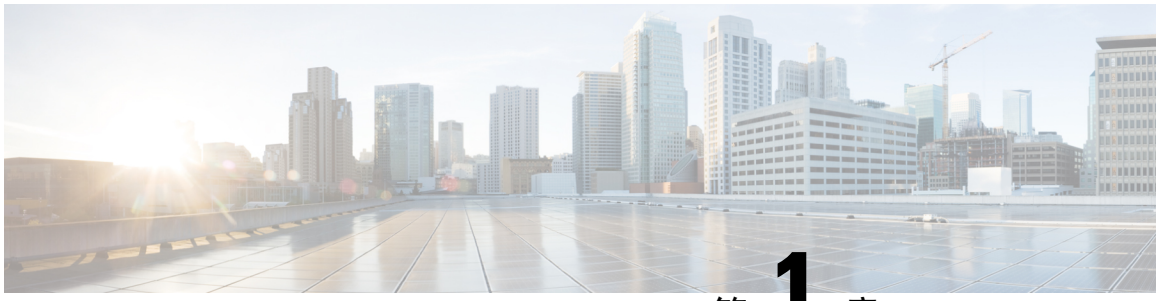
思科 Cisco Secure Firewall Threat Defense REST API 指南

首次发布日期: 2018 年 3 月 29 日

上次修改日期: 2023 年 7 月 19 日

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

目录



第 1 章

关于 Cisco Secure Firewall Threat Defense REST API

您可以通过 HTTPS 使用 Cisco Secure Firewall Threat Defense 具象状态传输 (REST) 应用编程接口 (API) 利用客户端程序与 威胁防御 设备交互。REST API 使用 JavaScript 对象表示法 (JSON) 格式表示对象。

Secure Firewall 设备管理器 包括一个 API Explorer（该资源管理器对可供您编程使用的所有资源和 JSON 对象进行说明）。Explorer 提供有关各对象中属性值对的详细信息，您可以尝试不同的 HTTP 方法，确保了解使用各资源所需的编码。API Explorer 还提供各资源所需的 URL 示例。

您还可以在 https://developer.cisco.com/site/ftd-api-reference/上找到参考信息和示例。

API 有其自己的版本号。无法保证设计用于一个 API 版本的客户端能够准确无误地用于将来的版本，且无需修改程序。

此编程指南的受众

本指南假设您对编程有基本认识并对 REST API 和 JSON 有特定理解。如果您不熟悉这些技术，请首先阅读有关 REST API 的一般指南。

支持的 HTTP 方法

仅可使用以下 HTTP 方法。不支持其他方法。

- GET - 从系统读取数据。

- POST - 创建新对象。
- PUT - 修改现有对象。使用 PUT 时，必须包含整个 JSON 对象。无法选择性地更新对象内的个别属性。
- DELETE - 删除用户定义的对象。

API 基准 URL

确定给定 威胁防御 设备基准 URL 的最简单方法是在 API Explorer 中尝试 GET 方法，且仅从结果中删除 URL 的对象部分。

例如，可执行 GET /object/networks，并在请求 URL 下的返回输出中看到与下面类似的内容：

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

URL 服务器名称部分是 威胁防御 设备的主机名或 IP 地址，因您的设备而不同，代替“ftd.example.com”。在本例中，从路径中删除 /object/networks 以获取基准 URL：

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

所有资源调用均将此 URL 作为请求 URL 的基础。

如果更改了 HTTPS 数据端口，则必须在 URL 中包含该自定义端口。例如，如果您将端口更改为 4443，则 URL 应为 https://ftd.example.com:4443/api/fdm/v1/

URL 中的“v”元素是 API 版本，通常随软件版本变化。例如，威胁防御 版本 6.3.0 的 API 版本是 v2，因此基准 URL 为：

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

注释 从 威胁防御 6.4 开始，可以在路径中使用 **latest** 代替 v 元素，从而无需在 API 调用中更新路径。例如，https://ftd.example.com/api/fdm/latest/。**latest** 别名将解析为设备支持的最新 API 版本。

在 API Explorer 中，如果滚动至页面底部，则可看到有关基准 URL（无服务器名称）和 API 版本的信息。

保护 REST API 的 SSL/TLS 通信

威胁防御设备附带自签证书，以便可以发起与设备的 HTTPS 通信。但是，由于证书并非由已知证书颁发机构 (CA) 签名，因此任何 SSL/TLS 访问尝试均将认为连接不安全。

使用浏览器进行连接时，系统会提示您接受自签证书，但 curl 等命令会拒绝该证书。对于 curl，可以通过添加 **--insecure** 关键字来避免证书检查失败。例如：

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

您应该做的第一件事是获取 威胁防御设备的 CA 签名设备证书。然后，使用 设备管理器 或 API 将此证书分配为管理证书。随后，SSL/TLS 证书检查应该不会失败，且无需在 API 调用中使用不安全的通信。

过程

步骤 1 使用 **POST /object/internalcertificates** 资源上传 CA 签名的设备证书。

步骤 2 使用 **PUT /devicesettings/default/webuicertificates/{objId}** 资源将此证书设置为管理证书。

使用 **GET /devicesettings/default/webuicertificates** 资源确定 Web UI 证书的对象 ID。

步骤 3 使用 **POST /operational/deploy** 资源部署更改。

确定支持的 API 版本

您可以使用 GET/api/versions (ApiVersions) 方法确定设备支持的 API 版本。此方法不需要身份验证，也不包括路径中的版本元素。例如：

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

“ftd.example.com”替换为 威胁防御 设备的主机名或 IP 地址。

此方法返回您可以使用的 API 版本的列表。例如：

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

版本字符串与您在后续 API 调用的 URL 中使用的版本字符串相同。如果使用的是 **latest** 而不是特定版本标识符，则可避免为后续版本更新调用。但是，使用此方法无法克服更改调用中使用的对象模型的问题，这可能需要随着不同版本的发行而进行调整。

通常情况下，下一步是获取访问令牌，如使用 OAuth 对 REST API 客户端进行身份验证，第 13 页中所述。

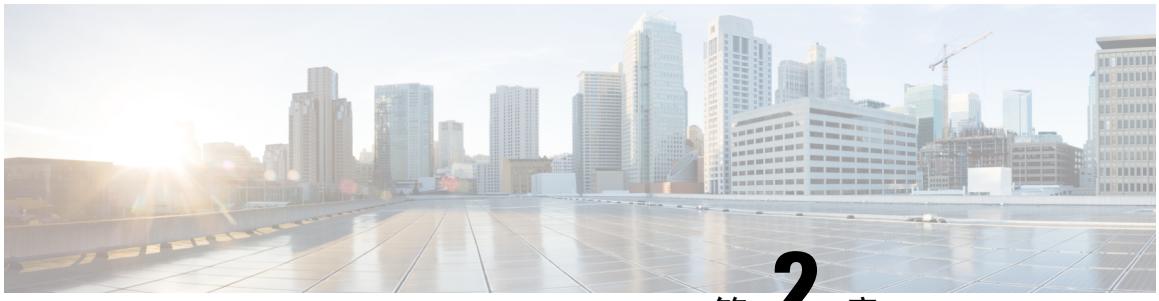
API 版本向后兼容性

威胁防御 API 版本随 威胁防御 软件的每个主要版本而变化。新功能会影响要添加或更改的功能的 API 调用。

但是，许多功能在各版本之间不会发生变化。例如，与网络和端口对象相关的 API 在新版本中通常保持不变。

从 威胁防御 版本 6.7 开始，如果某项功能的 API 资源模型在各版本之间未更改，则 威胁防御 API 可以接受基于较早 API 版本的调用。即使功能模型发生了变化，如果有将旧模型转换为新模型的合理方法，旧的调用也仍然可用。例如，可以在 v6 系统上接受 v5 调用。如果在调用中使用“latest”作为版本号，则在此场景中，这些“较早”的调用将被解释为v6调用，因此，是否要利用向后兼容性取决于 API 调用的构建方式。

如果某个功能模型在API版本之间发生了更改，导致无法支持向后兼容，则您会收到一条错误消息，您需要检查这些错误并为这些特定调用更新代码。



第 2 章

API Explorer

使用 API Explorer 了解 REST API。您还可以测试各种方法和资源，以验证您的配置是否正确。您可以将 JSON 模型复制粘贴到代码中作为起始点。

提示 API Explorer 旨在帮助您了解 API。通过 API Explorer 测试调用需要创建访问锁定，而这可能会干扰常规运行。建议您在非生产设备上使用 API Explorer。

打开 API Explorer

API Explorer 介绍了可用于编程用途的所有资源和 JSON 对象。Explorer 提供有关各对象中属性值对的详细信息，您可以尝试不同的 HTTP 方法，确保了解使用各资源所需的编码。

过程

步骤 1 使用浏览器打开系统主页，例如 https://ftd.example.com。

步骤 2 登录 设备管理器。

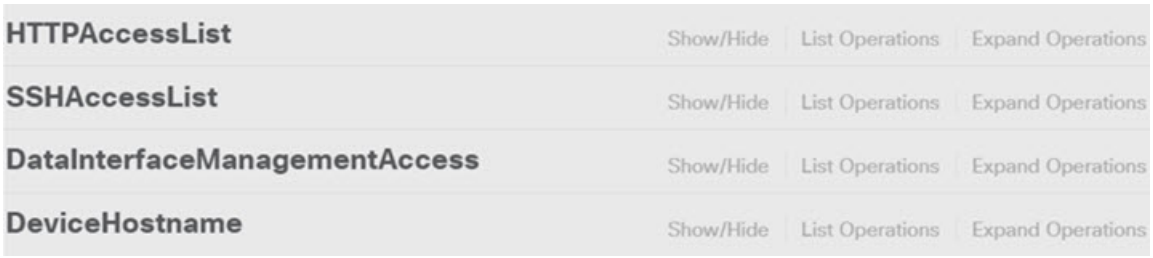
步骤 3 （6.4 及更早版本。）编辑 URL，使其指向 /#/api-explorer，例如 https://ftd.example.com/#/api-explorer。

步骤 4 （6.5 及更高版本。）点击“更多选项”按钮 (⋮) 并选择 **API Explorer**。

系统会在单独的选项卡或窗口中打开 API Explorer，具体取决于您的浏览器设置。

探索 API Explorer

进入 API Explorer 后，您会看到资源组列表。这些组包括 API 中可用的资源。下图显示的是示例列表的一小部分。

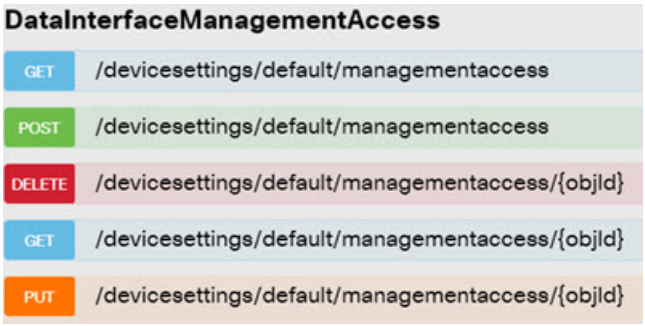


这些组名是链接：点击链接可打开组，从而查看可用于组中资源的方法。各组还包括右侧的以下命令：

- **显示/隐藏**打开和关闭组。这与点击组名的作用相同。最初，展开操作仅显示方法（与**列表操作**相同），但系统会记住最后的展开状态（在关闭前）并重新打开到相同的展开级别。
- **列表操作**显示可用于组中各资源的 HTTP 方法。信息包括各资源的统一资源定位符 (URL) 模板的相对路径。路径变量由标准约定表示：{*variable*}。需要用适当的值替换 {*variable*}，包括大括号。必须将基准 URL 添加至这一相对路径；请参阅API 基准 URL，第 2 页。

  点击操作 URL 模板，查看该方法的完整文档。
- **展开操作**打开组中可用的所有 HTTP 方法和资源。

一些组具有许多子资源。例如，DataInterface ManagementAccess 组包括 /devicesettings/default/managementaccess 的 GET、POST 和 DELETE 操作，以及 /devicesettings/default/managementaccess/{objId} 的 GET 和 PUT 操作。



