방법 및 리소스 사용

다음 주제에서는 다양한 방법 및 리소스를 사용하는 일반적인 방법에 대해 설명합니다.

방법 시도 및 결과 해석

API Explorer를 사용하여 다양한 방법을 테스트할 수 있습니다. 이 주제에서는 일반적인 프로세스와 시스템에서 반환하는 응답에 대해 설명합니다. 특정 방법 관련 기술에 대해서는 각 방법 유형에 관한 주제를 참조하십시오.

각 방법/리소스의 **Try It Out!(**시도**)** 버튼은 시스템과 직접 상호 작용합니다. GET은 실제 데이터를 검색하고, POST/PUT은 실제 리소스를 생성하거나 수정하며, DELETE는 실제 개체를 제거합니다. 변경 사항이 즉시 구축되지는 않지만 이는 시스템에서 실제 컨피그레이션을 변경하는 것입니다. 변경 사항을 실제로 적용하려면 POST /operational/deploy 리소스를 사용하여 구축 작업을 시작합니다.

Try It Out!(시도**)** 버튼은 방법/리소스를 열면 **Response Message(**응답 메시지**)** 섹션 다음에 있습니다. 일부 방법/리소스는 테스트하려면 개체 ID를 입력해야 합니다. 이 경우 일반적으로 먼저 상위 리소스에서 GET을 수행해야 합니다. 자세한 내용은 개체 ID(objId) 및 상위 ID 찾기의 내용을 참고하십시오.

POST/PUT의 경우 JSON 모델의 필수 값을 입력해야 합니다.

Try It Out!(시도**)**을 클릭하면 API Explorer에서 이 버튼 다음에 나오는 페이지에 결과를 추가합니다. 응답은 다음과 같은 섹션으로 구성되어 있습니다.

Curl

호출하는 데 사용한 **curl** 명령. 예를 들어 GET /object/networks 리소스에서 **Try It Out!(**시도**)**을 클릭하면 다음과 같은 결과가 반환됩니다. 경로의 "v" 요소는 API의 각 새 버전과 함께 변경됩니다.

```
curl -X GET --header 'Accept: application/json'
```

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

참고 여기에는 클라이언트에서 API 호출 시 필요한 **Authorization: Bearer(**권한 부여**:** 전달자**)** 헤더가 포함되어 있지 않습니다.

요청 **URL**

요청을 수행하기 위해 클라이언트에서 발행하는 URL입니다. 예를 들어, GET /object/networks의 경우 다음과 같습니다.

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

응답 본문

시스템이 클라이언트에게 반환하는 개체입니다. 리소스에 /object/network와 같은 개체가 여러 개 포함될 수 있는 경우, GET 요청으로 항목 목록을 가져옵니다. POST/PUT/DELETE 응답은 단일 개체에 대한 것입니다.

반환되는 특정 콘텐츠는 리소스 모델을 기반으로 합니다. 예를 들어, GET /object/networks는 개체 목록을 반환하며 각 개체는 다음과 유사하게 표시됩니다(항목 목록의 초기 지표도 표시됨). links/self 값은 이 개체를 참조하기 위해 사용하는 URL을 나타내며, 개체 ID는 URL에 포함되어 있습니다.

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

GET 요청에는 페이징 섹션도 포함되어 있습니다(GET: 시스템에서 데이터 획득, 3 페이지에 설명되어 있음).

응답 코드

숫자 HTTP 상태 코드는 HTTP 호출에 대해 반환됩니다. 이는 표준 HTTP 상태 코드이며 RFC 또는 Wikipedia에서 찾을 수 있습니다(예: https://en.wikipedia.org/wiki/List_of_HTTP_status_codes). 예를 들어, 200(OK)은 성공적인 GET/PUT/POST 호출을 나타내며 204는 성공적인 DELETE 호출을 나타냅니다.

