686ad"
  }
}
```

DELETE : supprimer un objet créé par l'utilisateur

Utilisez la méthode DELETE pour supprimer un objet que vous, ou un autre utilisateur, avez créé. Par exemple, utilisez DELETE pour supprimer un objet réseau que vous n'utilisez plus.

Vous ne pouvez pas supprimer des objets définis par le système ou des objets qui doivent obligatoirement exister.

Vous ne pouvez pas non plus supprimer un objet qui est actuellement utilisé par un autre objet, comme un objet réseau utilisé dans une règle d'accès. Pour supprimer un objet en cours d'utilisation, modifiez d'abord tous les objets qui l'utilisent, puis supprimez-le.

La procédure suivante explique l'approche générale pour effectuer des appels DELETE dans l'explorateur d'interface de protocole d'application. Utilisez l'exemple de code pour votre client API.

Avant de commencer

Utilisez la méthode GET pour la ressource parente afin d'obtenir une copie de l'état existant de l'objet, comme décrit dans GET : obtenir des données du système, à la page 41.

Vous devez disposer de l'identifiant de l'objet (la valeur **id**) pour le supprimer.

Procédure

Étape 1 Dans l'explorateur d'interface de protocole d'application, ouvrez une méthode DELETE (ouvrez d'abord le groupe pour afficher les méthodes et les ressources).

Étape 2 Sous l'en-tête **Parameters** (Paramètres), saisissez la valeur **id** (identifiant) de l'objet dans le champ **objId** (idObj). Par exemple : f6d8da49-7ed5-11e7-9bfd-27136f5686ad.

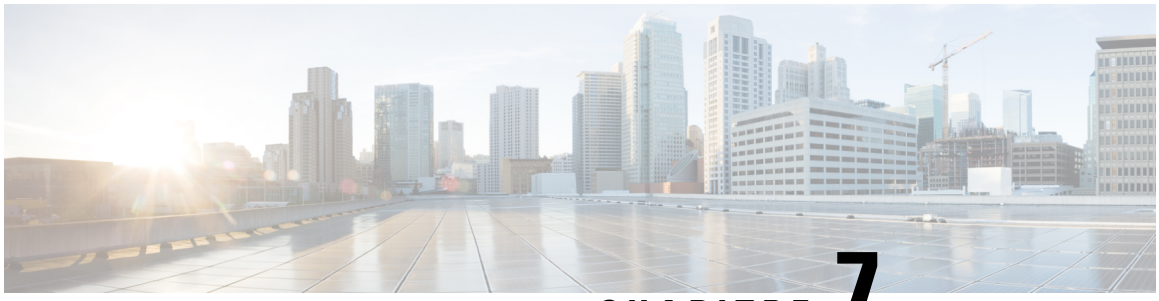
Si l'objet est contenu dans un conteneur, vous devez également saisir l'identifiant de l'objet parent dans le champ **parentId** (idParent).

Étape 3 Cliquez sur le bouton **Try It Out!** (Essayez!) et examinez la réponse.

Examinez la commande **curl** utilisée pour supprimer l'objet du système. Prenez note des en-têtes supplémentaires. Lorsque vous créez votre client API, vous devez également inclure ces champs d'en-tête et ces valeurs. Par exemple, la commande **curl** pour supprimer l'objet exemple est la suivante. Prenez note de l'en-tête Accept (Accepter).

```
curl -X DELETE --header 'Accept: application/json'
'https://ftd.example.com/api/fdm/dernière version/object/
networks/f6d8da49-7ed5-11e7-9bfd-27136f5686ad'
```

Pour les appels réussis (code de retour 204 « No Content » [Aucun contenu]), vous recevrez un corps de réponse vide. C'est le résultat attendu.



CHAPITRE 7

Déploiement des modifications de configuration

Déploiement des modifications de configuration

Bien que les appels POST, PUT et DELETE mettent directement à jour le dispositif Défense contre les menaces, ils ne sont pas actifs immédiatement. Vous devez déployer les modifications de configuration avant que le dispositif utilise vos nouveaux paramètres lors du traitement du trafic.

Procédure

Étape 1 Utilisez la ressource POST /operational/deploy dans le groupe Deployment (Déploiement) pour lancer un déploiement.

Par exemple, la commande **curl** ressemblerait à ce qui suit :

```
curl -X POST --header 'Content-Type: application/json'
--header 'Accept: application/json'
'https://ftd.example.com/api/fdm/dernière version/operational/deploy'
```

Étape 2 Évaluez la réponse pour vérifier que la tâche de déploiement a été mise en attente.

Une bonne réponse (code d'état 200) ressemble à ce qui suit. Notez l'état.

```
{
  "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/dernière version/operational/deploy/
62bf405f-796c-11e8-8640-a9156b92ec49"
```

```
  }
}
```

Remarque
Les attributs **cliErrorMessage** (messageD'ErreurDuClient) et **name** (nom) ont été ajoutés dans la v2 de l'API; ils ne sont pas inclus dans les réponses de la v1.

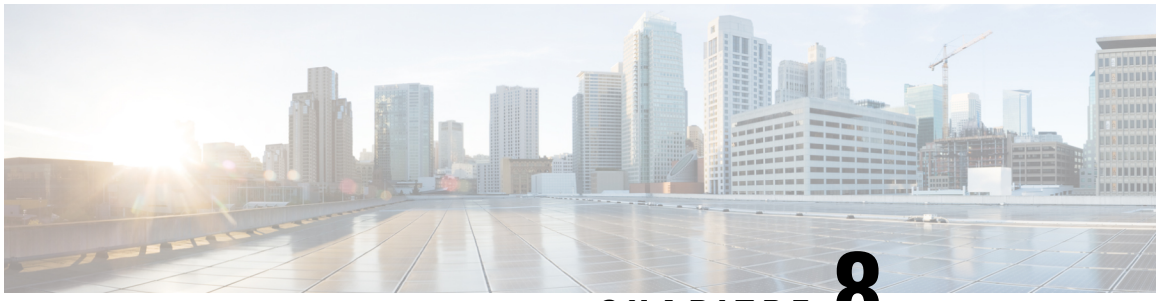
Étape 3 Utilisez la ressource GET /operational/deploy/{objId} pour vérifier l'état de la tâche.

Par exemple, la commande **curl** ressemblerait à ce qui suit :