查看有关资源的文档

每个资源中的属性记录在 API Explorer 中。

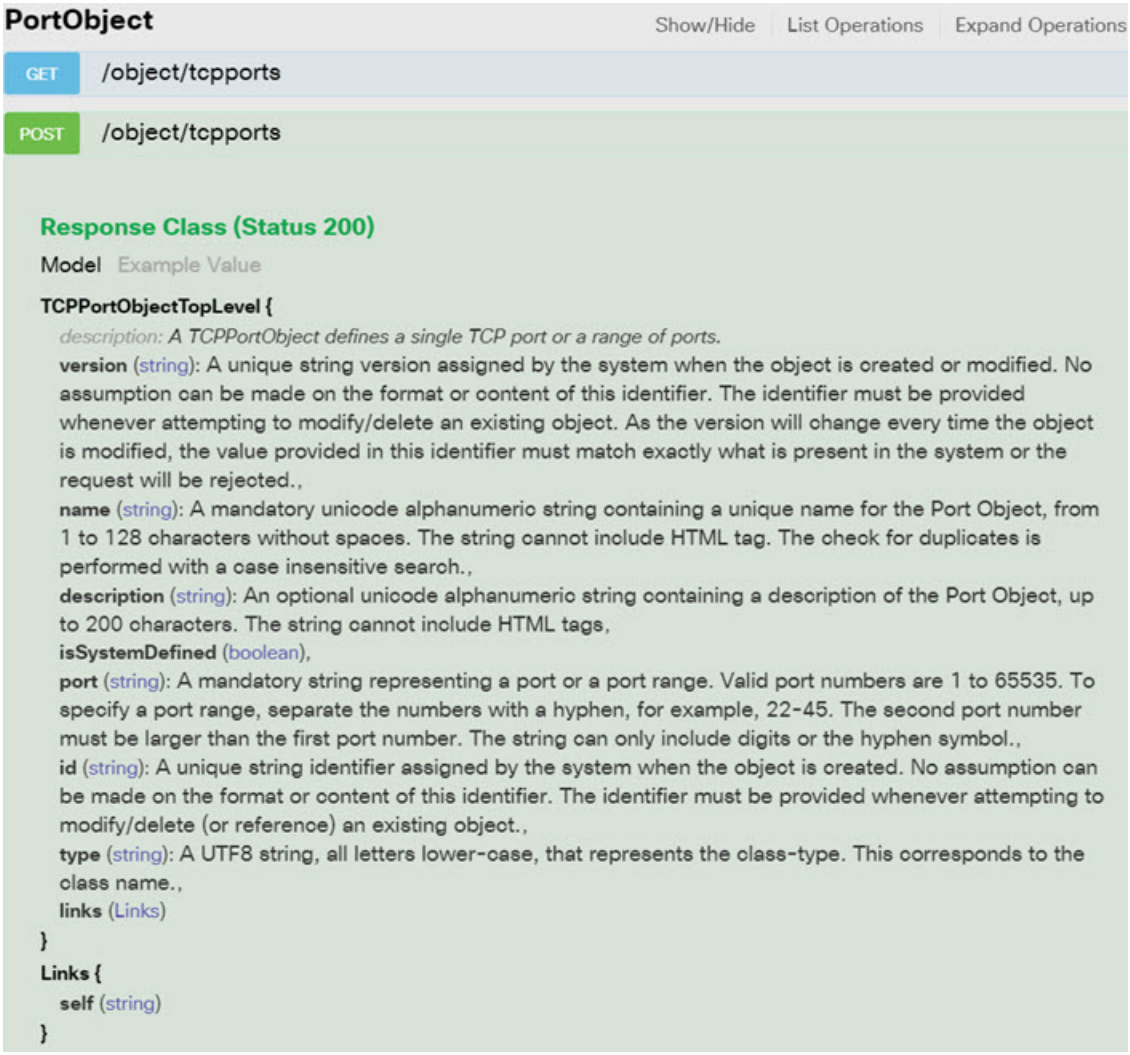
过程

步骤 1 详细了解特定资源和您感兴趣的方法。

步骤 2 在**响应类 (Response Class)** 部分，点击**模型 (Model)** 选项卡。

模型将列出属性以及说明和数据类型。对于 GET，还有用于可能返回的对象的分页选项。如果对象数量超出返回的对象数量，您将获得下一批和上一批对象的 URL。

例如，下图显示了 POST /object/tcpports 方法和资源，并且选中了**模型**选项卡。默认情况下，系统将选中**示例值 (Example Value)** 选项卡，因此必须始终点击**模型 (Model)** 才能查看文档。



查找对象 ID (objId) 和父 ID

某些资源需要 URL 中的对象 ID 或相关父 ID，如下所示：

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

在大多数情况下，可使用资源层次结构中较高级的 GET 方法获取对象或父 ID。object/parent ID 是给定对象的 **id** 参数 UUID。

例如，GET/object/networks 返回当前定义的所有网络对象的列表。您可能需要执行多次调用才可遍历列表获取所需对象，或纳入 **limit** 查询参数以增加为调用返回的对象数。各对象具有以下格式；对象 ID 突出显示。

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

注释 在某些情况下，{objId} 出现在层次结构的顶层。在这些情况下，有时可输入任何对象 ID 值并获得相同的结果。在其他情况下，检查对象模型文档以获取有效对象类型的信息；ID 是有效的对象类型之一。这些总是 GET 调用。例如，GET /operational/systeminfo/{objId} 和 GET /operational/featureinfo/{objId}。

查看错误目录并评估错误消息

作为 REST API，如果对不存在的对象执行 GET，则系统会返回标准 HTTP 错误代码，例如 404。

此外，系统还包括许多错误消息，这些错误消息会更具体地解释错误。如果 API 调用导致错误，则响应正文可能包括这些更具体的消息。

例如，如果尝试对以下网络对象 **POST /object/networks** 操作，会收到错误消息。在此例中，您正在尝试指定网络，但却忘了包含网络掩码（即，该值应为 10.10.10.0/24 或 10.10.10.0/255.255.255.0）：

```
{
  "name": "test-network",
  "subType": "NETWORK",
```

```
  "value": "10.10.10.0",
  "type": "networkobject"
}
```

结果将产生 HTTP 响应代码 422，以及包含特定错误消息的响应正文：

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

以下操作步骤介绍如何查看可在响应正文中返回的可能错误消息列表。

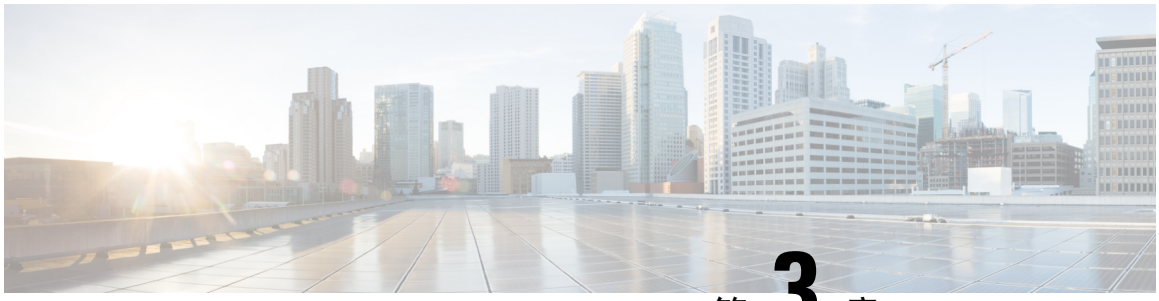
过程

步骤 1 在 设备管理器 中，点击“更多选项”按钮 (⋮) 并选择 **API Explorer**。

步骤 2 点击目录中的**错误目录 (Error Catalog)**。

这些消息具有以下组成部分。

- **类别** - 一般消息类型。此值显示在错误响应正文的 **key** 属性中。类别包括“验证”、“常规”和“部署”。
- **代码** - 标识错误消息的唯一字符串。此值显示在错误响应正文的 **code** 属性中。可以使用浏览器的“在页面上查找”功能在目录中找到使用该值的消息。
- **消息** - 解释错误的特定消息。此值显示在错误响应正文的 **description** 属性中。消息中的变量表示为 {0}、{1} 等。



第 3 章

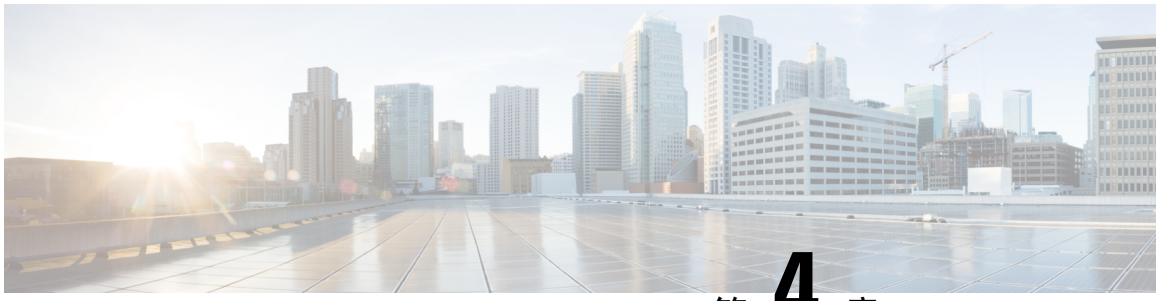
使用 REST API 的一般流程

使用 REST API 的一般流程

通常，客户端使用以下迭代过程与 威胁防御 设备进行交互：

1. 获取访问令牌以对 API 调用进行身份验证。请参阅 API 客户端身份验证过程概述，第 13 页。
2. 需建立 JSON 负载（只需读取数据时除外）
3. 使用资源的统一资源定位符 (URL) HTTPS 调用传输 JSON 负载。
4. 使用返回的 JSON 响应。
5. 如果更改配置，请部署更改。请参阅部署配置更改，第 45 页。

使用 REST API 的一般流程



第 4 章

使用 OAuth 对 REST API 客户端进行身份验证

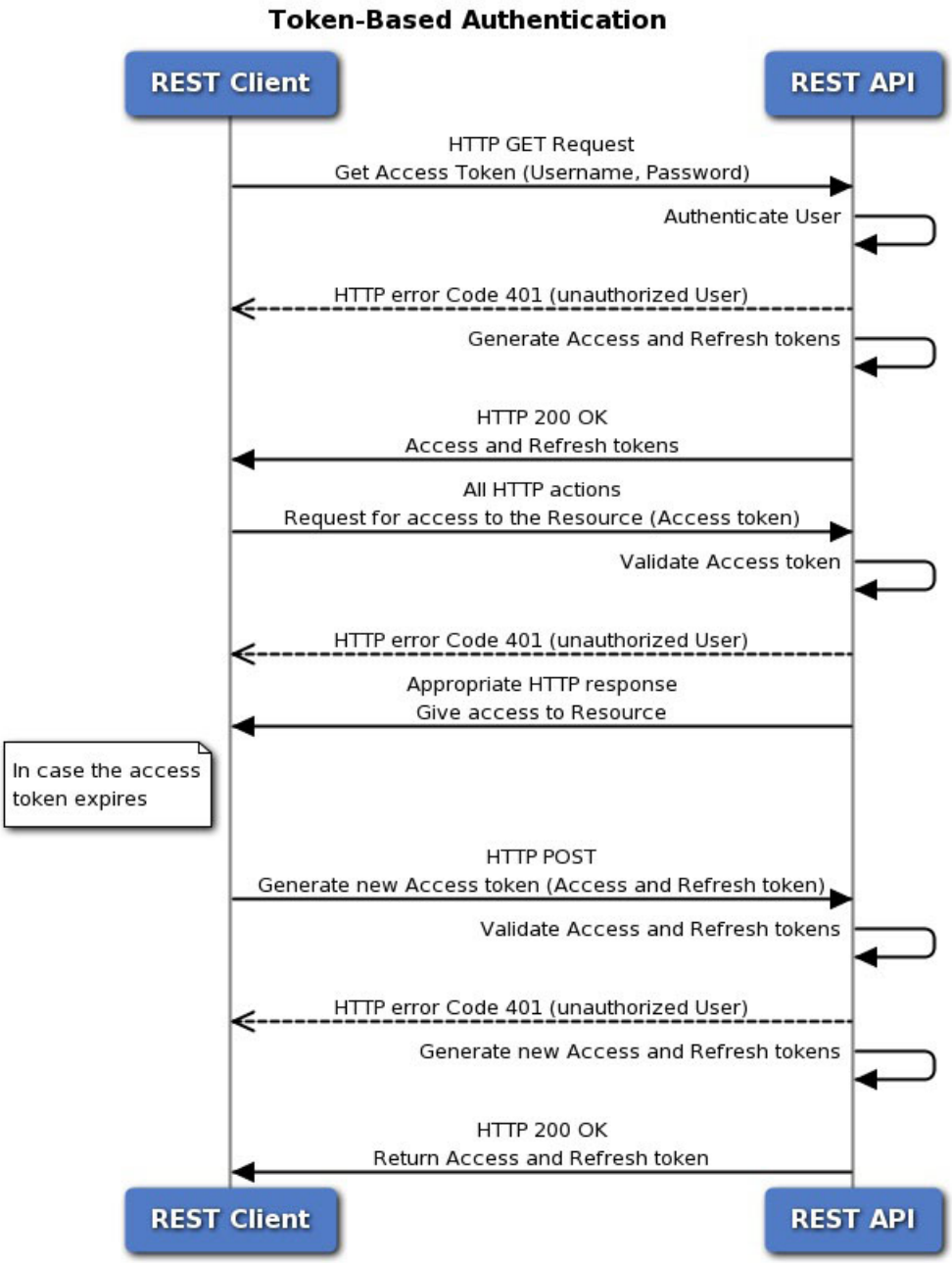
威胁防御 REST API 使用 Oauth 2.0 对来自 API 客户端的调用进行身份验证。OAuth 是一种基于访问令牌的方法，且 威胁防御 将 JSON Web 令牌用于此方案。相关标准如下：

