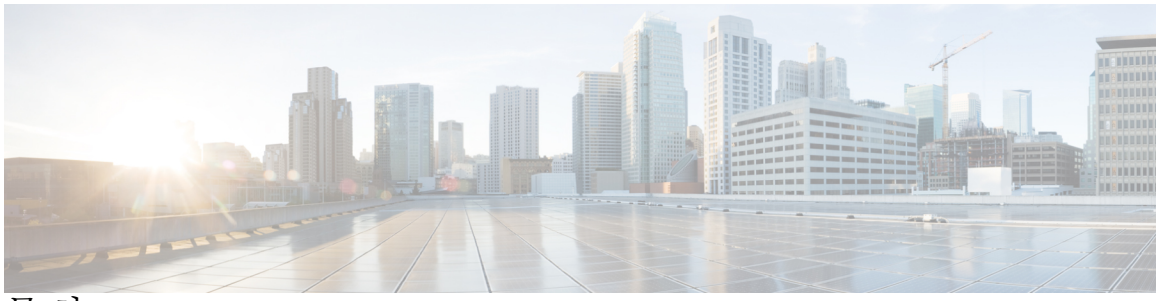
Cisco Secure Firewall Threat Defense REST API 가이드

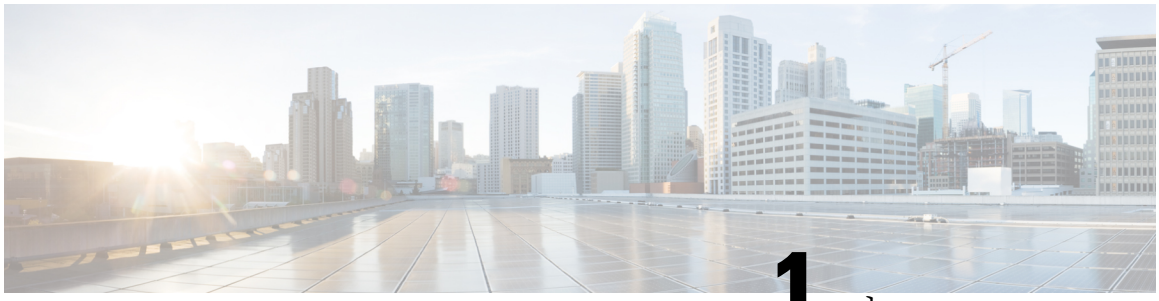
초판: 2018년 3월 29일

최종 변경: 2023년 7월 27일

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



목 차



1 장

Secure Firewall Threat Defense REST API 정보

HTTPS를 통해 Secure Firewall Threat Defense REST(Representational State Transfer) API(Application Programming Interface)를 사용하여 클라이언트 프로그램을 통해 위협 방어 디바이스와 상호 작용할 수 있습니다. REST API는 JSON(JavaScript Object Notation) 형식을 사용하여 개체를 표시합니다.

Secure Firewall device manager에는 프로그래밍 용도로 사용할 수 있는 모든 리소스 및 JSON 개체를 설명하는 API Explorer가 포함됩니다. API Explorer는 각 개체의 '특성-값' 쌍에 대한 자세한 정보를 제공합니다. 따라서 다양한 HTTP 방법을 실험하여 각 리소스를 사용하는 데 필요한 코딩을 파악할 수 있습니다. API Explorer는 각 리소스에 필요한 URL 예시도 제공합니다.

https://developer.cisco.com/site/ftd-api-reference/에서 참조 정보 및 예시 온라인을 확인할 수도 있습니다.

API에는 고유한 버전 번호가 있습니다. API의 한 버전을 위해 설계된 클라이언트가 오류 없이 또는 프로그램을 변경하지 않고도 향후 버전에서 작동한다는 보장은 없습니다.

이 프로그래밍 가이드의 대상 독자

이 가이드는 대상 독자가 프로그래밍에 대한 일반적인 지식을 갖추고 있으며 REST API 및 JSON에 대해 명확하게 이해하고 있다는 가정 하에 작성되었습니다. 이러한 기술을 처음 접하는 경우, REST API에 대한 일반적인 가이드를 먼저 읽어보십시오.

지원되는 HTTP 방법

다음과 같은 HTTP 방법만 사용할 수 있으며, 다른 방법은 지원되지 않습니다.