응답 헤더

이는 HTTP 응답의 패킷 헤더입니다. 예를 들어, GET /object/networks에는 다음과 같은 헤더가 있을 수 있습니다.


```
{
  "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: 시스템에서 데이터 획득

GET 방법을 사용하여 디바이스의 정보를 읽습니다.

리소스가 여러 개체를 포함할 수 있는 경우, 응답에 개체 목록을 가져옵니다. URL에 쿼리 파라미터를 포함하여 반환된 개체의 수를 제어할 수 있습니다. 기본값은 개체 목록의 처음부터 10개의 개체를 반환하는 것입니다.

다음 절차에서는 API Explorer에서 GET 호출 시 일반적인 접근 방식에 대해 설명합니다. API 클라이언트의 예시 코드를 사용합니다.

프로시저

단계 **1** API Explorer에서 GET 방법을 엽니다(먼저 그룹을 열어 방법 및 리소스 확인).

단계 **2** 사용하려는 방법에서 URL에 개체 또는 상위 ID를 필요로 하는 경우, 상위 방법을 사용하여 필요한 ID를 획득합니다.

예를 들어, GET /objects/networks/{objId}에는 지정된 개체의 ID가 필요합니다. GET /objects/networks 방법을 사용하여 네트워크 개체의 목록을 가져온 다음 검사하려는 개체의 ID 값을 찾습니다. 이 경우, GET /object/networks 호출에서 반환되는 정보는 GET /object/networks/{objId}에서 확인할 수 있는 정보와 동일합니다. 개체 ID(objId) 및 상위 ID 찾기의 내용을 참조하십시오.

단계 **3** **Parameters(**파라미터**)** 섹션에서 다음 옵션을 구성합니다.

- **objId** — 개체 ID는 URL에 필요한 경우 항상 필수 항목입니다. 예를 들어, 900fac69-7d19-11e7-bf7b-d9417b20e59e를 입력합니다.

- **parentId** — 상위 ID는 개체 ID에 해당하지만 계층 구조에서는 상위 ID가 더 높습니다. 예를 들어, GET /policy/intrusion은 침입 정책 목록을 반환하는 반면 GET /policy/intrusion/{parentId}/intrusionrules는 이러한 정책 중 하나에 정의되어 있는 규칙을 반환합니다. GET /policy/intrusion에서 상위 ID를 가져옵니다.
- **offset** — 다수의 개체를 지원하는 리소스의 경우, 개체 반환을 시작할 목록 범위를 나타냅니다. 기본값인 0은 목록의 처음을 나타냅니다.
- **limit** — 응답에서 반환할 개체의 최대 수입니다. 기본값은 10입니다. 최대한도는 1000입니다. 잘못된 값을 입력하는 경우, 1000으로 자동 변경됩니다.
- **sort** — 응답에서 반환된 개체를 정렬하는 방식입니다. 기본 정렬은 **name** 값을 기준으로 알파벳 순으로 수행됩니다. 정렬 방식을 변경하려면 정렬할 대상 리소스 내에서 특성 이름을 입력합니다. 예를 들어, 네트워크 개체에서 **sort=value**를 사용하여 값(즉, IP 주소) 특성을 정렬할 수 있습니다. 반대 순서로 정렬하려면 빼기 기호를 포함합니다(예: **sort=-name**).
- **filter**(모든 리소스에 사용할 수는 없음) — 필터 기준에만 일치하는 항목을 반환합니다. 필터 값 형식은 {key}{operator}{value}이며, 이 경우 키는 특성 이름이고 값은 필터링할 문자열입니다. 항목 사이에는 공백이 없습니다. 필터링할 수 있는 필드는 API Explorer의 **filter(**필터**)** 파라미터에 나열되며 개체별로 다릅니다. 개체가 여러 필드에서 필터링을 지원하는 경우에는 **filter(**필터**)** 파라미터에 여러 값을 세미콜론(;)으로 구분하여 포함할 수 있습니다. 예를 들어 GET /policy/intrusionpolicies/{parentId}/intrusionrules의 경우 gid:1;sid:105를 기준으로 필터링할 수 있습니다. 허용되는 연산자는 다음과 같습니다.
  - **:**은 같다는 의미입니다. 예: **filter=name:Canada**
  - **!**는 같지 않다는 의미입니다. 예: **filter=name!Canada**
  - **~**는 유사하다는 의미입니다. 예: **filter=name~United**
- **filter=fts~***string*(모든 리소스에 사용할 수는 없음) - 필터에 일치하는 항목만 반환합니다. **ft~**는 전체 텍스트를 검색할 수 있는 옵션입니다. 개체에 있는 모든 속성은 문자열에 대해 검색됩니다. 부분 문자열을 포함할 수 있습니다. 별표(*)를 와일드카드로 사용하여 하나 이상의 문자에 매칭하는 것은 선택 사항입니다. ?~!{}<>:% 문자는 검색 문자열의 일부로 지원하지 않으므로 포함하지 마십시오. ;#& 문자는 무시됩니다.

  예를 들어 GET /object/networks?filter=fts~10을 사용하여 첫 번째 옥텟이 10인 모든 네트워크 개체를 찾을 수 있습니다. 새로 생성된 또는 업데이트된 개체의 색인을 조회하는 데 3~5초 걸리므로 새로운 또는 변경된 개체에 대한 전체 텍스트 검색을 바로 수행하기 전에 잠시 멈추어야 합니다.