- RFC6749，OAuth 2.0 授权框架，https://tools.ietf.org/html/rfc6749。
- RFC7519，JSON Web 令牌 (JWT)，https://tools.ietf.org/html/rfc7519。

以下主题介绍获取和使用所需令牌的方法。

API 客户端身份验证过程概述

以下是如何使用威胁防御设备对 API 客户端进行身份验证的全方位解析。



开始之前

每个令牌代表一个 HTTPS 登录会话，该会话计入 API 会话和 设备管理器 会话。最多可以有 5 个活动 HTTPS 会话。如果超过此限制，则最早的会话（设备管理器 登录或 API 令牌）将过期以允许建立新会话。因此，重要的是，您只能获取所需的令牌，并重复使用每个令牌直到其到期，然后续约。为每个 API 调用获取新令牌将导致严重的会话中断，并可能使用户无法访问 设备管理器。这些限制不适用于 SSH 会话。

过程

步骤 1 使用所需的方法对 API 客户端用户进行身份验证。

您的客户端有义务对用户进行身份验证，并确保其有权访问和修改威胁防御 设备。如果希望根据授权权限提供不同的功能，则需要将其构建到客户端中。

例如，如果希望允许只读访问，则必须设置所需的身份验证服务器、用户账户等。然后，当拥有只读权限的用户登录客户端时，必须确保只发出 GET 调用。在 API v1 中，这种类型的变量访问不能由 威胁防御 设备自身控制。自 API v2 起，如果您使用外部用户且未根据用户授权修整调用，那么如果用户授权和尝试的调用之间不匹配，系统会出错。

在 v1 中，与设备通信时，必须在 威胁防御 设备上使用 **admin** 用户账户。**管理员**账户对所有用户可配置的对象拥有完全的读/写授权权限。

步骤 2 使用**管理员**账户，根据用户名/密码请求密码授予的访问令牌。

请参阅请求密码授予的访问令牌，第 15 页。

步骤 3 或者，为客户端请求自定义访问令牌。

使用自定义令牌，可以明确为该令牌请求一个有效期并分配一个使用者名称。请参阅请求自定义访问令牌，第 17 页。

步骤 4 在“授权：无记名”报头中使用 API 调用的访问令牌。

请参阅在 API 调用中使用访问令牌，第 19 页。

步骤 5 在访问令牌到期之前，刷新该令牌。

请参阅刷新访问令牌，第 19 页。

步骤 6 完成后，如果令牌尚未过期，系统将撤销该令牌。

请参阅撤销访问令牌，第 21 页。

请求密码授予的访问令牌

每个 REST API 调用都必须包括一个身份验证令牌，以验证调用方是否被授权执行请求的操作。最初，需要通过提供**管理员**用户名/密码获取访问令牌。这被称为密码授予的访问令牌，即 grant_type = 密码。

过程

步骤 1 为密码授予的访问令牌授权创建 JSON 对象。

```
{
```

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

指定**管理员**用户名和正确的密码，例如:

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

步骤 2 使用 POST /fdm/token 获取访问令牌。

例如，**curl** 命令将如下所示：

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

步骤 3 从响应中检索访问令牌和刷新令牌。

良好响应（状态代码 200）如下所示：

```
{
  "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
}
```

其中：

- **access_token** 是需要包含在 API 调用中的不记名令牌。请参阅在 API 调用中使用访问令牌， 第 19 页。
- **expires_in** 是访问令牌自颁发时起的有效时间（单位：秒）。
- **refresh_token** 是用于刷新请求的令牌。请参阅刷新访问令牌， 第 19 页。

- **refresh_expires**_in 是刷新令牌的有效时间（单位：秒）。此时间总是比访问令牌有效时间长。

请求自定义访问令牌

可以使用密码授予的访问令牌。但是，也可请求自定义访问令牌。使用自定义令牌，可以提供一个使用者名称，以帮助区分令牌用途（用于您的自身目的）。如果密码令牌返回的默认值不满足要求，也可请求特定的有效期。

开始之前

在获取自定义令牌之前，必须先获取密码授予的访问令牌。请参阅请求密码授予的访问令牌，第15页。

此外：

- 仅当您是本地用户时，您才可以请求自定义令牌。外部用户无法请求自定义令牌。
- 您仅可在为其获取令牌的设备上使用此自定义令牌。您不能在高可用性组中的对等设备上使用此令牌。

过程

步骤 1 为自定义访问令牌授权创建 JSON 对象。

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

其中：

- **access_token** 是有效的密码授予的访问令牌。
- **desired_expires_in** 是一个整数，表示自定义访问令牌的有效秒数。相比之下，密码授予的令牌有效期为 1800 秒。
- **desired_refresh_expires_in** 是一个整数，表示自定义刷新令牌的有效秒数。如果获得刷新令牌，请确保此值大于 **desired_expires_in** 值。相比之下，密码授予的刷新令牌有效期为 2400 秒。如果为 **desired_refresh_count** 指定 0，则不需要此参数。
- **desired_subject** 是为自定义令牌提供的名称。

- **desired_refresh_count** 是希望可以刷新令牌的次数。如果不希望获得刷新令牌，请指定 0。如果没有刷新令牌，则当现有访问令牌过期时，必须获取新的访问令牌。

例如，以下命令用于请求 API 客户端的自定义令牌（在 2400 秒后过期）和刷新令牌（在 3000 秒后过期）。该令牌可以刷新 3 次。

```
{{
  "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
}
```

步骤 2 使用 POST /fdm/token 获取访问令牌。

例如，**curl** 命令将如下所示：

```
curl -X POST --header 'Content-Type: application/json' --header
'Accept: application/json' -d '{
   "grant_type": "custom_token",
   "access_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1MDI4MzU5N
jgsInN1YiI6ImFkbWluIiwianRpIjoiYmMyNjM4N2EtODIwOC0xMWU3LWE4MWM
tYzNlYTZkZjJjZThjIiwibmJmIjoxNTAyODM1OTY4LCJleHAiOjE1MDI4Mzc3N
jgsInJlZnJlc2hUb2tlbkV4cGlyZXNBdCI6MTUwMjgzODM2ODYwNiwidG9rZW5
UeXBlIjoiSldUX0FjY2VzcyIsIm9yaWdpbiI6InBhc3N3b3JkIn0.acOE_Y4SE
ds-NE4Qw99fQlUzdoSkhsjInaCh0a9WK38",
   "desired_expires_in": 2400,
   "desired_refresh_expires_in": 3000,
   "desired_subject": "api-client",
   "desired_refresh_count": 3
 }' 'https://ftd.example.com/api/fdm/最新/fdm/token'
```

步骤 3 从响应中检索访问令牌和刷新令牌。

良好响应（状态代码 200）如下所示：

```
{
  "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
}
```

其中：

- **access_token** 是需要包含在 API 调用中的不记名令牌。请参阅在 API 调用中使用访问令牌，第 19 页。
- **expires_in** 是访问令牌自颁发时起的有效时间（单位：秒）。
- **refresh_token** 是用于刷新请求的令牌。请参阅刷新访问令牌，第 19 页。
- **refresh_expires**_in 是刷新令牌的有效时间（单位：秒）。此时间总是比访问令牌有效时间长。

在 API 调用中使用访问令牌

获得密码授予的或自定义访问令牌后，必须在各 API 调用上将其纳入 HTTPS 请求的**授权：不记名**报头中。

例如，执行 GET /object/networks 的 **curl** 命令可能如下所示：

```
curl -k -X GET -H 'Accept: application/json'
-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.yJp
YXQiOjE1MDI4MzU5OTEsInN1YiI6ImFwaS1jbGllbnQiLCJqdG
kiOiJjOWIxYzdjYi04MjA4LTExZTctYTgxYy02YmY0NzY3ZmRm
ZGUiLCJuYmYiOjE1MDI4MzU5OTEsImV4cCI6MTUwMjgzODM5MS
wicmVmcmVzaFRva2VuRXhwaXJlc0F0IjoxNTAyODM4OTkxMzMx
LCJ0b2tlblR5cGUiOiJKV1RfQWNjZXNzIiwib3JpZ2luIjoiY3
VzdG9tIn0.9IVzLjGffVQffHAWdrNkrYfvuO6TgpJ7Zi_z3RYu
bN8'
'https://ftd.example.com/api/fdm/最新/object/networks'
```

注释 使用 API Explorer 尝试方法和资源时，所示的 **curl** 命令不包括**授权：不记名**报头。但是，从 API 客户端进行调用时，必须添加此报头。

刷新访问令牌

访问令牌到期后，需要使用原始授权中提供的刷新令牌来进行刷新。刷新后的访问令牌实际上与原始访问令牌有所不同。“刷新”实际上提供了一对新的访问令牌和刷新令牌，不仅仅是延长了旧访问令牌的使用时间。

过程

步骤 1 为刷新令牌授权创建 JSON 对象。

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

refresh_token 可能来自密码授予的访问令牌或自定义访问令牌授权。

例如：

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

步骤 2 使用 POST /fdm/token 获取刷新后的访问令牌。

例如，**curl** 命令将如下所示：

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

步骤 3 从响应中检索访问令牌和刷新令牌。

良好响应（状态代码 200）如下所示。在此示例中，刷新令牌用于自定义令牌。过期期限基于原始自定义访问令牌请求的值。

```
{
  "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
}
```


其中：

- **access_token** 是需要包含在 API 调用中的不记名令牌。请参阅在 API 调用中使用访问令牌，第 19 页。
- **expires_in** 是访问令牌自颁发时起的有效时间（单位：秒）。
- **refresh_token** 是用于刷新请求的令牌。
- **refresh_expires**_in 是刷新令牌的有效时间（单位：秒）。此时间总是比访问令牌有效时间长。

撤销访问令牌

由于访问令牌在特定时间段内有效，因此在用户注销 API 客户端时，应通过撤销令牌来进行清理。这样可以确保没有后门通向 威胁防御 设备。

过程

步骤 1 为撤销令牌授权创建 JSON 对象。

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

其中：

- **access_token** 必须为密码授予的访问令牌。不能使用自定义访问令牌撤销令牌。
- 必须指定以下内容中的一个，且只可指定一个：
  - **token_to_revoke** 是要撤销的密码授予的令牌或自定义令牌。这可以与 **access_token** 的令牌相同，因此可以使用密码授予的令牌来自行撤销。

- (不使用。)**custom_token_id_to_revoke** 通过内部唯一 ID 标识自定义访问令牌。但是，无法直接获取该值。请使用其他选项。
- **custom_token_subject_to_revoke** 是要撤销的自定义访问令牌的 **desired_subject** 值。