- GET — 시스템에서 데이터를 읽을 때 사용합니다.

- POST — 새로운 개체를 생성할 때 사용합니다.
- PUT — 기존 개체를 수정할 때 사용합니다. PUT을 사용할 때는 전체 JSON 개체를 포함해야 합니다. 개체 내에서 개별 특성을 선택적으로 업데이트할 수는 없습니다.
- DELETE — 사용자 정의 개체를 제거할 때 사용합니다.

API의 기본 URL

특정 위협 방어 디바이스의 기본 URL을 확인하는 가장 쉬운 방법은 API Explorer에서 GET 방법을 시도하는 것으로, 그 결과에서 URL의 개체 부분을 삭제하기만 하면 됩니다.

예를 들어, GET /object/networks를 수행하고 요청 URL 아래에 반환된 출력에서 다음과 유사한 출력을 확인할 수 있습니다.

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

URL의 서버 이름 부분은 위협 방어 디바이스의 호스트네임 또는 IP 주소이며, 사용자 디바이스의 경우 "ftd.example.com"이 다른 내용으로 대체됩니다. 이 예에서 기본 URL을 얻으려면 경로에서 /object/networks를 삭제합니다.

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

모든 리소스 호출은 이 URL을 요청 URL에 대한 기본 URL로 사용합니다.

HTTPS 데이터 포트를 변경한 경우 URL에 맞춤형 포트를 포함해야 합니다. 예를 들어 포트를 4443으로 변경한 경우 https://ftd.example.com:4443/api/fdm/v1/과 같습니다.

URL의 "v" 요소는 API 버전이며 일반적으로 소프트웨어 버전과 함께 변경됩니다. 예를 들어 위협 방어 버전 6.3.0의 API 버전은 v2이므로 기본 URL은 다음과 같습니다.

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

참고 위협 방어 6.4부터는 경로에서 v 요소 대신 **latest(**최신**)** 버전을 사용하여 API 호출에서 경로를 업데이트하지 않아도 됩니다. 예를 들어, https://ftd.example.com/api/fdm/latest/와 같습니다. **latest(**최신**)**라는 별칭은 디바이스에서 지원하는 최신 API 버전으로 해석됩니다.

API Explorer에서 페이지 하단으로 스크롤하면 기본 URL(서버 이름 제외)과 API 버전에 대한 정보를 확인할 수 있습니다.

REST API에 대한 SSL/TLS 통신 보안 유지

Threat Defense 디바이스는 디바이스와의 HTTPS 통신을 시작할 수 있도록 자체 서명 인증서와 함께 제공됩니다. 그러나 인증서가 알려진 CA(Certificate Authority)를 통해 서명되지 않기 때문에 모든 SSL/TLS 액세스 시도에서 연결을 안전하지 않은 것으로 간주합니다.

브라우저에 연결할 때 자체 서명 인증서를 수락하라는 프롬프트가 표시되지만, curl과 같은 명령에서는 인증서를 거부합니다. curl의 경우 **--insecure** 키워드를 추가하여 인증서 확인 실패를 우회할 수 있습니다. 예를 들면 다음과 같습니다.

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

가장 먼저 수행해야 하는 작업 중 하나는 위협 방어 디바이스에 대한 CA 서명 디바이스 인증서를 획득하는 것입니다. 그런 다음 device manager 또는 API를 사용하여 이 인증서를 관리 인증서로 할당합니다. 이후에는 SSL/TLS 인증서 검사가 실패하지 않으며 API 호출에서 보안되지 않은 통신을 사용할 필요가 없습니다.

프로시저

단계 **1** **POST /object/internalcertificates** 리소스를 사용하여 CA 서명 디바이스 인증서를 업로드합니다.

단계 **2** **PUT /devicesettings/default/webuicertificates/{objId}** 리소스를 사용하여 이 인증서를 관리 인증서로 설정합니다.

GET /devicesettings/default/webuicertificates 리소스를 사용하여 웹 UI 인증서의 개체 ID를 확인합니다.

단계 **3** **POST /operational/deploy** 리소스를 사용하여 변경 사항을 구축합니다.

지원되는 API 버전 확인

GET /api/versions (ApiVersions) 메서드를 사용해 디바이스에서 지원되는 API 버전을 확인할 수 있습니다. 이 메서드에서는 인증이 필요 없고 경로에 버전 요소를 포함하지도 않습니다. 예를 들면 다음과 같습니다.