```
curl -X GET --header 'Accept: application/json'
'https://ftd.example.com/api/fdm/dernière version/operational/deploy/
a7a227fb-82ab-11e7-8186-0dc471ff0672'
```

La réponse pourrait ressembler à ce qui suit. Notez que l'état, DEPLOYED (DÉPLOYÉ), indique que la tâche s'est terminée avec succès. Le paramètre modifiedObjects (objetsModifiés) répertorie les objets qui ont été modifiés dans la tâche de déploiement. Dans ce cas, il y a une seule modification, apportée à un objet réseau nommé new-network (nouveau-réseau).

```
{
  "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/dernière version/operational/deploy/
62bf405f-796c-11e8-8640-a9156b92ec49"
  }
}
```



CHAPITRE 8

Importation/exportation de la configuration

Exigence relative à la version : pour utiliser l'importation/exportation de la configuration, vous devez exécuter la version 6.5(0) ou supérieure de Défense contre les menaces, et la version v4 ou supérieure de l'API REST Défense contre les menaces.

Vous pouvez exporter la configuration d'un dispositif géré avec le gestionnaire d'appareil et l'importer dans le même dispositif ou dans un autre dispositif compatible. Par exemple, vous pouvez utiliser l'importation/exportation de la configuration pour reproduire une configuration de référence sur plusieurs dispositifs similaires, puis utiliser le gestionnaire d'appareil sur chaque dispositif pour configurer les caractéristiques uniques de chaque dispositif.

À propos de l'importation et de l'exportation de la configuration

Lorsque vous gérez le dispositif Défense contre les menaces localement, avec le gestionnaire d'appareil ou par l'intermédiaire du Security Cloud Control, vous pouvez exporter la configuration du dispositif à l'aide de l'API Défense contre les menaces. Cette méthode ne fonctionne pas avec un dispositif géré par le Cisco Secure Firewall Management Center.

Lorsque vous exportez la configuration, le système crée un fichier zip. Vous pouvez ensuite télécharger le fichier zip sur votre ordinateur. La configuration elle-même est représentée sous forme d'objets définis à l'aide de paires attribut-valeur dans un fichier texte au format JSON. Vous pouvez modifier le fichier avant de le réimporter dans le même dispositif ou dans un autre.

Ainsi, vous pouvez utiliser un fichier d'exportation pour créer un modèle que vous pouvez déployer sur d'autres dispositifs de votre réseau.

Lorsque vous importez des objets, vous avez également la possibilité de définir les objets directement dans la commande d'importation plutôt que dans un fichier de configuration. Cependant, vous ne devriez définir directement des objets que dans les cas où vous importez un petit nombre de modifications.

Les rubriques suivantes expliquent plus en détail les importations/exportations de configuration.

Ce qui est inclus dans le fichier d'exportation

Lorsque vous procédez à une exportation, vous devez préciser quelles configurations doivent être incluses dans le fichier d'exportation. Une exportation complète comprend tout ce qui se trouve dans le fichier d'exportation zip. En fonction de ce que vous choisissez d'exporter, le fichier d'exportation zip peut comprendre ce qui suit :

- Paires attribut-valeur qui définissent chaque objet configuré. Tous les éléments configurables sont modélisés en tant qu'objets, et pas seulement ceux qui sont appelés « objects » (objets) dans le gestionnaire d'appareil.
- Si vous avez configuré le VPN d'accès à distance, les paquets AnyConnect et tous les autres fichiers référencés, tels que les fichiers XML du profil client, le fichier XML DAP et les paquets Hostscan.
- Si vous avez configuré des stratégies de fichiers personnalisées, toute liste de nettoyage référencée ou toute liste de détection personnalisée.

Comparer l'importation/exportation et la sauvegarde/restauration

L'importation/exportation de la configuration n'est pas identique à la sauvegarde/restauration.

- La sauvegarde/restauration sert à récupérer les sinistres. Vous pouvez restaurer une sauvegarde sur un dispositif uniquement si celui-ci est du même modèle et exécutant la même version de logiciel que le dispositif à partir duquel la sauvegarde a été effectuée. Principalement, il s'agit de récupérer la « dernière bonne » configuration sur le même dispositif ou de restaurer la configuration sur un dispositif de remplacement.
- Importer/Exporter permet de conserver tout ou partie d'une configuration. Vous pouvez utiliser un fichier d'exportation pour restaurer la configuration sur un dispositif après l'avoir réinitialisé. Sinon, vous pouvez utiliser le fichier d'exportation comme modèle et modifier le contenu avant de l'importer dans un autre dispositif. Importer/Exporter vous permet de configurer rapidement un nouveau dispositif jusqu'à un certain niveau de base, afin de le déployer plus rapidement dans votre réseau. Avec certaines limites, vous pouvez même importer un fichier sur différents modèles de dispositif, par exemple, d'un châssis Firepower 2120 à un 2130. Si le fichier d'importation ne comprend que des objets pris en charge sur tous les modèles de dispositifs, il ne devrait pas y avoir beaucoup de restrictions à l'importation. La seule restriction est que le dispositif doit utiliser la même version d'API utilisée pour le fichier d'exportation.

Stratégies pour l'importation/exportation

Voici quelques façons d'utiliser l'importation/exportation.

- **Créez un modèle pour les nouveaux dispositifs.** Configurez votre dispositif de modèle avec la référence dont vous avez besoin, puis exportez la configuration complète. Vous pourrez ensuite importer cette configuration dans de nouveaux dispositifs, puis utiliser l'API gestionnaire d'appareil ou Défense contre les menaces pour apporter les modifications nécessaires. Vous pouvez également modifier le modèle avant l'importation pour apporter ces modifications, par exemple aux adresses IP de chaque interface. Notez que l'exportation complète comprend l'objet ManagementIP (type=managementip); en supposant que vous avez déjà configuré l'adresse de gestion et la passerelle sur le dispositif cible, vous devez supprimer cet objet du fichier d'exportation lorsque vous créez le modèle pour le nouveau dispositif, ou vous remplacerez les informations d'adressage de gestion.

- **Déployer les modifications de configuration d'un dispositif sur d'autres dispositifs similaires.** Par exemple, lors de la modification de la configuration du dispositif A, vous créez quelques nouveaux objets réseau et règles de contrôle d'accès. Vous pouvez ensuite exporter les modifications en cours et importer ces modifications dans le dispositif B. Après avoir déployé la configuration sur les deux dispositifs, ils exécutent les mêmes nouvelles règles.
- **Réappliquez la configuration après avoir recréé l'image du système.** Le fait de recréer l'image d'un dispositif efface la configuration. Si vous exportez la configuration complète pour la première fois, vous pouvez l'importer après avoir terminé la recréation de l'image.
- **Appliquer les configurations ciblées.** Étant donné que vous pouvez modifier ou même créer manuellement un fichier d'exportation, vous pouvez supprimer tous les objets, à l'exception de ceux que vous souhaitez importer dans un autre dispositif. Par exemple, vous pouvez créer un fichier de configuration qui contient un ensemble d'objets réseau et l'utiliser pour importer le même groupe d'objets réseau dans tous vos dispositifs Défense contre les menaces.