例如：

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

步骤 2 使用 POST /fdm/token 撤销访问令牌。

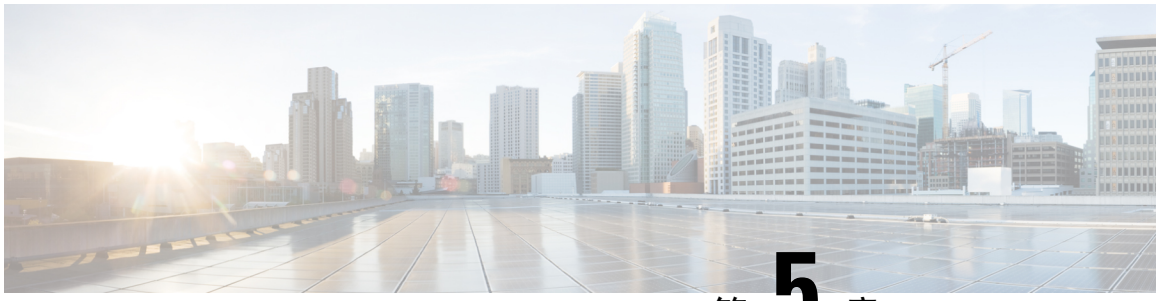
例如，**curl** 命令将如下所示：

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

步骤 3 评估响应以验证令牌是否已撤销。

良好响应（状态代码 200）如下所示。

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



第 5 章

配置 API 的外部用户

版本要求：要使用外部 AAA，您必须运行 威胁防御版本 6.3(0) 或更高版本，以及 威胁防御 REST API v2 或更高版本。

您可以将设备配置为使用外部 RADIUS AAA 服务器在用户访问 威胁防御 REST API 时进行身份验证和授权。您可以使用 RADIUS 用户账户取代内置本地 **admin** 用户账户，或作为后者的补充。

使用外部 AAA 时，您可以定义账户具有不同的授权级别。通过此功能限制可以更改设备配置的用户，同时仍然为支持人员提供只读权限。

以下过程介绍　设置 RADIUS 账户和配置设备使用外部 AAA 进行身份验证和授权的端到端流程。

开始之前

使用外部授权时，请记住以下操作因素。

- 如果设备已配置为高可用性，请在主用设备上配置外部授权。然后，您必须运行授权设置的部署作业，以允许用户访问备用设备。
- 每次新用户访问系统，系统都会为该用户创建用户资源。您需要部署配置以保存该用户对象。

  （威胁防御 6.6 之前的版本。）如果在高可用性 (HA) 模式下运行，必须先部署配置，用户才能登录备用设备。由于只有管理员或读写用户可以启动部署作业，因此首次操作的只读用户必须让其他人部署配置，才能保存“用户”对象。

  从 威胁防御 6.6 开始，将删除对 HA 的限制。外部用户无需先登录主用设备并部署配置即可登录备用设备。系统不会在备用设备上创建用户对象，但会缓存用户特征并为用户提供访问权限（假设用户提供了有效的用户名/密码）。

过程

步骤 1 使用 RADIUS 用户账户定义授权权限，第 24 页。

步骤 2 定义 RADIUS 服务器，第 24 页。

步骤 3 创建 RADIUS 服务器的 AAA 服务器组，第 26 页。

步骤 4 创建 AAA 服务器作为 HTTPS 访问的身份验证源，第 28 页。

步骤 5 使用 **POST /operational/deploy** 启动部署作业。

curl 命令类似于下文：

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

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

有关部署更改的详细信息，请参阅部署配置更改，第 45 页。

步骤 6 验证外部用户访问，第 31 页。

使用 RADIUS 用户账户定义授权权限

您可以从外部 RADIUS 服务器提供对 威胁防御 REST API 的访问权限。通过启用 RADIUS 身份验证和授权，您可以提供不同级别的访问权限，使并非每个用户都通过本地 **admin** 账户登录。

注释 这些外部用户也会获得 设备管理器 授权。

要提供基于角色的访问控制 (RBAC)，请更新 RADIUS 服务器上的用户账户，以定义 **cisco-av-pair** 属性。必须在用户账户上正确定义此属性，否则系统会拒绝用户访问 REST API。以下是受支持的 cisco-av-pair 属性值：

- **fdm.userrole.authority.admin** 提供完全管理员访问权限。这些用户可以执行本地 **admin** 用户可以执行的所有操作。
- **fdm.userrole.authority.rw** 提供读写访问权限。这些用户可以执行只读用户可以执行的任何操作，还可以编辑和部署配置。唯一的限制是无法执行系统关键型操作，包括安装升级、创建和恢复备份、查看审核日志以及注销其他用户。
- **fdm.userrole.authority.ro** 提供只读访问权限。这些用户可以查看控制面板和配置，但无法进行任何更改。如果用户尝试进行更改，会显示错误消息，指明权限不足。

定义 RADIUS 服务器

在 RADIUS 服务器配置用户账户，以定义相应的授权权限之后，可以配置设备使用服务器对 REST API 访问进行身份验证和授权。

使用 **POST /object/radiusidentitysources** 资源为您想要定义的每个 RADIUS 服务器创建对象。

过程

步骤 1 为 RADIUS 服务器创建 JSON 对象正文。

以下是要与此调用结合使用的 JSON 对象示例。

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

属性包括：

- **name** - 对象名称。不需要与 RADIUS 服务器上定义的内容匹配。
- **description** -（可选。）对象的说明。
- **host** - RADIUS 服务器的 IP 地址或完全限定主机名。
- **timeout** -（可选。）系统将请求发送至下一服务器之前等待服务器响应的时长，1-300 秒之间的数值。如果不包含此属性，默认值为 10 秒。
- **serverAuthenticationPort** -（可选。）执行 RADIUS 身份验证和授权的端口。如果不包含此属性，默认值为 1812。
- **serverSecretKey** -（可选。）用于加密 威胁防御设备和 RADIUS 服务器之间数据的共享密钥。该密钥是一个区分大小写的字母数字字符串，最多 64 个字符，且不含空格。密钥必须以字母数字字符或下划线开头，它可以包含特殊字符：$ & - _ . + @。字符串必须匹配 RADIUS 服务器上配置的字符串。如果不配置密钥，则不加密连接。

步骤 2 发布对象。

例如，**curl** 命令会如下所示：

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

步骤 3 验证响应。

您应获得的响应代码为 200。成功的响应正文应类似如下内容。请注意，响应中的密钥等敏感信息会被屏蔽。

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

创建 RADIUS 服务器的 AAA 服务器组

创建 RADIUS 服务器对象后，使用 **POST /object/radiusidentitysourcegroups** 资源创建 AAA 组，以包含 radiusidentitysource 对象。

您可以向 RADIUS AAA 服务器组添加最多 16 个 RADIUS 服务器。这些服务器必须彼此备份，即它们必须具有相同的用户账户列表。

过程

步骤 1 为 RADIUS 服务器组创建 JSON 对象正文。

以下是要与此调用结合使用的 JSON 对象示例。

```
{
  "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"
}
```

属性包括：

- **name** - 对象名称。不需要与 RADIUS 成员服务器上定义的内容匹配。

- **maxFailedAttempts** -（可选。）只有在所有服务器都发生故障后才会重新激活故障服务器。断路时间是指最后一台服务器发生故障后，在重新激活所有服务器之前所等待的时间，其值为 0-1440 分钟。如果不包含此属性，默认值为 10 分钟。
- **deadTime** -（可选。）尝试组中下一个服务器之前发送到 RADIUS 服务器的失败请求（即未收到响应的请求）数。您可以指定 1 到 5 之间的数字，默认值为 3。超过最大失败尝试次数时，系统会将服务器标记为故障。

  对于给定功能，如果您使用本地数据库配置回退方法，并且组中的所有服务器都无法响应，则会将该组视为无法响应，并将尝试回退方法。该服务器组会在断路时间内保持标记为无响应，以确保该时段内其他 AAA 请求不会尝试联系该服务器组，而是立即使用回退方法。
- **description** -（可选。）对象的说明。
- **radiusIdentitySources** - 这是一组定义每个 radiusidentitysource 对象的项目，其中 radiusidentitysource 对象定义包含在组中的 RADIUS 服务器。将项目放置在方括号 [] 中。以下是每个对象的属性和语法。从单个对象获取 **id**、**version** 和 **name** 属性的值；创建对象时，信息位于响应正文中。您还可以从 **GET/object/radiusidentitysources** 调用获取信息。**类型**必须为 **radiusidentitysource**。

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

步骤 2 发布对象。

例如，**curl** 命令会如下所示：

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

步骤 3 验证响应。

您应获得的响应代码为 200。成功的响应正文应类似如下内容。

```
{
  "version": "7r572novdiyy",
```

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

创建 AAA 服务器作为 HTTPS 访问的身份验证源

使用 **PUT /devicesettings/default/aaasettings/{objId}** 资源确定作为用户授权身份源的 RADIUS AAA 服务器组。

没有 POST 方法：系统身份验证所需的对象已存在。必须首先执行 GET 命令，以确定相关的 ID 和版本值。

过程

步骤 1 使用 **GET /devicesettings/default/aaasettings** 确定 aaasettings 对象的属性。

curl 命令类似于下文：

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

例如，响应正文类似于下文。本例展示本地身份源是为HTTPS访问定义的身份源。它还用于与REST API 不相关的 SSH 访问。

```
{
  "items": [
    {
      "version": "du52clrtmawlt",
      "name": "HTTPS",
      "identitySourceGroup": {
        "version": "cynutari5ffkl",
        "name": "LocalIdentitySource",
        "id": "e3e74c32-3c03-11e8-983b-95c21a1b6da9",
        "type": "localidentitysource"
      },
      "description": null,
      "protocolType": "HTTPS",
```

```
      "useLocal": "NOT_APPLICABLE",
      "id": "00000003-0000-0000-0000-000000000007",
      "type": "aaasetting",
      "links": {
        "self": "https://ftd.example.com/api/fdm/最新/
devicesettings/default/aaasettings/00000003-0000-0000-0000-000000000007"
      }
    },
    {
      "version": "fgkhvu4kwucgv",
      "name": "SSH",
      "identitySourceGroup": {
        "version": "cynutari5ffkl",
        "name": "LocalIdentitySource",
        "id": "e3e74c32-3c03-11e8-983b-95c21a1b6da9",
        "type": "localidentitysource"
      },
      "description": null,
      "protocolType": "SSH",
      "useLocal": "NOT_APPLICABLE",
      "id": "00000003-0000-0000-0000-000000000008",
      "type": "aaasetting",
      "links": {
        "self": "https://ftd.example.com/api/fdm/最新/
devicesettings/default/aaasettings/00000003-0000-0000-0000-000000000008"
      }
    }
  ],
  "paging": {
    "prev": [],
    "next": [],
    "limit": 10,
    "offset": 0,
    "count": 2,
    "pages": 0
  }
}
```


步骤 2 （可选。）使用 **GET /devicesettings/default/aaasettings/{objId}** 获取 HTTPS AAA 设置对象副本，以缩小查找范围。

PUT 调用仅更新 HTTPS 对象。不需要更新 SSH 对象。

在本示例中，HTTPS 对象的 ID 是 00000003-0000-0000-0000-000000000007，因此 **curl** 应类似于以下命令：

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

响应正文类似于下文：


```
{
  "version": "ha4653ootep7z",
  "name": "HTTPS",
  "identitySourceGroup": {
    "version": "cynutari5ffkl",
    "name": "LocalIdentitySource",
    "id": "e3e74c32-3c03-11e8-983b-95c21a1b6da9",
    "type": "localidentitysource"
  },
```

```
  "description": null,
  "protocolType": "HTTPS",
  "useLocal": "NOT_APPLICABLE",
  "id": "00000003-0000-0000-0000-000000000007",
  "type": "aaasetting",
  "links": {
    "self": "https://ftd.example.com/api/fdm/最新/
devicesettings/default/aaasettings/00000003-0000-0000-0000-000000000007"
  }
```

步骤 3 为 AAA 管理访问创建 JSON 对象正文。

以下是要与此调用结合使用的 JSON 对象示例。

```
{
  "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"
}
```

属性包括：

- **version** - HTTPS 对象的版本。在响应正文中找到 GET 调用的此值。
- **name** - 对象名称，**HTTPS**。在响应正文中找到 GET 调用的此值。
- **identitySourceGroup** - 可识别 RADIUS 服务器组。创建服务器（或 **GET/object/radiusidentitysourcegroups** 调用）时，从响应正文获取 **id**、**version** 以及 **name** 值。类型必须是 **radiusidentitysourcegroup**。
- **description** -（可选。）对象的说明。
- **protocolType** - 此源应用的协议，**HTTPS**。
- **useLocal** - 如何使用本地身份源，其中包含本地管理员用户账户。输入以下选项之一：
  - **之前** - 系统首先对照本地源检查用户名和密码。
  - **之后** - 仅当外部源不可用或在外部源中找不到用户账户时，检查本地源。
  - **从不** -（不推荐。）从不使用本地源，因此不能以 **admin** 用户身份登录。

    注意 如果您选择 **从不**，将无法使用 **管理员** 账户登录 设备管理器 或使用 API。如果 RADIUS 服务器不可用，或者未在 RADIUS 服务器中配置账户，您将被锁定在系统外面。

