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。

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 3 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。

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 1 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 1 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 3 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 1 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 3 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

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。