- **filter=fetchZeroHitCount:**{**true**|**false**}(액세스 규칙에만 사용할 수 있음) - **includeHitCounts=true**를 지정하면 이 필터 옵션을 사용해 적중되지 않은, 즉 적중 횟수가 0인 규칙을 포함(**true**) 또는 제외(**false**)할 수 있습니다. 기본값은 **true**입니다.
- **includeHitCounts**(액세스 규칙에만 사용할 수 있음) - 규칙에 대한 적중 횟수 정보를 정책에 포함할지 여부. **includeHitCounts=true**를 지정하여 적중 횟수를 가져옵니다. **false**(기본값)를 지정하여 적중 횟수를 제외합니다. 적중 횟수 정보는 반환되는 개체의 hitCount(적중 횟수) 속성으로 반환됩니다.

- **time_duration**(경향 보고서에만 사용 가능) — 보고서에 포함되어야 하는 지난 시간(초)입니다. 예를 들어, 1800은 지난 30분에 대한 보고서를 반환합니다.

참고 반면, {objId} 및 {parentId}는 URL 경로의 일부이므로 URL 끝의 **?** 문자 다음에 **offset**, **limit**, **sort**, **filter**, **includeHitCounts**, 및 **time_duration** 파라미터를 추가합니다.

단계 **4** **Try It Out!(**시도**)** 버튼을 클릭하고 응답을 확인합니다.

호출에 성공하는 경우(반환 코드 200) 응답 본문에는 수행한 호출에 따라 하나의 개체 또는 개체 목록이 포함됩니다. 응답의 일반적인 구조와 콘텐츠에 대한 내용은 방법 시도 및 결과 해석, 1 페이지를 참조하십시오.

GET 요청에는 paing 섹션이 포함되어 있습니다. 호출에 반환된 개체보다 더 많은 개체가 있는 경우, **prev** 및 **next** 값은 개체의 이전 또는 다음 집합을 가져오는 방법을 나타냅니다. **count** 값은 개체의 전체 개수를 나타냅니다. **limit** 값은 응답에서 반환되는 항목의 수를 나타냅니다. **offset** 값은 반환된 개체의 시작 위치를 나타냅니다(0은 목록의 처음을 나타냄).

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

POST: 새 개체 생성

POST 방법을 사용하여 리소스 유형의 새 개체를 생성합니다. 예를 들어, POST를 사용하여 새 네트워크 개체를 생성합니다.

다음 절차에서는 API Explorer에서 POST 호출 시 일반적인 접근 방식에 대해 설명합니다. API 클라이언트의 예시 코드를 사용합니다.

프로시저

단계 **1** API Explorer에서 POST 방법을 엽니다(먼저 그룹을 열어 방법 및 리소스 확인).

단계 **2** **Response Class(**응답 클래스**)** 머리글 아래에서 **Model(**모델**)**을 클릭하고 리소스 특성의 데이터 유형과 값을 읽습니다.

단계 **3** **Parameters(**파라미터**)** 머리글 아래에서 다음 옵션(사용 가능한 경우)을 구성합니다.

- **parentId** — 이 개체를 포함할 상위 개체의 ID입니다. 예를 들어 SSL 규칙 추가 시에는 SSL 암호 해독 정책의 ID입니다.

- **at(**위치**)** — SSL 암호 해독 정책 등 순서가 지정된 목록에서 개체를 구성하는 상위 항목에 있는 개체의 경우 개체를 삽입할 위치입니다. 정수를 사용하여 위치를 표시합니다. 0이 목록의 시작 위치입니다. 기본적으로는 목록의 끝에 새 개체가 삽입됩니다.

참고 반면, {objId} 및 {parentId}는 URL 경로의 일부이므로 URL 끝의 **?** 문자 다음에 **at** 파라미터를 추가합니다.

단계 **4** 또한 **Parameters(**파라미터**)** 머리글 아래에서 **body** 파라미터에 대한 **Data Type(**데이터 유형**)** > **Example Value(**예시 값**)** 열에 표시된 JSON 모델을 클릭합니다.

이 상자를 클릭하면 **body** 파라미터에 대한 Value(값) 열에 JSON 모델이 로드됩니다. 예를 들어, POST /object/networks 리소스 상자를 클릭하면 다음과 같은 본문이 로드됩니다.

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

단계 **5** **body** JSON 개체 특성에 대한 필수 값을 입력합니다.

열거 값의 경우, **Response Class(**응답 클래스**)** > **Model(**모델**)** 아래에서 허용된 값을 읽어야 합니다. 예를 들어, 다음과 같이 값을 입력하고 **subType**의 기본값을 변경하여 호스트 대신 서브넷 주소의 네트워크 개체를 생성할 수 있습니다.

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

단계 **6** **Try It Out!(**시도**)** 버튼을 클릭하고 응답을 확인합니다.

시스템을 업데이트하는 데 사용된 **curl** 명령을 확인합니다. 추가 헤더에 주목합니다. API 클라이언트를 생성할 때, 이 헤더 필드 및 값도 포함해야 합니다. 예를 들어 샘플 개체를 생성하기 위한 **curl** 명령은 다음과 같습니다. Content-Type 및 Accept 헤더에 주목합니다.

