# 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).

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 3.

### 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).

**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 1.

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 1.

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 3.

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 1.

**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 3.

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.

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.