- **id** - HTTPS 对象的 ID 值。在响应正文中找到 GET 调用的此值。

- **类型** - 对象类型，**aaasetting** 。

步骤 4 放置对象。

例如，**curl** 命令会如下所示：注意，URL 中的 {objId} 与 JSON 对象中 aaasettings 对象的 ID 相同。

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

步骤 5 验证响应。

您应获得的响应代码为 200。成功的响应正文应类似如下内容。

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

验证外部用户访问

部署作业完成后，您可以测试外部用户对 设备管理器 和 REST API 的访问权限。

过程

步骤 1 使用具有有效 cisco-av-pair 属性的外部用户名登录到 设备管理器。

登录应该是成功的，且页面的右上方应显示用户名和权限级别。

步骤 2 为外部用户获取 REST API 令牌。

如果用户可以获取令牌，用户就可以使用所分配的权限级别所允许的资源和方法。

a) 为简单的密码授予的令牌创建 JSON 对象正文。

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

b) 使用 **POST /fdm/token** 获取令牌。

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

c) 评估响应以验证是否已授予令牌。

您应获得的响应代码为 200。获取令牌意味着系统能够对用户进行身份验证。

```
{
  "access_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1Mjg4MjM
3MTAsInN1YiI6InJhZGl1c3JlYWR3cml0ZXVzZXIxIiwianRpIjoiMjk5Zj
Q5YjYtNmU2NC0xMWU4LWJkNjUtNmY0ZmVmYjY1MzI5IiwibmJmIjoxNTI4O
DIzNzEwLCJleHAiOjE1Mjg4MjU1MTAsInJlZnJlc2hUb2tlbkV4cGlyZXNB
dCI6MTUyODgyNjExMDg4OSwidG9rZW5UeXBlIjoiSldUX0FjY2VzcyIsInV
zZXJVdWlkIjoiMjliMjBlNjctNmU2NC0xMWU4LWJkNjUtMzU4MmUwZjU5Yj
Q4IiwidXNlclJvbGUiOiJST0xFX1JFQURfV1JVEUiLCJvcmlnaW4iOiJwY
XNzd29yZCJ9.dtKsl9IB4ds3RAktEeaSuQy_Zs2SrzLr976UtblBt28",
  "expires_in": 1800,
  "token_type": "Bearer",
  "refresh_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1Mjg4Mj
M3MTAsInN1YiI6InJhZGl1c3JlYWR3cml0ZXVzZXIxIiwianRpIjoiMjk5Z
jQ5YjYtNmU2NC0xMWU4LWJkNjUtNmY0ZmVmYjY1MzI5IiwibmJmIjoxNTI4
ODIzNzEwLCJleHAiOjE1Mjg4MjYxMTAsImFjY2Vzc1Rva2VuRXhwaXJlc0F
0IjoxNTI4ODI1NTEwODg5LCJyZWZyZXNoQ291bnQiOi0xLCJ0b2tlblR5cG
UiOiJKV1RfUmVmcmVzaCIsInVzZXJVdWlkIjoiMjliMjBlNjctNmU2NC0xM
WU4LWJkNjUtMzU4MmUwZjU5YjQ4IiwidXNlclJvbGUiOiJST0xFX1JFQURf
V1JVEUiLCJvcmlnaW4iOiJwYXNzd29yZCJ9.Lc7MYmieNMMrjx7XoTiW-x
8Z8qFCnzfNM1apgbwLQvo",
  "refresh_expires_in": 2400
}
```

步骤 3 使用 **GET /object/users** 验证是否已为每个用户创建用户对象。

系统将为每个登录到设备管理器或获取访问令牌的新用户自动创建用户对象。您必须运行部署作业才能保存这些用户对象。在高可用性模式下，运行部署作业之后，用户才可以登录到备用设备。

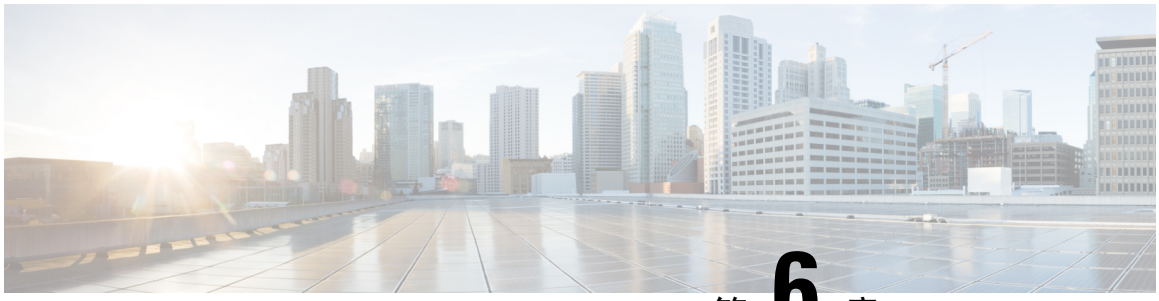
例如，**curl** 命令会如下所示：

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

以下响应正文显示两个外部用户登录。请注意，**userRole** 显示从RADIUS服务器所配置的cisco-av-pair中获取的针对这些用户账户的权限。使用此信息验证是否已正确配置 RADIUS 用户账户。**admin** 用户是本地定义的用户。


```
{
  "items": [
    {
      "version": "h2vom4wckm2js",
      "name": "radiusadminuser1",
      "password": null,
      "newPassword": null,
      "userPreferences": {
        "preferredTimeZone": "(UTC+00:00) UTC",
        "colorTheme": "NORMAL_CISCO_IDENTITY",
        "type": "userpreferences"
      },
      "userRole": "ROLE_ADMIN",
      "identitySourceId": "0a7996ae-6e5b-11e8-bd65-dbab801c44b9",
      "userServiceTypes": [
        "MGMT"
      ],
      "id": "150d9754-6e63-11e8-bd65-ed9b20f62114",
      "type": "user",
      "links": {
        "self": "https://ftd.example.com/api/fdm/最新/
object/users/150d9754-6e63-11e8-bd65-ed9b20f62114"
      }
    },
    {
      "version": "p4rgwcjr5colj",
      "name": "admin",
      "password": null,
      "newPassword": null,
      "userPreferences": {
        "preferredTimeZone": "(UTC-07:00) America/Los_Angeles",
        "colorTheme": "NORMAL_CISCO_IDENTITY",
        "type": "userpreferences"
      },
      "userRole": "ROLE_ADMIN",
      "identitySourceId": "e3e74c32-3c03-11e8-983b-95c21a1b6da9",
      "userServiceTypes": [
        "MGMT"
      ],
      "id": "5023d3ab-6dc5-11e8-b9ed-db6dba9bf94c",
      "type": "user",
      "links": {
        "self": "https://ftd.example.com/api/fdm/最新/
object/users/5023d3ab-6dc5-11e8-b9ed-db6dba9bf94c"
      }
    },
    {
      "version": "ngx7a2dixngoq",
      "name": "radiusreadwriteuser1",
      "password": null,
      "newPassword": null,
      "userPreferences": {
```

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



第 6 章

使用方法和资源

以下主题介绍一般如何使用各种方法和资源。

尝试方法并解释结果

可使用 API Explorer 测试各种方法。本主题介绍一般过程，并解释系统返回的响应。有关特定问题相关技术，请参阅各方法类型主题。

各方法/资源的**试用！**按钮与系统直接交互。GET 检索实际数据，POST/PUT 创建或修改实际资源，DELETE 删除实际对象。您正在系统中进行实际的配置更改，但不会立即部署更改。要使更改处于活动状态，请使用 POST /operational/deploy 资源启动部署作业。

打开方法/资源后，可在**响应消息**部分后找到**试用！**按钮。对于某些方法/资源，必须输入对象 ID 对其进行测试。在这种情况下，您首先通常需要在父资源上执行 GET。有关详细信息，请参阅查找对象 ID (objId) 和父 ID，第 8 页。

对于 POST/PUT，还需要在 JSON 模型中填写所需值。

点击**试用！(Try It Out!)** 后，API Explorer 将结果添加到按钮后的页面上。响应包括以下部分：

Curl

curl 命令用于进行调用。例如，点击 GET /object/networks 资源上的**试用！(Try It Out!)** 返回如下所示的内容：路径中的“v”元素随每个新的 API 版本变化。

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

注释 这不包括**授权：不记名**报头，在客户端的 API 调用中需要该报头。

请求 URL

从客户端发出进行请求的 URL。例如，对于 GET /object/networks：

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

响应正文

系统返回至客户端的对象。如果资源可包含多个对象（如 /object/network），则会得到 GET 请求的项目列表。POST/PUT/DELETE 响应涉及单个对象。

返回的具体内容基于资源模型。例如，GET /object/networks 返回对象列表，各对象类似于以下内容（项目列表的最初指示也已显示）。请注意，links/self 值表示将用于引用此对象的 URL；对象 ID 包含在 URL 中。

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

GET 请求还包括一个分页区，有关说明请见 GET：从系统中获取数据，第 37 页。

响应代码

为 HTTP 调用返回的数字 HTTP 状态代码。这些是标准的 HTTP 状态代码，可以在 RFC 或 Wikipedia 中找到（如 https://en.wikipedia.org/wiki/List_of_HTTP_status_codes）。例如，200 (确定) 表示成功的 GET/PUT/POST 调用，204 表示成功的 DELETE 调用。

响应头

这些是 HTTP 响应中的报头。例如，GET/object/networks 可能具有以下报头：

```
{
  "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：从系统中获取数据

使用 GET 方法从设备中读取信息。

如果一个资源可以包含多个对象，则响应中会提供对象列表。可以在URL中加入查询参数，以控制返回的对象数目。默认设置是从对象列表的开头返回 10 个对象。

以下程序介绍了在 API Explorer 中进行 GET 调用的一般方法。使用 API 客户端示例代码。

过程

步骤 1 在 API Explorer 中，打开 GET 方法（首先，打开组，查看方法和资源）。

步骤 2 如果要使用的方法需要 URL 中的对象或父 ID，可使用父方法获取所需的 ID。

例如，GET/objects/networks/{objId} 需要特定对象的 ID。使用 GET/objects/networks 方法获取网络对象列表，然后查找要检查的对象 ID 值。请注意，在这种情况下，GET/object/networks 调用中返回的信息将与您在 GET/objects/network/{objId} 中看到的信息相同。请参阅查找对象 ID (objId) 和父 ID，第 8 页。

步骤 3 在**参数**部分，配置以下选项：

- **objId** - 如果 URL 中需要，则始终需要对象 ID。例如，900fac69-7d19-11e7-bf7b-d9417b20e59e。
- **parentId** - 父 ID 相当于对象 ID，但仅针对层次结构中较高的父级。例如，GET/policy/intrusion 返回入侵策略列表，而 GET/policy/intrusion/{parentId}/intrusionrules 返回这些策略之一中定义的规则。您将从 GET/policy/intrusion 中获取父 ID。
- **偏移** - 对于支持多个对象的资源，该选项表示从列表的什么位置开始返回对象。默认值 0 表示从列表的开头开始返回对象。
- **限制** - 响应中返回的最大对象数目。默认值为 10。最大限制为 1000；如果您输入了无效的值，则会自动更改为 1000。
- **排序** - 如何对响应中返回的对象进行排序。默认按照**名称**值的字母顺序排列。要更改排序，请在要排序的资源中输入属性名称。例如，可以在网络对象中使用**排序=数值**，对数值（即 IP 地址）属性进行排序。要按相反顺序排序，请加入减号，例如**排序=-名称**。

- **过滤器**（并非适用于所有资源）- 仅返回与过滤条件匹配的项目。过滤值的格式为 {key}{operator}{value}，其中 key 为属性名称，value 为要过滤的字符串。项目之间没有空格。您可以过滤的字段在 API Explorer 中 **filter** 参数的说明中列出；每个对象的字段都不相同。如果对象支持过滤多个字段，可以在 **filter** 参数上添加多个分号分隔的值。例如，您可以对 GET /policy/intrusionpolicies/{parentId}/intrusionrules 过滤 gid:1;sid:105。允许使用以下运算符：
  - **:** 表示等于。例如，**filter=name:Canada**。
  - **!** 表示不等于。例如，**filter=name!Canada**。
  - **~** 表示类似。例如，**filter=name~United**。
- **filter=fts~***string*（并非适用于所有资源）- 仅返回与过滤条件匹配的项目。**fts ~** 选项适用于全文本搜索。会在对象的所有属性中搜索此字符串。您可以包括部分字符串；可以使用星号 * 作为通配符，来匹配一个或多个字符。请勿输入以下字符，因为搜索字符串不支持这些字符：?~!{}<>:%。以下字符将被忽略：;#&。

  例如，您可以使用 GET /object/networks?filter=fts~10 来查找第一个八位组为 10. 的所有网络对象。请注意，系统需要 3-5 秒钟来将新创建或更新的对象添加到索引中，因此您需要先暂停一下，再开始对新增或更改的对象执行全文本搜索。
- **filter=fetchZeroHitCount:{true|false}**（仅可用于访问规则）- 如果指定 **includeHitCounts=true**，则可以使用此过滤器选项来纳入 **(true)** 或排除 **(false)** 未命中的规则，即命中计数为零的规则。默认值为 **true**。
- **includeHitCounts**（仅可用于访问规则）- 是否在策略中包含规则的命中计数信息。指定 **includeHitCounts=true** 可获取命中计数。指定 **false**（默认值）可排除命中计数。命中计数信息返回到返回对象的 hitCount 属性中。
- **time_duration**（仅适用于趋势报告）- 报告应包含过去的多少秒。例如，1800 返回过去 30 分钟的报告。

注释　鉴于 {objId} 和 {parentId} 是 URL 路径的一部分，请在 URL 末尾的 **?** 字符之后添加 **offset**、**limit**、**sort**、**filter**、**includeHitCounts** 和 **time_duration** 参数。

步骤 4 点击**试用！(Try It Out!)** 按钮，检查响应。

对于成功的调用（返回代码200），响应正文包括一个对象或对象列表，具体取决于所进行的调用。有关响应的一般结构和内容的信息，请参阅尝试方法并解释结果，第 35 页。

GET 请求包括一个分页部分。如果对象多于为调用返回的对象，则 **prev** 值和 **next** 值指示如何获取上一组或下一组对象。**count** 值指示对象的总数。**limit** 值指示响应中返回的项数。**offset** 值指示返回对象的起始位置，其中 0 表示列表的开头。