```
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/최신/object/networks'
```

호출에 성공하는 경우(반환 코드 200) 응답 본문에는 **version** 및 **id**(와)과 같은 시스템 생성 값을 비롯하여 생성한 개체가 완전히 포함됩니다. 버전 및 ID 값은 나중에 PUT을 사용하여 개체를 변경하는 경

우 필요하므로 특히 중요합니다. 응답의 일반적인 구조와 콘텐츠에 대한 내용은 방법 시도 및 결과 해석, 1 페이지를 참조하십시오.

응답 본문에는 **links/self** 값도 포함되는데, 이 값은 생성한 개체의 URL입니다. 예를 들어, 샘플 개체에 대한 응답 본문은 다음과 같습니다.

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

PUT: 기존 개체 수정

PUT 방법을 사용하여 기존 개체의 특성을 변경합니다. 예를 들어, PUT을 사용하여 개체 ID를 변경하지 않고 기존 네트워크 개체 내에 포함된 주소를 수정합니다.

PUT 방법은 전체 개체를 대체합니다. 한 가지 특성만 변경할 수는 없습니다. 따라서 JSON 개체에 보존하려는 기존 값이 포함되어 있는지 확인해야 합니다.

다음 절차에서는 API Explorer에서 PUT 호출 시 일반적인 접근 방식에 대해 설명합니다. API 클라이언트의 예시 코드를 사용합니다.

시작하기 전에

GET: 시스템에서 데이터 획득, 3 페이지에 설명된 대로 상위 리소스에 대한 GET 방법을 사용하여 기존 상태의 개체의 복사본을 획득합니다.

최소한 다음 파라미터에 대해 올바른 값을 사용하고, 사용자가 제공한 모든 값을 변경하지 않고 사용해야 합니다.

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

프로시저

단계 **1** API Explorer에서 PUT 방법을 엽니다(먼저 그룹을 열어 방법 및 리소스 확인).

단계 **2** **Parameters(**파라미터**)** 머리글 아래에서 다음 옵션을 구성합니다.

- **objId** - 개체의 **id** 값입니다. 예를 들어, 900fac69-7d19-11e7-bf7b-d9417b20e59e를 입력합니다.
- **parentId** - 다른 개체 내에 있는 개체의 경우 이 개체를 포함할 상위 개체의 ID입니다. 예를 들어 SSL 규칙 수정 시에는 SSL 암호 해독 정책의 ID입니다.
- **at(**위치**)** — SSL 암호 해독 정책 등 순서가 지정된 목록에서 개체를 구성하는 상위 항목에 있는 개체의 경우 개체를 삽입할 위치입니다. 정수를 사용하여 위치를 표시합니다. 0이 목록의 시작 위치입니다. 기본적으로는 목록의 끝에 개체가 삽입됩니다.

참고 반면, {objId} 및 {parentId}는 URL 경로의 일부이므로 URL 끝의 **?** 문자 다음에 **at** 파라미터를 추가합니다.

단계 **3** 또한 **Parameters(**파라미터**)** 머리글 아래에서 **body** 파라미터에 대한 **Data Type(**데이터 유형**)** > **Example Value(**예시 값**)** 열에 표시된 JSON 모델을 클릭합니다.

이 상자를 클릭하면 **body** 파라미터에 대한 Value(값) 열에 JSON 모델이 로드됩니다. 예를 들어, PUT /object/networks 리소스 상자를 클릭하면 다음과 같은 본문이 로드됩니다. 동일한 리소스에 대한 POST 버전과 약간 다르게 PUT 본문에는 **version** 속성이 포함되어 있다는 점에 유의하십시오.

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

단계 **4** **body** JSON 개체 속성에 대한 필수 값을 입력합니다.

변경하지 않을 기존 값은 복제해야 합니다.

열거 값의 경우, **Response Class(**응답 클래스**)** > **Model(**모델**)** 아래에서 허용된 값을 읽어야 합니다. 개체를 다른 하위 유형으로 변경하지 않는 한 기존 값을 반복합니다. 예를 들어 네트워크 개체에 대한 기본 PUT 모델에는 **subType**에 대한 HOST가 있지만, 서브넷 개체를 변경하는 경우에는 **subType**(을)를 NETWORK로 변경해야 합니다.

예를 들어 네트워크 개체에서 서브넷 IP 주소를 업데이트하려면 **value**(을)를 제외한 모든 속성에 대해 모든 기존 값을 반복합니다. **value**에 새 서브넷 주소를 입력합니다.