Lignes directrices pour les importations/exportations de configuration

- Pendant une tâche d'exportation, le système s'assure que l'écriture soit verrouillée sur la base de données de configuration. Vous ne pouvez pas utiliser l'API ni le gestionnaire d'appareil, pour modifier la configuration tant que la tâche n'est pas terminée. Cependant, vous pouvez consulter la configuration dans le gestionnaire d'appareil ou utiliser des appels GET dans l'API pendant la tâche d'exportation.
- Pendant une tâche d'importation, le système s'assure que l'écriture et la lecture soient verrouillées sur la base de données de configuration. Vous ne pouvez pas utiliser l'API ni le gestionnaire d'appareil, pour consulter ou modifier la configuration tant que la tâche n'est pas terminée.
- La configuration importée est ajoutée à la configuration existante. Vous ne pouvez pas effacer la configuration du dispositif et la remplacer par la configuration importée. Si vous devez réinitialiser la configuration du dispositif avant l'importation, vous pouvez accéder à l'interface de ligne de commande du dispositif et exécuter la commande **configure manager delete**, suivie de la commande **configure manager local**. Seule la configuration de l'interface de gestion sera conservée.
- Vous pouvez importer un fichier dans un dispositif uniquement si celui-ci exécute la même version de l'API que celle définie dans l'attribut apiVersion (versionDeL'Api) dans l'objet metadata (métadonnées) contenu dans le fichier.
- La version de SRU doit être la même sur les dispositifs d'exportation et d'importation, sinon l'importation échouera.

Importer et exporter des configurations

Le processus d'importation et d'exportation commence par l'exportation de la configuration à partir d'un dispositif géré localement. Vous pouvez ensuite télécharger le fichier d'exportation, et éventuellement le modifier, avant de le téléverser sur le même dispositif ou dans un dispositif compatible. Les rubriques suivantes expliquent chaque étape.

Exporter la configuration

Utilisez la méthode POST /action/configexport pour créer et démarrer une tâche d'exportation de la configuration.

Procédure

Étape 1 Créez le corps de l'objet JSON pour la tâche d'exportation.

Voici un exemple d'objet JSON à utiliser avec cet appel.

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

Les attributs sont :

- **diskFileName** (nomDuFichierDeDisque) : (facultatif.) Le nom du fichier zip d'exportation. Si vous ne spécifiez pas de nom, le système en générera un pour vous. Même si vous spécifiez un nom, le système peut ajouter des caractères au nom pour garantir l'unicité. Le nom a une longueur maximale de 60 caractères.
- **encryptedKey** (cléChiffrée) : (facultatif.) Une clé de chiffrement pour le fichier zip. Si vous ne voulez pas chiffrer le fichier, ignorez ce champ et spécifiez plutôt « "doNotEncrypt": true » (« nePasChiffrer » : vrai). Si vous spécifiez une clé, vous devrez l'utiliser pour ouvrir le fichier zip après l'avoir téléchargé sur votre ordinateur. Remarquez que le fichier de configuration exporté expose des clés secrètes, des mots de passe et d'autres données sensibles en texte clair (car, sinon, ils ne peuvent pas être importés), de sorte qu'il est probable que vous souhaitiez y appliquer une clé de chiffrement pour protéger les données sensibles. Le système utilise le chiffrement AES 256.
- **doNotEncrypt** (nePasChiffrer) : (facultatif.) Indique si le fichier d'exportation doit être chiffré (false [faux]) ou non (true [vrai]). La valeur par défaut est false (faux), ce qui signifie que vous devez spécifier un attribut encryptionKey (cléDeChiffrement) non vide. Si vous spécifiez true (vrai), l'attribut encryptionKey (cléDeChiffrement) est ignoré.
- **configExportType** (typeD'ExportationDeConfiguration) : l'une des valeurs enum (énumération) suivantes :
  - **FULL_EXPORT** (EXPORTATION_COMPLÈTE) : comprend toute la configuration dans le fichier d'exportation. Il s'agit du paramètre par défaut.
  - **PARTIAL_EXPORT** (EXPORTATION_PARTIELLE) : n'inclure que les objets et leurs objets descendants qui sont identifiés dans la liste entityIds (identifiantsD'Identité). Les objets non exportables ne sont pas inclus, même si vous spécifiez leurs identités. Tous les objets définis par l'utilisateur sont exportables.

  - **PENDING_CHANGE_EXPORT** (EXPORTATION_DES_MODIFICATIONS_EN_ATTENTE) : n'inclure que les objets qui n'ont pas encore été déployés, c'est-à-dire les modifications en attente.

- **DeployedObjectsOnly** (ObjetsDéployésSeulement) : (facultatif.) Indique si les objets doivent être inclus dans le fichier d'exportation uniquement s'ils ont été déployés. C'est-à-dire qu'il ne faut pas inclure les modifications en attente. Cet attribut est ignoré pour les tâches PENDING_CHANGE_EXPORT, car ces tâches n'incluent que les objets non déployés. La valeur par défaut est false (faux), ce qui signifie que tout changement en attente est inclus dans l'exportation. Spécifiez true (vrai) pour exclure les changements en attente.
- **entityIds** (identifiantsD'Entité) : une liste d'identités séparées par des virgules avec un ensemble d'objets de point de départ, entre [crochets]. La liste est nécessaire pour une tâche PARTIAL_EXPORT (EXPORTATION_PARTIELLE). Chaque élément de cette liste peut être une valeur UUID ou une paire attribut-valeur correspondant à des motifs tels que « **id=***uuid-value* », « **type**=*objet-type* » ou « **name**=*objet*-name ». Par exemple, **« type=networkobject »**

  Le **type** peut être soit une entité leaf, telle qu'un networkobject (objetréseau), ou un alias d'un ensemble de types leaf. Voici quelques alias de types typiques : network (NetworkObject et NetworkObjectGroup), port (tous les types de port, de protocole et de groupe TCP/UDP/ICMP), url (objets et groupes URL), ikepolicy (politiques IKE V1/V2), ikeproposal (propositions Ike V1/V2), identitysource (toutes les sources d'identité), certificate (tous les types de certificats), object (tous les types d'objets/groupes qui seraient répertoriés dans le gestionnaire d'appareil sur la page Objects), interface (toutes les interfaces réseau, s2svpn (tous les types de VPN site à site), ravpn (tous les types de VPN RA), vpn (s2svpn et ravpn).

  Tous ces objets et leurs descendants référentiels sortants seront inclus dans le fichier de sortie PARTIAL_EXPORT. Remarquez que tous les objets non exportables seront exclus de la sortie même si vous spécifiez leurs identités. Utilisez la méthode GET pour les types de ressources appropriés afin d'obtenir les UUID, les types ou les noms des objets cibles.

  Par exemple, pour exporter tous les objets réseau, ainsi qu'une règle d'accès nommée myaccessrule et deux objets identifiés par UUID, vous pouvez spécifier :

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

- **jobName** (nomDeLaTâche) : (facultatif.) Un nom pour la tâche d'exportation. Donner un nom à la tâche d'exportation permet de la retrouver plus facilement lorsque vous récupérez le statut de la tâche.
- **type** (type) : le type de tâche, qui est toujours **scheduleconfigexport** (exportationdelaconfigurationducalendrier).

Exemple :

Dans l'exemple suivant, on effectue une exportation complète vers le fichier export-config-1 et on accepte les valeurs par défaut pour tous les autres attributs :

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

Étape 2 Postez l'objet.

Par exemple, la commande curl ressemblerait à ce qui suit :