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

threat defense 디바이스의 호스트네임 또는 IP 주소를 "ftd.example.com"으로 대체합니다.

이 메서드에서는 사용할 수 있는 API 버전의 목록을 반환합니다. 예를 들면 다음과 같습니다.

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

버전 문자열은 후속 API 호출에 대한 URL에서 사용하는 것과 동일합니다. 특정 버전의 식별자 대신 **latest(**최신**)** 버전을 사용하는 경우, 후속 릴리스를 위해 호출을 업데이트하지 않아도 됩니다. 그러나 이 기술을 사용해도 호출에 사용된 개체 모델에 대한 변경 사항은 해결되지 않으므로 릴리스에서 릴리스로 조정해야 할 수 있습니다.

일반적으로 다음 단계는 OAuth를 사용한 REST API 클라이언트 인증, 13 페이지에 설명된 대로 액세스 토큰을 가져오는 것입니다.

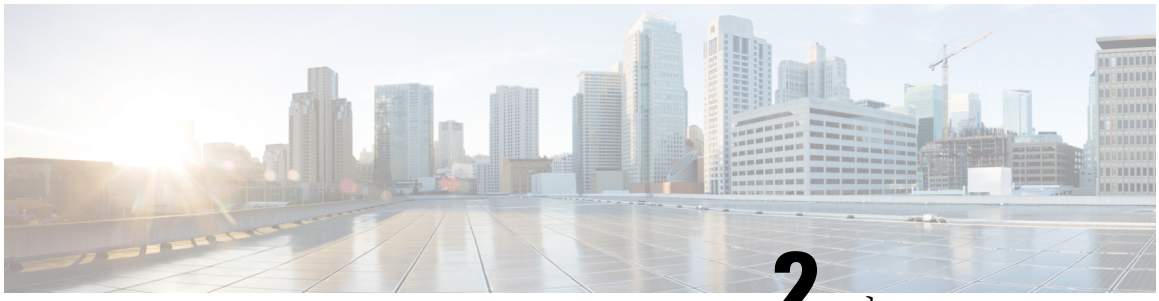
API 버전 이전 버전과의 호환성

threat defense API 버전은 threat defense 소프트웨어의 각 주요 릴리스마다 변경됩니다. 새로운 기능은 추가 또는 변경되는 기능에 대한 API 호출에 영향을 줍니다.

그러나, 많은 기능은 릴리스에서 릴리스로 변경되지 않습니다. 예를 들어 네트워크 및 포트 개체와 관련된 API는 종종 새 릴리스에서 변경되지 않은 상태로 유지됩니다.

threat defense 버전 6.7부터는 기능에 대한 API 리소스 모델이 릴리스 간에 변경되지 않는 경우 threat defense API는 이전 API 버전을 기반으로 하는 통화를 수락할 수 있습니다. 기능 모델이 변경된 경우에도 이전 모델을 새 모델로 변환하는 논리적 방법이 있는 경우 이전 콜을 사용할 수 있습니다. 예를 들어, v5 통화는 v6 시스템에서 허용될 수 있습니다. "최신"을 통화의 버전 번호로 사용하는 경우 이러한 "이전" 통화는 이 시나리오에서 v6 통화로 해석되므로 이전 버전과의 호환성을 활용하고 있는지 여부는 API 호출을 어떻게 구성할지에 따라 달라집니다.

이전 버전과의 호환성을 지원할 수 없는 방식으로 API 버전 간에 기능 모델이 변경된 경우, 오류 메시지가 표시되며 이러한 오류를 확인하고 해당 특정 통화에 대한 코드를 업데이트해야 합니다.



2 장

API Explorer

API Explorer를 사용하여 REST API에 대해 알아봅니다. 또한, 다양한 방법 및 리소스를 테스트하여 올바르게 구성하고 있는지 확인할 수 있습니다. 코드에 시작점으로 JSON 모델을 복사하여 붙여넣을 수 있습니다.

팁 API Explorer의 목적은 API에 대해 알아보는 데 도움이 되는 것입니다. API Explorer를 통한 호출을 테스트하려면 일반적인 작업을 방해할 수 있는 액세스 잠금을 생성해야 합니다. 프로덕션이 아닌 디바이스에서 API Explorer를 사용하는 것이 좋습니다.