```
"paging": {
    "prev": [],
    "next": [
      "https://ftd.example.com/api/fdm/最新/object/networks?limit=10&offset=10"
    ],
    "limit": 10,
    "offset": 0,
    "count": 22,
```

```
  "pages": 0
}
```

POST：创建新对象

使用 POST 方法为某种资源创建新对象。例如，使用 POST 方法创建新的网络对象。

以下程序介绍了在 API Explorer 中进行 POST 调用的一般方法。使用 API 客户端示例代码。

过程

步骤 1 在 API Explorer 中，打开 POST 方法（首先，打开组，查看方法和资源）。

步骤 2 点击**响应类 (Response Class)** 标题下的**模型 (Model)**，并阅读有关数据类型和资源属性值的信息。

步骤 3 在**参数**标题下，配置以下选项（如果可用）：

- **parentId** - 包含此对象的父对象的 ID。例如，添加 SSL 规则时，SSL 解密策略的 ID。
- **at** - 插入对象的位置，针对驻留在按照顺序列表安排对象的父对象中的对象，例如 SSL 解密策略。使用整数指示位置，0 为列表的开端。默认将新对象添加到列表的末尾。

注释 如果 {objId} 和 {parentId} 是 URL 路径的一部分，请将 **at** 参数添加至 URL 末尾的 **?** 字符后面。

步骤 4 在**参数 (Parameters)** 标题下，点击 **body** 参数的**数据类型 (Data Type)** > **示例值 (Example Value)** 列中显示的 JSON 模型。

点击此框将 JSON 模型加载到 **body** 参数的“数值”(Value) 列中。例如，点击“POST /object/networks resource”框将加载以下正文：

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

步骤 5 填写**正文** JSON 对象属性所需的值。

对于枚举值，一定要在**响应类** > **模型**下读取允许的值。例如，通过填写值并更改**子类型**的默认值，可以为子网（而不是主机）地址创建一个网络对象：

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

```
  "type": "networkobject"
}
```

步骤 6 点击**试用！(Try It Out!)** 按钮，检查响应。

检查用于更新系统的 **curl** 命令。请注意附加报头。创建 API 客户端后，还需要纳入这些报头字段和值。例如，以下是创建示例对象的 **curl** 命令。请注意“内容类型”和“接受”报头。

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

在成功的调用中（返回代码 200），响应正文包含创建的完整对象，包括系统生成的其他值，例如 **version** 和 **id**。版本和 ID 尤为重要，因为后续使用 PUT 更改对象时需要提供这类信息。有关响应的一般结构和内容的信息，请参阅尝试方法并解释结果，第 35 页。

响应正文还包括 **links/self** 值（这是所创建对象的 URL）。例如，以下是示例对象的响应正文。

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

PUT：修改现有对象

使用 PUT 方法更改现有对象的属性。例如，使用 PUT 方法修改现有网络对象中包含的地址，而不更改对象的 ID。

PUT 方法将替换整个对象。不能仅更改一个属性。因此，必须确保 JSON 对象包含要保留的旧值。

以下程序介绍了在 API Explorer 中进行 PUT 调用的一般方法。使用 API 客户端示例代码。

开始之前

使用适用于父资源的 GET 方法，获取对象的现有状态副本，如 GET：从系统中获取数据，第 37 页所述。

您必须至少拥有以下参数的正确值以及不希望更改的所有用户提供的值。

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

过程

步骤 1 在 API Explorer 中，打开 PUT 方法（首先，打开组，查看方法和资源）。

步骤 2 在 **参数**标题下，配置以下选项：

- **objId** - 对象的 **id** 值。例如，900fac69-7d19-11e7-bf7b-d9417b20e59e。
- **parentId** - 针对驻留在其他对象中的对象，包含此对象的父对象的 ID。例如，修改 SSL 规则时，SSL 解密策略的 ID。
- **at** - 插入对象的位置，针对驻留在按照顺序列表安排对象的父对象中的对象，例如 SSL 解密策略。使用整数指示位置，0 为列表的开端。默认将对象添加到列表的末尾。

注释 如果 {objId} 和 {parentId} 是 URL 路径的一部分，请将 **at** 参数添加至 URL 末尾的 **?** 字符后面。

步骤 3 在**参数 (Parameters)** 标题下，点击 **body** 参数的**数据类型 (Data Type)** > **示例值 (Example Value)**列中显示的 JSON 模型。

点击此框将 JSON 模型加载到 **body** 参数的“数值”(Value) 列中。例如，点击“PUT/object/networks resource”框加载以下正文。请注意，这与同一资源的 POST 版本稍有不同：PUT 正文包括 **version** 属性。

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

步骤 4 填写 **body** JSON 对象属性所需的值。

一定要复制不希望更改的旧值。

对于枚举值，一定要在**响应类** > **模型**下读取允许的值。除非要将对象更改为其他子类型，否则请重复旧值。例如，网络对象的默认 PUT 模型具有 **subType** 的 HOST，但如果要更改子网对象，请确保将 **subType** 更改为“网络”。

例如，要更新网络对象中的子网 IP 地址，请对除 **value** 之外的所有属性重复所有旧值。在 **value** 中输入新的子网地址。