```
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/dernière version/action/configexport'
```

Étape 3 Vérifiez la réponse.

Vous devriez obtenir un code réponse de 200. Si vous publiez l'objet JSON minimum, le corps de la réponse ressemblera à ce qui suit : Si vous spécifiez une clé de chiffrement, elle sera masquée dans la réponse.

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

Vérifier l'état de la tâche d'exportation

La réalisation d'une tâche d'exportation prend un certain temps. Plus la configuration est volumineuse, plus de temps la tâche prendra. Vérifiez le statut de la tâche pour vous assurer qu'elle se réalise complètement avant d'essayer de télécharger le fichier.

La manière la plus simple de récupérer l'état est d'utiliser GET /jobs/configexportstatus. Par exemple, la commande curl ressemblerait à ce qui suit :

```
curl -X GET --header 'Accept: application/json'
'https://10.89.5.38/api/fdm/dernière version/jobs/configexportstatus'
```

Une tâche terminée avec succès renverra un état semblable à ce qui suit.

```
{
  "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/dernière version
/jobs/configexportstatus/1f0aad8e-62a5-11e9-8b8d-bb1ebb4d1300"
  }
}
```

Vous pouvez également utiliser la méthode GET /jobs/configexportstatus/{objId} pour récupérer l'état d'une tâche spécifique. Vous obtiendrez l'identifiant d'objet dans le champ **id** (identifiant) de l'objet de réponse.

Télécharger le fichier d'exportation

Lorsqu'une tâche d'exportation est terminée, le fichier d'exportation est écrit sur le disque système et est appelé fichier de configuration. Vous pouvez télécharger ce fichier d'exportation sur votre ordinateur à l'aide de la méthode GET /action/downloadconfigfile/{objId}. Pour obtenir une liste des fichiers disponibles, utilisez la méthode GET /action/configfiles.

Remarque Avec GET /action/downloadconfigfile/{objId}, vous spécifiez généralement le nom du fichier comme identifiant de l'objet. Vous pouvez également spécifier l'identifiant de l'objet ConfigExportStatus associé au fichier.

Procédure

Étape 1 Obtenez une liste des fichiers de configuration sur le disque.

La liste des fichiers de configuration comprend les fichiers d'exportation et tous les fichiers que vous avez téléversés pour l'importation.

La commande curl ressemblerait à ce qui suit :

```
curl -X GET --header 'Accept: application/json'
'https://10.89.5.38/api/fdm/dernière version/action/configfiles'
```

La réponse montrerait une liste d'items, chacun étant un fichier de configuration. Par exemple, la liste suivante affiche 2 fichiers. Notez que la valeur **id** (identifiant) affiche « default » (par défaut) pour tous les fichiers. Ignorez la valeur id (identifiant) et utilisez plutôt **diskFileName** (nomDeFichierDeDisque).

```
{
  "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/dernière version/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/dernière version/action/configfiles/default"
      }
    }
  ],
```


Étape 2 Téléchargez le fichier utilisant le diskFileName comme l'identifiant de l'objet.

La commande curl ressemblerait à ce qui suit :

```
curl -X GET --header 'Accept: application/octet-stream'
'https://10.89.5.38/api/fdm/dernière version/action/downloadconfigfile/export-config-2.zip'
```

Le fichier est téléchargé dans votre dossier de téléchargement par défaut. Si vous utilisez la méthode GET à partir de l'explorateur API et que votre navigateur est configuré pour demander l'emplacement du téléchargement, vous serez invité à enregistrer le fichier.

Un téléchargement réussi résultera en un code de retour 200 et aucun corps de réponse.

Modifier le fichier de configuration exporté

Après avoir téléchargé le fichier de configuration, vous pouvez le décompresser et ouvrir le fichier texte qui contient les objets. WordPad formatera le contenu de manière plus facile à lire que NotePad. Vous pouvez également utiliser d'autres éditeurs de texte installés sur votre dispositif, le cas échéant. Vous pouvez même créer votre propre fichier de configuration à partir de zéro, mais vous devrez quand même exporter la configuration pour comprendre la structure du fichier.

Les rubriques suivantes expliquent les exigences pour le fichier texte.

Exigences minimales du fichier de configuration

Un fichier de configuration doit comporter les éléments minimaux suivants :

- Placez les objets dans le fichier entre [crochets]. Le fichier entier utilise la notation JSON standard et est un tableau d'objets.
- Enveloppez chaque objet dans des {accolades}.
- Utilisez des virgules pour séparer les objets dans le fichier de configuration. C'est-à-dire que l'accolade marquant la fin d'un objet doit être suivie d'une virgule, sauf pour l'objet final.
- Le premier objet du fichier doit être un objet de métadonnées. Le moyen le plus simple d'obtenir les bons attributs d'objet consiste à exporter la configuration à partir d'un dispositif du modèle souhaité. Par exemple, voici l'objet de métadonnées d'un dispositif Cisco Secure Firewall Threat Defense Virtual.

Avant d'importer le dispositif, vous pouvez modifier les types de configuration et d'exportation, et si vous le souhaitez, supprimer l'attribut generatedOn (généréLe).

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

- L'objet de métadonnées doit spécifier la valeur de type de configuration (configType) appropriée.
  - FULL_CONFIG : ce fichier texte comprend la configuration complète du dispositif.