API Explorer 열기

API Explorer에서는 프로그래밍 용도로 사용할 수 있는 모든 리소스 및 JSON 개체에 대해 설명합니다. API Explorer는 각 개체의 '특성-값' 쌍에 대한 자세한 정보를 제공합니다. 따라서 다양한 HTTP 방법을 실험하여 각 리소스를 사용하는 데 필요한 코딩을 파악할 수 있습니다.

프로시저

단계 **1** 브라우저를 사용하여 시스템의 홈페이지(예: https://ftd.example.com)를 엽니다.

단계 **2** device manager에 로그인합니다.

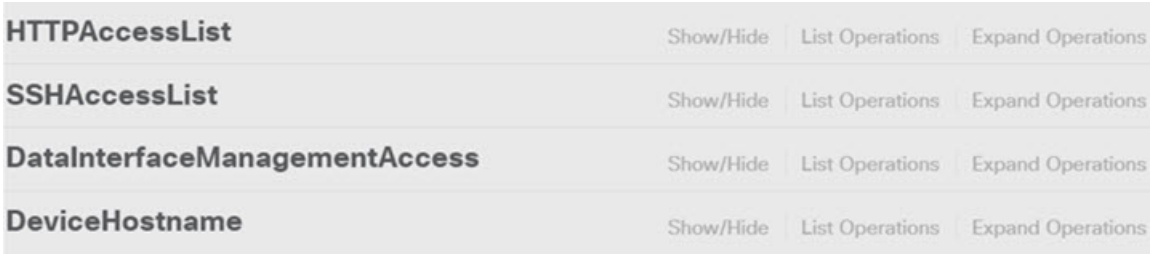
단계 **3** (6.4 및 이전 버전) /#/api-explorer를 가리키도록 URL을 편집합니다(예: https://ftd.example.com/#/api-explorer).

단계 **4** (6.5 이상 버전) More options(추가 옵션) 버튼(⋮)을 클릭하고 **API Explorer**를 선택합니다.

브라우저 설정에 따라 별도의 탭 또는 창에 API Explorer가 열립니다.

API Explorer 사용 방법

API Explorer를 시작하면 리소스 그룹의 목록이 함께 표시됩니다. 이 그룹에는 API에서 사용할 수 있는 리소스가 포함되어 있습니다. 다음 그림에는 목록의 일부 샘플이 나와 있습니다.

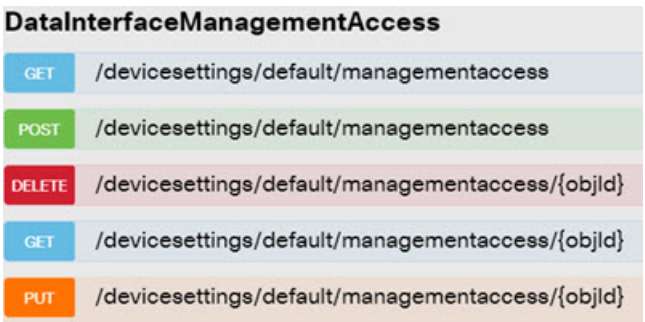


이러한 그룹 이름은 링크입니다. 링크를 클릭하여 그룹을 열면 해당 그룹의 리소스에 사용할 수 있는 방법이 표시됩니다. 각 그룹의 오른쪽에는 다음과 같은 명령도 있습니다.

- **Show/Hide(**표시/숨기기**)** - 그룹을 열고 닫습니다. 이는 그룹 이름을 클릭하는 것과 같습니다. 그룹을 처음 펼치면 방법이 간단히 표시(**List Operations(**작업 나열**)**와 동일)되지만, 그 이후에는 시스템에서 그룹을 닫기 전에 마지막으로 펼쳤던 상태를 기억하여 그룹을 다시 열 때 이전 상태 그대로 펼칩니다.
- **List Operations(**작업 나열**)** - 그룹의 각 리소스에 사용할 수 있는 HTTP 방법을 표시합니다. 이 정보에는 각 리소스의 URL(Universal Resource Locator) 템플릿에 대한 상대 경로가 포함되어 있습니다. 경로 변수는 표준 규칙인 {*variable*} 형식으로 표시되며, {*variable*}(중괄호 포함)은 적절한 값으로 대체해야 합니다. 기본 URL을 이 상대 경로에 추가해야 합니다. API의 기본 URL, 2 페이지의 내용을 참조하십시오.