```
{
  "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",
}
```

步骤 5 点击**试用！(Try It Out!)** 按钮，检查响应。

检查用于更新系统的 **curl** 命令。请注意附加报头。创建 API 客户端后，还需要纳入这些报头字段和值。例如，以下是用于更新示例对象的 **curl** 命令。请注意“内容类型”和“接受”报头。

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

对于成功的调用（返回代码 200），响应正文包括已更新的完整对象。请注意，版本值会更改，但对象 ID（以及链接/自身）保持不变。每次修改对象时，版本都会更改。有关响应的一般结构和内容的信息，请参阅尝试方法并解释结果，第 35 页。

注释 如果未对对象进行任何更改（即，更新的对象与先前的版本相同），则系统不会处理请求，而是会发送代码 204，表明该资源未发生任何更改。

例如，以下是用于更新示例对象的响应正文。

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

DELETE：删除用户创建的对象

使用 DELETE 方法可删除您或其他用户创建的对象。例如，使用 DELETE 方法删除您不再使用的网络对象。

不能删除系统定义的对象或必须存在的对象。

同时也不能删除当前正被另一个对象使用的对象（例如访问规则中使用的网络对象）。对于使用中的对象，首先修改使用该对象的所有对象，然后删除该对象。

以下程序介绍了在 API Explorer 中进行 DELETE 调用的一般方法。使用 API 客户端示例代码。

开始之前

使用适用于父资源的 GET 方法，获取对象的现有状态副本，如 GET：从系统中获取数据，第 37 页所述。

必须拥有对象 ID（**id** 值）才可删除该对象。

过程

步骤 1 在 API Explorer 中，打开 DELETE 方法（首先，打开组，查看方法和资源）。

步骤 2 在**参数**标题下，在 **objId** 字段中输入对象的 **id** 值。例如，f6d8da49-7ed5-11e7-9bfd-27136f5686ad。

如果该对象位于容器内，您还需要在 **parentId** 字段输入父对象的 ID。

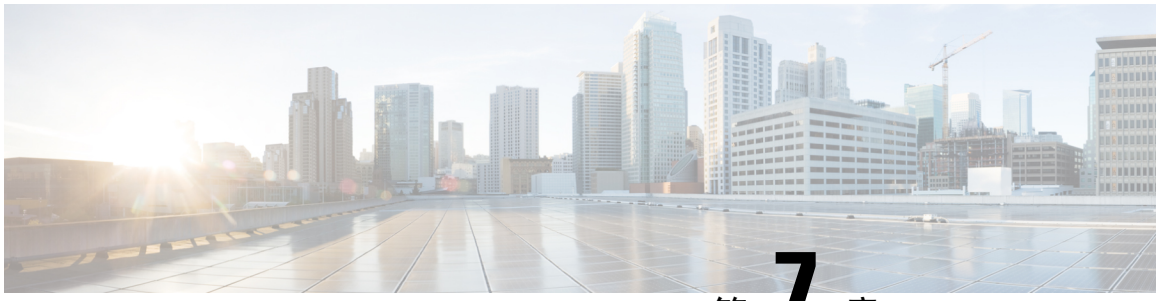
步骤 3 点击**试用！(Try It Out!)** 按钮，检查响应。

检查用于从系统中删除对象的 **curl** 命令。请注意附加报头。创建 API 客户端后，还需要纳入这些报头字段和值。例如，以下是用于删除示例对象的 **curl** 命令。注意“接受”报头。

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

对于成功的调用（返回代码 204“无内容”），将得到一个空的响应正文。这是预期结果。

- **DELETE**：删除用户创建的对象



第 7 章

部署配置更改

部署配置更改

虽然 POST、PUT 和 DELETE 调用将直接更新 威胁防御 设备，但它们并不会立即激活。处理流量时，必须先部署配置更改，然后设备才可以使用新设置。

过程

步骤 1 使用部署组中的 POST/operational/deploy 资源启动部署。

例如，**curl** 命令将如下所示：

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

步骤 2 评估响应以验证部署作业是否已排入队列。

良好响应（状态代码 200）如下所示。注意状态。

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

部署配置更改

注释　　API v2 中添加了 v1 响应中不包含的 **cliErrorMessage** 和 **name** 属性。

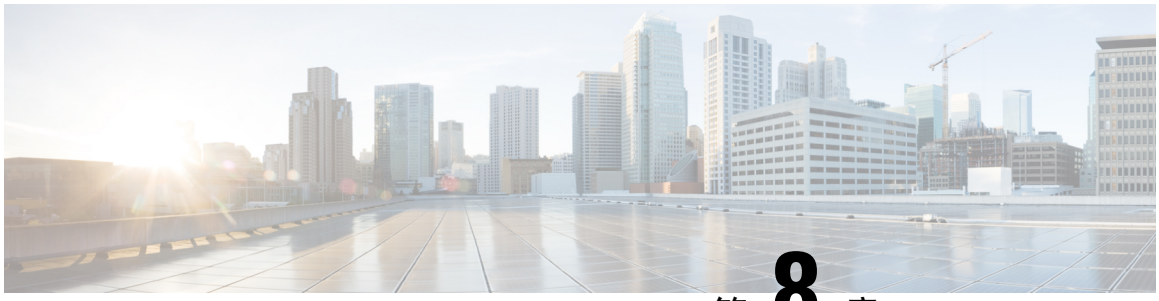
步骤 3 使用 GET/operational/deploy/{objId} 资源检查作业状态。

例如，**curl** 命令将如下所示：

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

响应可能如下所示。注意，状态“已部署”表示已成功完成作业。“modifiedObjects”参数列出了已在部署作业中更改的对象。在这种情况下，名为“new-network”的网络对象发生了一项更改。

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



第 8 章

配置导入/导出

版本要求：要使用配置导入/导出，必须运行 威胁防御版本 6.5(0) 或更高版本，以及 威胁防御 REST API v4 或更高版本。

可以从 设备管理器托管的设备导出配置，并将其导入至同一设备或另一个兼容设备。例如，可以使用配置导入/导出在多个类似设备之间复制基线配置，然后在每台设备上使用 设备管理器 配置各设备的独有特征。

关于配置导入/导出

在本地管理 威胁防御设备时，您可借助 设备管理器FDM 或 CDO 使用 威胁防御 API 导出设备配置。此方法不能与 Cisco Secure Firewall Management Center 管理的设备配合使用。

导出配置后，系统创建 zip 文件。然后，可以将 zip 文件下载至工作站。配置本身表示为在 JSON 格式文本文件中使用属性-值对定义的对象。在将文件导回至同一设备或其他设备前，可以对其进行编辑。

因此，可以使用导出文件创建可部署到网络中其他设备的模板。

导入对象时，还可以选择直接在导入命令中定义对象，而不是在配置文件中定义对象。但是，仅在导入少量更改的情况下，才应直接定义对象。

以下主题介绍有关配置导入/导出的更多信息。

导出文件中包含的内容

导出时，要指定导出文件中所含的配置。完整导出包括导出 zip 文件中的所有内容。根据所选的待导出内容，导出 zip 文件可能包括以下内容：

- 定义各已配置对象的属性-值对。所有可配置项均建模为对象，而不仅仅是 设备管理器 中称为“对象”的那些对象。

- 如果配置了远程接入 VPN，则包括 AnyConnect 软件包和任何其他引用文件，如客户端配置文件 XML 文件、DAP XML 文件和 Hostscan 软件包。
- 如果已配置自定义文件策略，则包括任何引用的干净列表或自定义检测列表。

比较导入/导出和备份/还原

配置导入/导出与备份/恢复不同。

- 备份/恢复用于灾难恢复。仅当设备是同一型号，且其运行的软件版本与进行备份的设备相同时，才能将备份恢复至设备。首先，这是为了将“最后正确的”配置恢复至同一设备，或将配置恢复至替换设备。
- 导入/导出用于保留全部或部分配置。可以使用导出文件在重新映像设备后将配置恢复至设备。或者，也可以使用导出文件作为模板，编辑其内容再将其导入至其他设备。通过导入/导出，可以快速地将新设备配置为特定基线配置，以便更快地将其部署至网络中。在相应限制范围内，甚至可以将文件导入至不同的设备型号，例如从 Firepower 2120 导入到 2130。如果导入文件仅包含所有设备型号上支持的对象，则导入的限制应非常少。其中一个限制是设备需要使用与导出文件相同的 API 版本。

导入/导出策略

以下是可以使用导入/导出的一些方法。

- **创建用于新设备的模板**。将您的型号设备配置为所需基线，然后导出完整配置。随后，可以将该配置导入新设备，然后使用 设备管理器 或 威胁防御 API 进行所需的任何修改。还可以在导入之前编辑模板，以进行上述修改，例如修改各接口的 IP 地址。请注意，完整导出包括 ManagementIP 对象 (type=managementip)；假设您已在目标设备上配置管理地址和网关，则在为新设备创建模板时，应从导出文件中删除此对象，否则将覆盖管理寻址信息。
- **将配置更改从一台设备部署至其他类似设备**。例如，在编辑设备 A 的配置时，会创建一些新的网络对象和访问控制规则。然后，可以导出待处理更改，并将这些更改导入设备 B。在两台设备上部署配置后，它们运行相同的新规则。
- **重新映像系统后重新应用配置**。重新映像设备会擦除配置。如果首先导出完整配置，则可以在完成重新映像后将其导入。
- **应用有针对性的配置**。由于可以编辑甚至手动创建导出文件，因此可以删除除要导入其他设备中的对象以外的所有对象。例如，可以创建包含一组网络对象的配置文件，并用该文件将相同的网络对象组导入至您的所有 威胁防御设备中。

配置导入/导出准则

- 导出作业期间，系统在配置数据库上保持写锁定。作业完成之前，无法使用 API 或 设备管理器 进行配置更改。但是，可以在导出作业期间查看 设备管理器 中的配置或使用 API 中的 GET 调用。
- 导入作业期间，系统在配置数据库上保持读写锁定。作业完成之前，无法使用 API 或 设备管理器 查看配置或对其进行更改。
- 导入配置会添加至现有配置。无法擦除设备配置，并将其替换为导入的配置。如果需要在导入前重置设备配置，则可转至设备 CLI 并发出 **configure manager delete** 命令，然后发出 **configure manager local** 命令。系统将仅保留管理接口配置。
- 仅当设备运行的 API 版本与文件中包含的元数据对象内 apiVersion 属性的定义相同时，才能将文件导入设备。

导入和导出配置

导入/导出过程始于从本地管理的设备导出配置。然后，可以下载导出文件，并根据需要进行编辑，然后再将其上传至同一设备或兼容设备。以下主题介绍各步骤。

导出配置

使用 POST /action/configexport 方法创建和开始配置导出作业。

过程

步骤 1 创建用于导出作业的 JSON 对象正文。

以下是要与此调用结合使用的 JSON 对象示例。

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

属性包括：

- **diskFileName** -（可选。）导出 zip 文件的名称。如果不指定名称，系统会为您生成一个名称。即使指定名称，系统也可能在名称之后附加某些字符以确保唯一性。名称的最大长度为 60 个字符。
- **encryptionKey** -（可选。) zip 文件的加密密钥。如果不想加密文件，请忽略此字段并指定 "doNotEncrypt": true。如果指定密钥，则在将 zip 文件下载至工作站后，需要使用该密钥打开 zip 文件。请注意，导出的配置文件以明文形式公开密钥、密码和其他敏感数据（否则将无法导入），因此您可能希望应用加密密钥来保护敏感数据。系统使用 AES 256 加密。
- **doNotEncrypt** -（可选。）导出文件应该加密 (false) 还是不加密 (true)。默认值为 false，这意味着必须指定非空的 encryptionKey 属性。如果指定 true，则将忽略 encryptionKey 属性。
- **configExportType** - 以下枚举值之一：
  - **FULL_EXPORT** - 在导出文件中包括整个配置。这是默认值。
  - **PARTIAL_EXPORT** - 仅包含在 entityIds 列表中标识的对象及其后代对象。对于不可导出的对象，即使指定其身份，也无法将其包括在内。所有用户定义的对象均可导出。
  - **PENDING_CHANGE_EXPORT** - 仅包括尚未部署的对象，即待处理更改。
- **deployedObjectsOnly** -（可选。）是否仅在对象已部署时才将其包含在导出文件中。也就是说，不包括待处理更改。对于 PENDING_CHANGE_EXPORT 作业，忽略此属性，因为这些作业仅包括未部署对象。默认值为 false，表示导出中包含所有待处理更改。指定 true 以排除待处理更改。
- **entityIds** - 一组起始点对象的身份列表，其中对象以逗号分隔并括在 [方括号] 中。PARTIAL_EXPORT 作业需要此列表。此列表中的各项均可以是 UUID 值或与 "**id=***uuid-value*"、"**type=***object-type*" 或 "**name=***object-name*" 等模式匹配的属性值对。例如，**"type=networkobject"**。

  type 可以是叶实体（例如 networkobject），也可以是一组叶类型的别名。一些典型的类型别名包括：network（NetworkObject 和 NetworkObjectGroup）、port（所有 TCP/UDP/ICMP 端口、协议和组类型）、url（URL 对象和组）、ikepolicy（IKE V1/V2 策略）、ikeproposal（Ike V1/V2 提议）、identitysource（所有身份源），certificate（所有证书类型）、object（将在“对象”(Objects) 页面上的 设备管理器 中列出的所有对象/组类型）、interface（所有网络接口）、s2svpn（所有站点间 VPN 相关类型）、ravpn（所有 RA VPN 相关类型）和 vpn（s2svpn 和 ravpn）。

  所有这些对象及其传出引用后代将包含在 PARTIAL_EXPORT 输出文件中。请注意，所有不可导出对象都将从输出中排除，即使您指定其身份。使用相应资源类型的 GET 方法获取目标对象的 UUID、类型或名称。

  例如，要导出所有网络对象以及名为 myaccessrule 的访问规则和由 UUID 标识的两个对象，可指定：

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

- **jobName** -（可选。）导出作业的名称。在检索作业状态时，给出作业名称可以更轻松地进行查找。
- **type** - 作业类型，始终为 **scheduleconfigexport**。

示例：

以下示例对文件 export-config-1 执行完全导出，并接受所有其他属性的默认值：

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

步骤 2 发布对象。

例如，curl 命令会如下所示：

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

步骤 3 验证响应。

您应获得的响应代码为200。如果发布了最小的JSON对象，则成功的响应正文将类似于以下内容。如果指定加密密钥，则会在响应中屏蔽该密钥。

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

检查导出作业的状态

导出作业需要一些时间才能完成。配置越大，作业所需的时间就越多。检查作业状态，确保其成功完成，然后再尝试下载文件。

获取状态的最简单方法是使用 GET/jobs/configexportstatus。例如，curl 命令会如下所示：

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

已成功完成的作业将返回类似于以下内容的状态。

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

也可以使用 GET /jobs/configexportstatus/{objId} 方法检索特定作业的状态。您会从响应对象的 **id** 字段中获取对象 ID。

下载导出文件

导出作业完成后，系统会将导出文件写入系统磁盘，并将其称为配置文件。可以使用 GET /action/downloadconfigfile/{objId} 方法将此导出文件下载至工作站。要获取可用文件的列表，请使用 GET /action/configfiles 方法。

注释 对于 GET /action/downloadconfigfile/{objId}，通常将文件名指定为对象 ID。或者，也可以指定与该文件相关联的 ConfigExportStatus 对象的 ID。

过程

步骤 1 获取磁盘上的配置文件列表。

配置文件列表包括导出文件和上传用于导入的任何文件。

curl 命令类似于下文：

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

响应将显示项目列表，每个项目都是一个配置文件。例如，以下列表显示 2 个文件。请注意，所有文件的 **id** 均是默认值。忽略 ID，并用 **diskFileName** 代替它。

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

步骤 2 使用 diskFileName 作为对象 ID 下载文件。

curl 命令类似于下文：

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

文件将下载至默认下载文件夹。如果是从 API Explorer 发出 GET 方法，且浏览器配置为提示下载位置，系统会提示您保存文件。

下载成功后会出现 200 返回代码，但无响应正文。

编辑导出的配置文件

下载配置文件后，可以将其解压缩并打开包含这些对象的文本文件。WordPad 格式比 NotePad 格式更便于阅读。还可以使用您可能已安装的其他文本编辑器。甚至可以从头创建自己的配置文件，但需要导出配置以了解文件结构。

以下主题介绍文本文件的要求。

最低配置文件要求

配置文件必须具有以下最小元素：

- 将文件中的对象放在 [方括号] 内。整个文件使用标准 JSON 表示法，是一组对象。
- 将各对象包含在 {大括号} 中。
- 使用逗号分隔配置文件中的对象。也就是说，对象的右括号后面应跟一个逗号，但最后一个对象除外。
- 文件中的第一个对象必须是元数据对象。获取正确对象属性的最简单方法是从所需模型的设备中导出配置。例如，以下是来自 Cisco Secure Firewall Threat Defense Virtual 设备的元数据对象。导入设备之前，可以编辑配置和导出类型，如果需要，请删除 generatedOn 属性。

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

- 元数据对象必须指定适当的配置类型 (configType) 值。
  - FULL_CONFIG - 此文本文件包含完整设备配置。
  - DELTA_CONFIG - 此文本文件包含部分配置，甚至只是几个对象。
- ExportType 是以下类型之一：FULL_EXPORT、PARTIAL_EXPORT、PENDING_CHANGE_EXPORT。
- 如果正在进行完全配置导入，元数据对象必须指定以下属性：hardwareModel、softwareVersion、apiVersion。
- 可以在一行或多行写入对象，但不要在对象中的属性之间添加空行或注释行。文件中不允许使用注释。
- 尽管对象按依赖关系项顺序导出，即首先定义被其他对象引用的对象，但在导入配置文件中无需保留该顺序。系统将在导入期间自动解析关系，假定在相关对象之间会正确解析对象名称和 ID。

身份封装对象基本结构

配置文件使用身份封装对象定义任何可以导出或导入的 ConfigEntity 或 ManagementEntity 对象。以下是身份封装对象的基本结构：

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