  - DELTA_CONFIG : ce fichier texte comprend une configuration partielle, peut-être même quelques objets seulement.
- L'option exportType (typeD'Exportation) est l'une des options suivantes : FULL_EXPORT, PARTIAL_EXPORT, PENDING_CHANGE_EXPORT.
- Si vous effectuez une importation de configuration complète, l'objet de métadonnées doit spécifier les attributs suivants : hardwareModel (modèleDuMatériel), softwareVersion (versionDuLogiciel), apiVersion (versionDeL'Api).
- Vous pouvez écrire des objets sur une ou plusieurs lignes, mais ne placez pas de lignes vides ou de lignes de commentaires entre les attributs d'un objet. Les commentaires ne sont pas autorisés dans le fichier.
- Bien que les objets soient exportés en ordre de dépendance, où un objet référencé par un autre objet est défini en premier, il n'est pas obligatoire de maintenir cet ordre dans le fichier de configuration d'importation. Le système résoudra automatiquement les relations lors de l'importation, en supposant que les noms d'objet et les identifiants se résolvent correctement entre les objets dépendants.

Structure de base des objets de classe d'enveloppe d'identité

Le fichier de configuration utilise des objets d'enveloppe d'identité pour définir tout objet ConfigEntity ou ManagementEntity qui peut être exporté ou importé. Voici la structure de base d'un objet d'enveloppe d'identité :

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

L'objet contient les attributs suivants :

- **type** (type) : la valeur est toujours **identitywrapper** (enveloppeD'Identité).
- **data** (données) : il s'agit de l'ensemble des paires attribut-valeur qui définissent l'objet à partir de la configuration, comme un objet réseau, une règle de contrôle d'accès, etc. Les attributs nécessaires à cette collecte dépendent du modèle du type d'objet et d'action spécifiques que vous effectuez. Mettez les paires attribut-valeur entre accolades. Séparez les attributs du tableau de données par des virgules.

- **parentName** (nomDuParent) : (si nécessaire.) Un nombre limité d'objets sont des ContainedObjects (ObjetsContenus), qui ont une relation avec un objet qui les contient. Cela inclut par exemple les critères d'accès, les règles NAT manuelles et les sous-interfaces. Pour ces éléments, le parentName (nomDuParent) spécifie le nom de l'objet le contenant (le parent). Précisez cet attribut pour les objets contenus. Ne le spécifiez pas pour les objets non contenus. Vous devrez peut-être également spécifier un index pour ces objets.

  Vous pouvez en fait omettre cet attribut si le parent est un objet unique (c'est-à-dire que vous ne pouvez pas en créer plusieurs), comme l'objet AccessPolicy (PolitiqueD'Accès), et le système sera en mesure de comprendre la référence.

- **oldName** (ancienNom) – (si nécessaire.) Si vous renommez un objet existant, vous pouvez spécifier l'ancien nom sur cet attribut et le nouveau nom dans l'attribut **name** des attributs de données. L'action doit être EDIT (MODIFIER) pour pouvoir utiliser cet attribut.

- **action** (action) : l'action à entreprendre par rapport à l'objet défini. Dans les exportations complètes, l'action est toujours **CREATE** (CRÉER). Pour les modifications en attente ou les exportations partielles, d'autres actions peuvent être **EDIT** (MODIFIER) ou **DELETE** (SUPPRIMER).

  Lorsque vous modifiez le fichier pour l'importation, précisez l'action souhaitée. Notez que si vous spécifiez CREATE (CRÉER), mais que l'objet existe déjà, l'action est remplacée par EDIT (MODIFIER); si l'objet n'existe pas, EDIT (MODIFIER) est remplacé par CREATE (CRÉER). L'action DELETE (SUPPRIMER) n'est pas modifiée. Les références aux objets sont résolues en fonction du type et du nom de l'objet, ou du type d'objet et de l'ancien nom, ou du type d'objet et du nom parent.

  - CREATE (CRÉER) : vous créez un nouvel objet. Vous devez spécifier les attributs de données requis lors de la publication d'un objet. Notez que si le **name** (nom) correspond à un objet existant du type spécifié, l'action passe automatiquement à EDIT (MODIFIER).

    Notez que si vous créez un objet et le référencez à partir d'autres objets, par exemple en définissant un objet réseau et en l'utilisant dans une règle d'accès, le **name** (nom) d'objet doit être le bon dans la référence.

  - EDIT (MODIFIER) : vous mettez à jour un objet. Vous devez spécifier les attributs de données requis lors de la mise en place d'un objet, à l'exception de la version et de l'identifiant. Le nom et le type d'objet sont utilisés pour déterminer l'objet à mettre à jour, et l'attribut de version est toujours ignoré.

  - DELETE (SUPPRIMER) : vous supprimez l'objet. Vous devez préciser les attributs **type** (type) et **name** (nom) dans les données de l'objet.

- **index** (index) : (facultatif; nombre entier.) Pour les objets qui font partie d'une liste ordonnée, comme les règles de contrôle d'accès et les règles NAT manuelles, il s'agit de la position de l'objet dans la politique. Si vous créez une nouvelle règle et que vous ne spécifiez pas de valeur d'index, la règle est ajoutée à la fin de la politique en tant que dernière règle. Si vous modifiez la règle, le système conservera la position existante de la règle.

Exemple : Modifier un objet de réseau pour l'importation dans un dispositif différent

Chaque objet est structuré comme l'objet qui suit, un objet d'hôte de réseau qui définit l'adresse IP du serveur de journalisation du système :