  해당 방법에 대한 전체 설명서를 참조하려면 작업 URL 템플릿을 클릭합니다.
- **Expand Operations(**작업 확장**)** - 그룹에서 사용 가능한 모든 HTTP 방법 및 리소스를 엽니다.

하위 리소스가 많이 있는 그룹도 있습니다. 예를 들어, DataInterface ManagementAccess 그룹은 /devicesettings/default/managementaccess에 대한 GET, POST 및 DELETE 작업과 /devicesettings/default/managementaccess/{objId}에 대한 GET 및 PUT 작업을 포함합니다.



리소스 설명서 보기

각 리소스의 속성에 대한 설명서가 API Explorer에 있습니다.

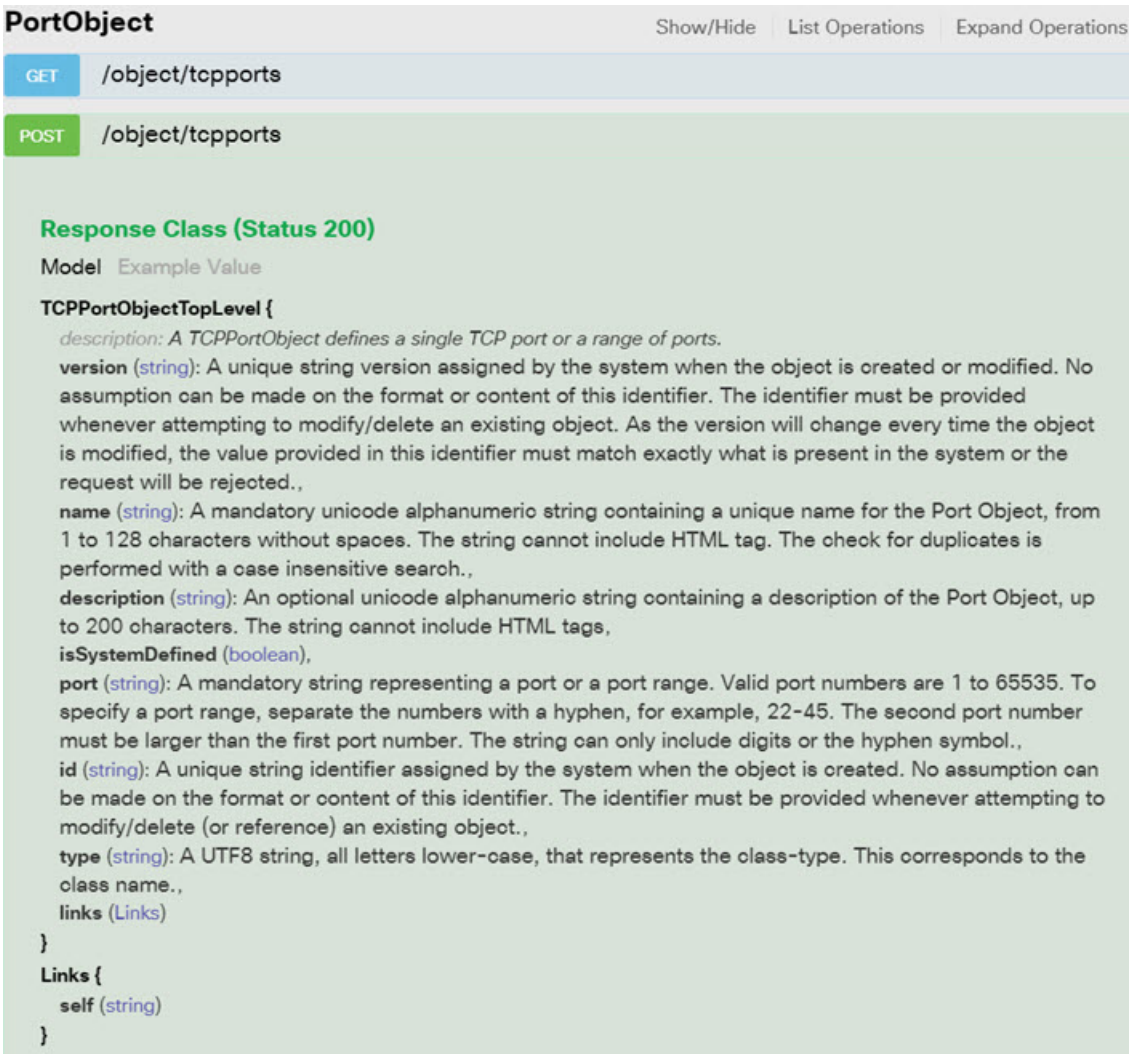
프로시저

단계 **1** 관심 있는 특정 리소스 및 방법에 대한 내용을 드릴다운합니다.

단계 **2** **Response Class(**응답 클래스**)** 섹션에서 **Model(**모델**)** 탭을 클릭합니다.

모델에는 설명 및 데이터 유형과 함께 특성이 나열되어 있습니다. GET의 경우 반환될 수 있는 paging 옵션도 있습니다. 응답에 반환된 개체보다 더 많은 개체가 있는 경우, 개체의 다음 및 이전 배치용 URL을 얻습니다.

예를 들어, 다음 그래픽에는 **Model(**모델**)** 탭이 선택된 상태의 POST /object/tcpports 방법 및 리소스가 나와 있습니다. 기본적으로 **Example Value(**예시 값**)** 탭이 선택되므로 설명서를 보려면 항상 **Model(**모델**)**을 클릭해야 합니다.



개체 ID(objId) 및 상위 ID 찾기

일부 리소스에는 다음과 같이 URL에 개체 ID 또는 관련된 상위 개체 ID가 필요합니다.

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

대부분의 경우 GET 방법을 사용하여 리소스 계층 구조에서 한 레벨 높은 개체 또는 상위 ID를 얻을 수 있습니다. 개체/상위 ID는 해당 개체에 대한 **id** 파라미터의 UUID입니다.