```
{
  "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",
}
```

단계 **5** **Try It Out!(**시도**)** 버튼을 클릭하고 응답을 확인합니다.

시스템을 업데이트하는 데 사용된 **curl** 명령을 확인합니다. 추가 헤더에 주목합니다. API 클라이언트를 생성할 때, 이 헤더 필드 및 값도 포함해야 합니다. 예를 들어 샘플 개체를 업데이트하기 위한 **curl** 명령은 다음과 같습니다. Content-Type 및 Accept 헤더에 주목합니다.

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

호출에 성공하는 경우(반환 코드 200) 응답 본문에는 업데이트한 개체가 완전히 포함됩니다. 버전 값은 변경되지만 개체 ID(따라서 link/self도)는 동일하게 유지됩니다. 개체를 수정할 때마다 버전이 변경됩니다. 응답의 일반적인 구조와 콘텐츠에 대한 내용은 방법 시도 및 결과 해석, 1 페이지를 참조하십시오.

참고 개체를 변경하지 않은 경우, 즉 업데이트 중인 개체가 이전 버전과 동일한 경우 시스템에서는 요청을 처리하지 않고 대신 해당 리소스에 대해 변경된 사항이 없음을 나타내는 204 코드를 다시 전송합니다.

예를 들어, 샘플 개체 업데이트에 대한 응답 본문은 다음과 같습니다.

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

DELETE: 사용자가 생성한 개체 제거

DELETE 방법을 사용하여 자신 또는 다른 사용자가 생성한 개체를 제거합니다 예를 들어, DELETE를 사용하여 더 이상 사용하지 않는 네트워크 개체를 제거합니다.

시스템 정의 개체 또는 있어야 하는 개체는 삭제할 수 없습니다.

또한, 다른 개체에서 현재 사용 중인 개체(예: 액세스 규칙에 사용되는 네트워크 개체)도 삭제할 수 없습니다. 사용 중인 개체의 경우 먼저 해당 개체를 사용하는 모든 개체를 수정한 다음 해당 개체를 삭제합니다.

다음 절차에서는 API Explorer에서 DELETE 호출 시 일반적인 접근 방식에 대해 설명합니다. API 클라이언트의 예시 코드를 사용합니다.

시작하기 전에

GET: 시스템에서 데이터 획득, 3 페이지에 설명된 대로 상위 리소스에 대한 GET 방법을 사용하여 기존 상태의 개체의 복사본을 획득합니다.

개체를 삭제하려면 개체 ID(**id** 값)가 있어야 합니다.

프로시저

단계 **1** API Explorer에서 DELETE 방법을 엽니다(먼저 그룹을 열어 방법 및 리소스 확인).

단계 **2** **Parameters(**파라미터**)** 머리글 아래에서 **objId** 필드의 개체에 대한 **id** 값을 입력합니다. 예를 들어, f6d8da49-7ed5-11e7-9bfd-27136f5686ad를 입력합니다.

개체가 컨테이너 내에 있는 경우에는 **parentId** 필드에 상위 개체 ID도 입력해야 합니다.

단계 **3** **Try It Out!(**시도**)** 버튼을 클릭하고 응답을 확인합니다.

시스템에서 개체를 삭제하는 데 사용된 **curl** 명령을 확인합니다. 추가 헤더에 주목합니다. API 클라이언트를 생성할 때, 이 헤더 필드 및 값도 포함해야 합니다. 예를 들어 샘플 개체를 삭제하기 위한 **curl** 명령은 다음과 같습니다. Accept 헤더에 주목합니다.

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

호출에 성공하면(반환 코드 204 "No Content(콘텐츠 없음)") 빈 응답 본문을 받게 되며, 이는 정상적인 결과입니다.