```
{"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"}}
```

Supposons que vous ayez exporté cet objet à partir d'un dispositif et que vous souhaitez importer l'objet dans un autre dispositif, mais le nouveau dispositif doit utiliser un serveur de journalisation du système à une adresse différente, 192.168.5.15. Puisque vous allez créer un nouvel objet, supprimez les attributs **version** (version) et **id** (identifiant) de l'attribut data (données). Vous pouvez également supprimer **isSystemDefined** (estDéfiniParLeSystème) (dont la valeur par défaut est false [faux]) et **dnsResolution** (résolutionDuDns) (qui ne concerne qu'un objet FQDN uniquement). Le nouvel objet résultant ressemblerait à ce qui suit :

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

En haut du fichier, vous devez garder (ou ajouter) l'objet metadata (métadonnées). Vous pouvez également ajouter des retours à la ligne pour faciliter la lecture et la vérification du contenu du fichier. Ainsi, le fichier de configuration complet devrait ressembler à ce qui suit :

```
[
{"hardwareModel":"Cisco Firepower Threat Defense for VMWare",
 "type":"metadata",
 "configType":"DELTA_CONFIG",
 "apiVersion":"dernière version",
 "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"}}
]
```

Téléverser le fichier d'importation

Avant de pouvoir importer un fichier de configuration dans un dispositif, vous devez d'abord téléverser le fichier sur ce dernier. Vous pouvez téléverser des fichiers zip ou texte. Vous pouvez inclure des paquets et des profils client AnyConnect si vous utilisez un fichier zip.

Utilisez la ressource POST /action/uploadconfigfile pour téléverser le fichier. Le nom a une longueur maximale de 60 caractères.

- Si vous utilisez cette méthode à partir de l'explorateur d'interface de protocole d'application, cliquez sur le bouton **Choose File** (Choisir un fichier) à côté de l'attribut **fileToUpload** (fichierÀTéléverser) pour sélectionner le fichier dans le lecteur de votre ordinateur.

- Si vous utilisez la méthode de votre propre programme, la charge utile de la demande doit contenir un seul file-item (élément-de-fichier) avec un champ file-name (nom-de-fichier). L'extension de file-name (nom-de-fichier) doit être .txt ou .zip et le format du contenu du fichier réel doit être cohérent avec l'extension de fichier.

La commande curl ressemblerait à ce qui suit :

```
curl -F 'fileToUpload=@./import-1.txt'
'https://10.89.5.38/api/fdm/dernière version/action/uploadconfigfile'
```

Un transfert réussi génère un code de retour 200 et un corps de réponse semblable à ce qui suit, qui affiche le nom du fichier sur le système Défense contre les menaces (**diskFileName**), dont vous avez besoin pour la tâche d'importation.