예를 들어 GET /object/networks는 현재 정의된 모든 네트워크 개체의 목록을 반환합니다. 원하는 개체로 이동하려면 호출을 여러 번 수행하여 목록 전체를 확인해야 할 수 있습니다. 또는 **limit** 쿼리 파

라미터를 포함하여 한 번의 호출에 반환되는 개체 수를 늘려야 할 수 있습니다. 각 개체의 형식은 다음과 같습니다(개체 ID는 강조 표시되어 있음).

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

참고 가끔 {objId}가 계층 구조의 최상위 레벨에서 발생하는 경우가 있습니다. 이러한 경우, 때때로 개체 ID에 대한 값을 입력하여 동일한 결과를 얻을 수 있습니다. 다른 경우에는 개체 모델 설명서에서 유효한 개체 유형(ID는 유효한 유형 중 하나)에 대한 내용을 확인하십시오. 이는 항상 GET 호출 (예: GET /operational/systeminfo/{objId} 및 GET /operational/featureinfo/{objId})입니다.

오류 카탈로그 보기 및 오류 메시지 평가

REST API로서, 시스템에서는 존재하지 않는 개체에 대해 GET을 수행하는 경우 404 같은 표준 HTTP 오류 코드를 반환합니다.

또한, 시스템에는 더 구체적으로 오류를 설명하는 여러 오류 메시지가 포함되어 있습니다. API 호출로 인해 오류가 발생하는 경우 응답 본문에는 더 구체적인 메시지가 포함될 수 있습니다.

예를 들어 다음 네트워크 개체에 대해 **POST /object/networks** 작업을 시도하면 오류 메시지가 표시됩니다. 이 경우 네트워크를 지정하려고 시도하지만 넷마스크를 포함하지는 않습니다. 즉, 값은 10.10.10.0/24 또는 10.10.10.0/255.255.255.0 중 하나여야 합니다.

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

그 결과 HTTP 응답 코드는 422이고, 특정 오류 메시지를 포함하는 응답 본문은 다음과 같습니다.

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

다음 절차에서는 응답 본문에서 반환될 수 있는 오류 메시지의 목록을 보는 방법에 대해 설명합니다.