```
  "index" : integer,
}
```

该对象包含以下属性：

- **type** - 始终是 **identitywrapper**。
- **data** - 这是从配置中定义对象的属性-值对的集合，例如网络对象、访问控制规则等。此集合所需的属性取决于特定对象类型的型号以及您正在执行的操作。将属性-值对括在 {大括号} 中。使用逗号分隔数据数组中的各个属性。
- **parentName** -（如有需要）。有限数量的对象是 ContainedObjects，与包含这些对象的对象之间有关系。示例包括访问规则、手动 NAT 规则和子接口。对于这些项目，parentName 指定包含其他对象的对象（即父对象）的名称。为被包含于其他对象内的对象指定该属性。对于未包含于其他对象内的对象，请勿指定该属性。您可能还需要指定这些对象的索引。

  如果父对象是单个对象（即，您无法创建多个对象），例如 AccessPolicy，且系统可以解析引用，则实际上可以忽略此属性。
- **oldName** -（如有需要）。如果要重命名现有对象，可以在此属性上指定旧名称，并在数据属性的**名称**属性中指定新名称。必须使用“编辑”操作才能使用此属性。
- **action** - 要对已定义对象执行的操作。在完整导出中，操作始终是**创建**。对于待处理的更改或部分导出，可能会执行**编辑**或**删除**其他操作。

  编辑导入文件时，请指定所需操作。请注意，如果指定“创建”，但对象已存在，则会将操作更改为“编辑”；如果对象不存在，则会将“编辑”更改为“创建”。“删除”操作不会更改。对象引用根据对象类型和名称、对象类型和旧名称或对象类型和父名称予以解析。

  - 创建 - 这是个新对象。您需要指定发布对象时所需的数据属性。请注意，如果**名称**与指定类型的现有对象匹配，则该操作会自动更改为“编辑”。

    请注意，如果创建新对象并从其他对象引用该对象（例如定义网络对象，然后在访问规则中使用该对象），则该引用中的对象**名称**必须正确。
  - 编辑 - 更新对象。您需要指定放置对象时所需的数据属性，但版本和 ID 除外。名称和对象类型用于确定要更新的对象，且版本属性始终被忽略。
  - 删除 - 删除对象。您必须在对象数据中指定**类型**和**名称**属性。
- **index** -（可选；整数。）对于属于有序列表一部分的对象，例如访问控制和手动 NAT 规则，该属性是指对象在策略中的位置。如果要创建新规则而不指定索引值，则该规则将添加至策略末尾作为最新规则。如果您正在编辑规则，则系统将保留该规则的现有位置。

示例：编辑网络对象以导入至其他设备

各对象的结构如下所示，这是一个定义系统日志服务器 IP 地址的网络主机对象：

```
{"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"}}
```

假设您从设备导出此对象，且想要将该对象导入其他设备，但新设备应使用位于不同地址192.168.5.15上的系统日志服务器。由于您要创建新对象，请从数据属性中删除**版本**和 **ID** 属性。还可以删除 **isSystemDefined**（默认值为 false）和 **dnsResolution**（仅适用于 FQDN 对象）。生成的新对象如下所示：

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

在该文件的顶部，需要保留（或添加）元数据对象。您还可以添加换行符，以便更容易地扫描和验证文件内容。因此，完整的配置文件可能如下所示：

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

上传导入文件

必须先将文件上传至设备，然后才能将配置文件导入设备。可以上传 zip 或文本文件。如果使用 zip 文件，则可以包括 AnyConnect 软件包和客户端配置文件。

使用 POST/action/uploadconfigfile 资源上传文件。名称的最大长度为 60 个字符。

- 如果从 API Explorer 使用此方法，请点击 **fileToUpload** 属性旁的**选择文件 (Choose File)** 按钮，以从工作站驱动器选择文件。
- 如果您从自己的程序使用该方法，则请求负载必须包含带有文件名称字段的单个文件项。文件扩展名必须是 .txt 或 .zip，且实际的文件内容格式必须与文件扩展名一致。

curl 命令会如下所示：

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

如果传输成功，则会出现返回代码 200 和类似于以下内容的响应正文，其中显示导入作业所需的 威胁防御系统 (**diskFileName**) 上的文件名。

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

导入配置并检查作业状态

将配置文件上传至 威胁防御系统后，可以将配置文件中定义的对象导入到 威胁防御 配置中。使用 POST /action/configimport 方法。

导入对象时，还可以选择直接在导入命令中定义对象，而不是在配置文件中定义对象。但是，仅在导入少量更改的情况下（例如一两个网络对象），才应直接定义对象。

过程

步骤 1 创建用于导入作业的 JSON 对象正文。

以下是要与此调用结合使用的 JSON 对象示例。

```
{
  "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"
}
```

属性包括：

- **diskFileName** - 要导入的配置 zip 或 txt 文件的名称。
- **encryptionKey** - 用于加密 zip 文件的密钥（如有）。如果配置文件未加密，请勿指定密钥。
- **preserveConfigFile** -（可选。）在成功导入作业后，是否在 威胁防御 磁盘上保留导入的配置文件的副本。指定 true 以保留文件，指定 false 以使文件从 威胁防御 磁盘中删除。默认值为 false。
- **autoDeploy** -（可选。）导入成功时是否自动开始部署作业。导入的对象是待处理更改，在成功部署更改前，这些对象处于非活动状态。指定 true 以自动开始部署作业。如果指定 false，则必须手动部署更改。默认值为 false。
- **allowPendingChange** -（可选。）如有现有待处理更改，是否允许开始导入作业。如果将此属性设置为 true，并将 autoDeploy 设置为 true，则自动部署作业将包括已预先存在和已导入的所有更改。如果将此属性设置为 false，则如果存在待处理更改，导入作业将不会运行。默认值为 false。
- **excludeEntities** -（可选。）标识不应导入的对象的对象匹配字符串列表。仅当导入文件包含您不想导入的项目（即，您决定不从上传文件中删除这些项目）时，才需指定此属性。此列表中的各项目模式如下："**id**=*uuid-value*"、"**type**=*object-type*" 或 "**name**=*object-name*"。系统将从导入中删除与其中一个模式匹配的输入对象。

  type 可以是叶实体（例如 networkobject），也可以是一组叶类型的别名。一些典型的类型别名包括：network（NetworkObject 和 NetworkObjectGroup）、port（所有 TCP/UDP/ICMP 端口、协议和组类型）、url（URL 对象和组）、ikepolicy（IKE V1/V2 策略）、ikeproposal（Ike V1/V2 提议）、identitysource（所有身份源），certificate（所有证书类型）、object（将在“对象”(Objects) 页面上的 设备管理器 中列出的所有对象/组类型）、interface（所有网络接口）、s2svpn（所有站点间 VPN 相关类型）、ravpn（所有 RA VPN 相关类型）和 vpn（s2svpn 和 ravpn）。

  例如，要排除导入所有网络对象以及由名称 myobj 和 UUID 标识的其他两个对象，请指定以下定义：

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

- **inputEntities** - 如果要导入的对象数量很少，则可以在 inputEntities 对象列表中而不是配置文件中对其进行定义。要使用此属性，则不能包含 diskFileName 属性，否则必须将该属性设置为 null。
- **jobName** -（可选。）导出作业的名称。在检索作业状态时，给出作业名称可以更轻松地进行查找。
- **type** - 作业类型，始终为 **scheduleconfigimport**。

示例：

以下示例导入名为 import-1.txt 的配置文件：

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

步骤 2 发布对象。

例如，curl 命令会如下所示：

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

步骤 3 验证响应。

您应获得的响应代码为200。如果发布了最小的JSON对象，则成功的响应正文将类似于以下内容。如果指定加密密钥，则会在响应中屏蔽该密钥。

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

步骤 4 使用 GET /jobs/configimportstatus 检查导入作业的状态。

或者，也可以使用 GET/jobs/configimportstatus/{objId} 获取导入作业的状态。对于 objId，请使用对 POST /action/configimport 调用的响应正文中的 jobHistoryUuid 值。

curl 命令会如下所示：

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

成功导入的响应正文可能如下所示。如果导入失败，可能需要编辑文件以更正格式或内容错误，然后重试。

```
{
      "version": "pcgccfnk4hmiz",
      "jobName": "Config Import",
      "jobDescription": null,
      "user": "admin",
      "startDateTime": "2019-04-25 06:43:54Z",
      "endDateTime": "2019-04-25 06:44:01Z",
      "status": "SUCCESS",
      "statusMessage": "The configuration was imported successfully",
      "scheduleUuid": "7e2b52d8-6725-11e9-abb5-5dec35337506",
      "diskFileName": "import-2.txt",
      "messages": [],
      "preserveConfigFile": true,
      "autoDeploy": true,
      "allowPendingChange": true,
      "id": "7e360139-6725-11e9-abb5-078014531401",
      "type": "configimportjobstatus",
      "links": {
        "self": "https://10.89.5.38/api/fdm/最新
/jobs/configimportstatus/7e360139-6725-11e9-abb5-078014531401"
  }
}
```

下一步做什么

如果将 autoDeploy 设置为 false，则需要运行部署作业以应用导入的更改。使用 POST /operational/deploy 方法。如果将其设置为 true，则应已成功部署设置。在 设备管理器 或 API (GET/operational/auditevents) 中，可以检查审核日志，并将部署作业命名为“后配置导入部署”(Post Configuration Import Deployment)。

注释 某些功能需要特定的许可证。例如，设备必须具有用于任何远程接入 VPN 功能的许可证。但是，导入过程不验证许可证。因此，如果将许可证控制功能的对象导入至无所需许可证的设备中，则部署作业将失败。如果遇到此问题，请将所需许可证分配给设备，或删除这些对象。

删除不需要的导入/导出文件

如果不再需要配置文件（由导出作业创建的配置文件或上传用于配置导入的配置文件），则可删除该配置文件。

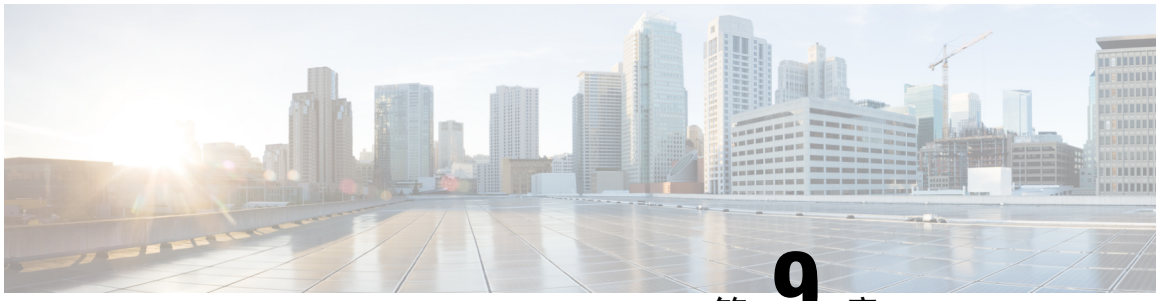
在将文件名作为 objId 值的情况下，请使用 DELETE /action/configfiles/{objId} 方法。

例如，要删除名为 export-config-2.zip 的文件，curl 命令如下所示：

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

成功的结果是出现 204 返回代码，但无响应正文。

可以使用 GET /action/configfiles 确认文件是否已删除。



第 9 章

有关详细信息和示例

有关详细信息和示例

可从以下站点找到有关如何使用 API 的更多信息：

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

  此站点中包括有关这些资源的更多参考信息，包括 Bash 调用示例和 Python 代码。此站点包括一个菜单，用于选择您所用 API 的版本。请选择正确的版本，以查看相关参考信息。此站点还包括一个列表，列出了您在使用 API 时可能会看到的所有唯一错误代码和消息。

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

  此站点包含有关配置选择功能（例如高可用性）的端到端示例，其中包括代码示例。

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

  此站点包括视频、学习模块和实验，以帮助您学习如何使用 API。

有关详细信息和示例

当地语言翻译版本说明

思科可能会在某些地方提供本内容的当地语言翻译版本。请注意，翻译版本仅供参考，如有任何不一致之处，以本内容的英文版本为准。