```
{
  "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/dernière version/action/uploadconfigfile/default"
  }
}
```

Importer la configuration et vérifier l'état de la tâche

Après avoir téléversé un fichier de configuration dans le système Défense contre les menaces, vous pouvez importer les objets définis dans le fichier de configuration dans la configuration de Défense contre les menaces. Utilisez la méthode POST /action/configimport.

Lorsque vous importez des objets, vous avez également la possibilité de définir les objets directement dans la commande d'importation plutôt que dans un fichier de configuration. Cependant, vous ne devriez définir directement des objets que dans les cas où vous importez un petit nombre de modifications, comme à un ou deux objets réseau seulement.

Procédure

Étape 1 Créez le corps de l'objet JSON pour la tâche d'importation.

Voici un exemple d'objet JSON à utiliser avec cet appel.

```
{
  "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"
}
```

Les attributs sont :

- **diskFileName** (nomDuFichierDeDisque) : le nom du fichier de configuration zip ou .txt à importer.
- **encryptedKey** (cléChiffrée) : la clé utilisée pour chiffrer le fichier zip, le cas échéant. Ne précisez pas de clé si le fichier de configuration n'est pas chiffré.
- **preserveConfigFile** : (facultatif.) Indique s'il faut conserver la copie du fichier de configuration importé sur le disque de Défense contre les menaces après une tâche d'importation réussie. Précisez true (vrai) pour conserver le fichier, false (faux) pour que le fichier soit supprimé du disque de Défense contre les menaces. La valeur par défaut est false (faux).
- **autoDeploy** (déploiementAutomatique) : (facultatif.) Indique s'il faut démarrer automatiquement une tâche de déploiement si l'importation est réussie. Les objets importés sont mis en attente avant que les modifications entrent en vigueur, et ces dernières ne seront actives que lorsque vous les déploierez. Précisez « true » (vrai) pour démarrer la tâche de déploiement automatiquement. Si vous indiquez false (faux), vous devrez déployer manuellement vos modifications. La valeur par défaut est false (faux).
- **allowPendingChange** (autoriserLesModificationsEnAttente) : (facultatif.) Indique s'il faut autoriser la tâche d'importation à démarrer si des modifications existantes sont en attente. Si vous définissez cet attribut sur « true » (vrai) et autoDeploy (déploiementAutomatique) sur « true » (vrai), la tâche de déploiement automatique inclura toutes les modifications, préexistantes et importées. Si vous définissez cet attribut sur false (faux), la tâche d'importation ne s'exécutera pas si des modifications sont en attente. La valeur par défaut est false (faux).
- **excludeEntities** (exclureEntités) : (facultatif.) Une liste de chaînes correspondant à des objets qui identifient les objets qui ne doivent pas être importés. Vous devez préciser cet attribut uniquement si le fichier d'importation comprend des éléments que vous ne souhaitez pas importer (c'est-à-dire que vous avez décidé de ne pas les supprimer du fichier que vous avez téléversé). Chaque élément de cette liste a un schéma ressemblant à « **id=***uuid-value* », « **type**=*object-type* » ou « **name**=*object-name* ». Les objets d'entrée correspondant à l'un de ces schémas seront exclus de l'importation.

  Le **type** peut être soit une entité leaf, telle qu'un networkobject (objetréseau), ou un alias d'un ensemble de types leaf. Voici quelques alias de types typiques : network (NetworkObject et NetworkObjectGroup), port (tous les types de port, de protocole et de groupe TCP/UDP/ICMP), url (objets et groupes URL), ikepolicy (politiques IKE V1/V2), ikeproposal (propositions Ike V1/V2), identitysource (toutes les sources d'identité), certificate (tous les types de certificats), object (tous les types d'objets/groupes qui seraient répertoriés dans le gestionnaire d'appareil sur la page Objects), interface (toutes les interfaces réseau, s2svpn (tous les types de VPN site à site), ravpn (tous les types de VPN RA), vpn (s2svpn et ravpn).

Par exemple, pour exclure de l'importation tous les objets réseau et deux autres objets identifiés par le nom myobj et un UUID, spécifiez :

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

- **inputEntities** (entitésDeSaisie) : si vous avez un petit nombre d'objets à importer, vous pouvez les définir dans la liste d'objets d'entrée d'entrée plutôt que dans un fichier de configuration. Pour utiliser cet attribut, vous ne pouvez pas inclure l'attribut diskFileName, sinon vous devrez le définir sur « null » (nul).
- **jobName** (nomDeLaTâche) : (facultatif.) Un nom pour la tâche d'exportation. Donner un nom à la tâche d'exportation permet de la retrouver plus facilement lorsque vous récupérez le statut de la tâche.
- **type** (type) : le type de tâche, qui est toujours **scheduleconfigimport** (importationdelaconfigurationducalendrier).

Exemple :

L'exemple suivant importe le fichier de configuration nommé import-1.txt :

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

Étape 2 Postez l'objet.

Par exemple, la commande curl ressemblerait à ce qui suit :

```
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/dernière version/action/configimport'
```

Étape 3 Vérifiez la réponse.

Vous devriez obtenir un code réponse de 200. Si vous publiez l'objet JSON minimum, le corps de la réponse ressemblera à ce qui suit : Si vous spécifiez une clé de chiffrement, elle sera masquée dans la réponse.

```
{
  "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/dernière version
/action/configimport/7e2b52d8-6725-11e9-abb5-5dec35337506"
  }
}
```

Étape 4 Utilisez GET /jobs/configimportstatus pour vérifier l'état de la tâche d'importation.

Vous pouvez également utiliser GET /jobs/configimportstatus/{objId} pour obtenir l'état d'une tâche d'importation. Pour objId (idObj), utilisez la valeur jobHistoryUuid du corps de la réponse à votre appel POST /action/configimport.

La commande curl ressemblerait à ce qui suit :

```
curl -X GET --header 'Accept: application/json'
'https://10.89.5.38/api/fdm/dernière version/jobs/configimportstatus'
```

Le corps de la réponse pourrait ressembler à ce qui suit pour une importation réussie. Si l'importation échoue, vous devrez peut-être modifier le fichier pour corriger les erreurs de formatage ou de contenu, puis réessayer.

```
{
      "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/dernière version
/jobs/configimportstatus/7e360139-6725-11e9-abb5-078014531401"
  }
}
```

Prochaine étape

Si vous définissez la valeur autoDeploy (déploiementAutomatique) sur false (faux), vous devrez exécuter une tâche de déploiement pour intégrer les modifications importées. Utilisez la méthode POST /operational/deploy. Si vous avez défini la valeur sur true (vrai), la configuration devrait avoir été déployée avec succès. Dans le gestionnaire d'appareil ou l'API (GET /operational/auditevents), vous pouvez vérifier le journal d'audit. La tâche de déploiement s'intitule « Post Configuration Import Deployment » (Déploiement après l'importation de la configuration).

Remarque Certaines fonctionnalités nécessitent des licences particulières. Par exemple, un dispositif doit avoir une licence pour toutes les fonctionnalités VPN d'accès à distance. Toutefois, le processus d'importation ne valide pas les licences. Par conséquent, si vous importez des objets pour une fonctionnalité contrôlée par licence sur un dispositif qui ne dispose pas de la licence requise, la tâche de déploiement échouera. Si vous rencontrez ce problème, attribuez les licences requises au dispositif ou supprimez les objets.

Supprimer les fichiers d'importation/exportation non requis

Si vous n'avez plus besoin d'un fichier de configuration, qu'il soit créé par une tâche d'exportation ou que vous l'ayez téléversé pour l'importation de la configuration, vous pouvez le supprimer.

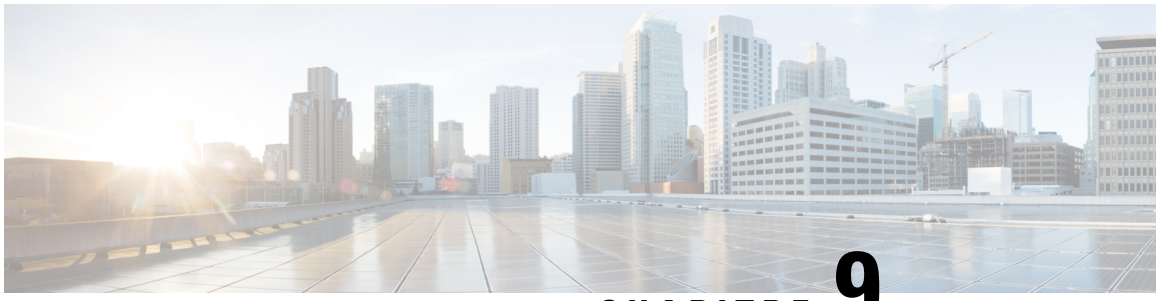
Utilisez la méthode DELETE /action/configfiles/{objId}, en utilisant le nom du fichier comme valeur objId.

Par exemple, pour supprimer le fichier nommé export-config-2.zip, la commande curl serait la suivante :

```
curl -X DELETE --header 'Accept: application/json'
'https://10.89.5.38/api/fdm/dernière version/action/configfiles/export-config-2.zip'
```

La commande a réussi si vous recevez un code de retour 204 et aucun corps de réponse.

Vous pouvez utiliser GET /action/configfiles pour confirmer que le fichier a été supprimé.



CHAPITRE 9

Pour en savoir plus et consulter des exemples

Pour en savoir plus et consulter des exemples

Vous pouvez trouver des informations supplémentaires sur l'utilisation de l'API sur les sites suivants :

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

  Ce site comprend des informations de référence pour les ressources, y compris des exemples d'appels de script Bash et de code Python. Il y a un menu pour sélectionner la version de l'API que vous utilisez. Sélectionnez la version appropriée pour afficher les bonnes informations de référence. Il y a également une liste de tous les codes d'erreur et de tous les messages d'erreur uniques que vous êtes susceptible de voir lors de votre utilisation de l'API.

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

  Ce site contient des exemples de bout en bout pour la configuration de certaines fonctionnalités, comme le mode haute disponibilité, y compris des exemples de code.

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

  Ce site comprend des vidéos, des modules d'apprentissage et des ateliers pour vous aider à apprendre comment utiliser l'API.

Pour en savoir plus et consulter des exemples

À propos de la traduction

Cisco peut fournir des traductions du présent contenu dans la langue locale pour certains endroits. Veuillez noter que des traductions sont fournies à titre informatif seulement et, en cas d’incohérence, la version anglaise du présent contenu prévaudra.