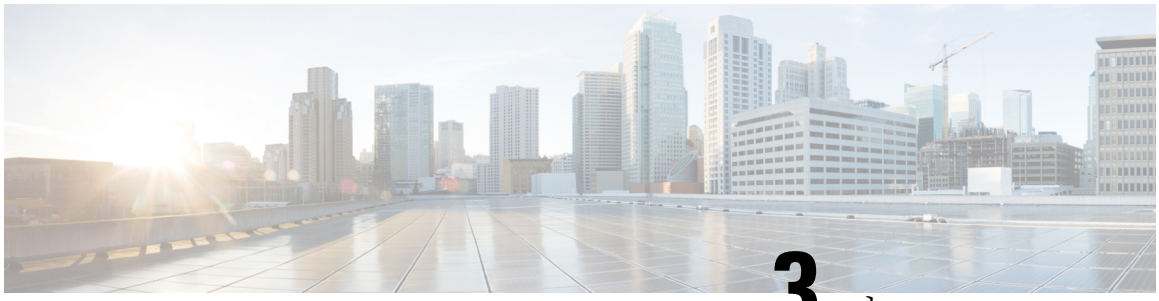
프로시저

단계 **1** device manager에서 More options(추가 옵션) 버튼(⋮)을 클릭하고 **API Explorer**를 선택합니다.

단계 **2** 목차에서 **Error Catalog(**오류 카탈로그**)**를 클릭합니다.

메시지에는 다음과 같은 구성 요소가 포함됩니다.

- **Category(**카테고리**)** - 메시지의 일반적인 유형입니다. 이 값은 오류 응답 본문의 **key** 특성에 표시됩니다. 카테고리에는 Validation(검증), General(일반) 및 Deployment(구축)가 포함됩니다.
- **Code(**코드**)** - 오류 메시지를 식별하는 고유한 문자열입니다. 이 값은 오류 응답 본문의 **code** 특성에 표시됩니다. 브라우저의 Find On Page(페이지에서 찾기) 기능을 사용하면 이 값을 사용하여 카탈로그에서 메시지를 찾을 수 있습니다.
- **Message(**메시지**)** - 오류를 설명하는 특정한 메시지입니다. 이 값은 오류 응답 본문의 **description** 특성에 표시됩니다. 메시지 내의 변수는 {0}, {1} 등으로 표시됩니다.



3 장

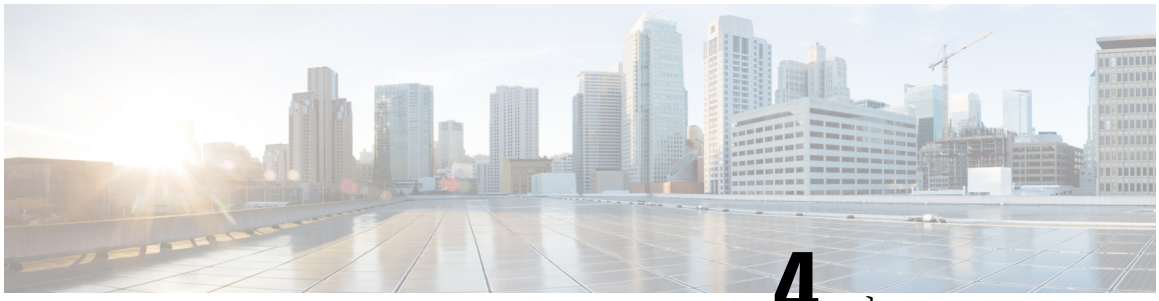
REST API 사용을 위한 일반 프로세스

REST API 사용을 위한 일반 프로세스

일반적으로 클라이언트는 다음과 같은 반복 프로세스를 사용하여 위협 방어 디바이스와 상호 작용합니다.

1. 액세스 토큰을 획득하여 API 호출을 인증합니다. API 클라이언트 인증 프로세스의 개요, 13 페이지의 내용을 참조하십시오.
2. 간단하게 데이터를 읽을 때를 제외하고 JSON 페이로드를 구성합니다.
3. 리소스의 URL(Universal Resource Locator)에 대한 HTTPS 호출을 사용하여 JSON 페이로드를 전송합니다.
4. 반환된 JSON 응답을 사용합니다.
5. 컨피그레이션을 변경한 후 변경 사항을 구축합니다. 컨피그레이션 변경 사항 구축, 49 페이지의 내용을 참조하십시오.

REST API 사용을 위한 일반 프로세스



4 장

OAuth를 사용한 REST API 클라이언트 인증

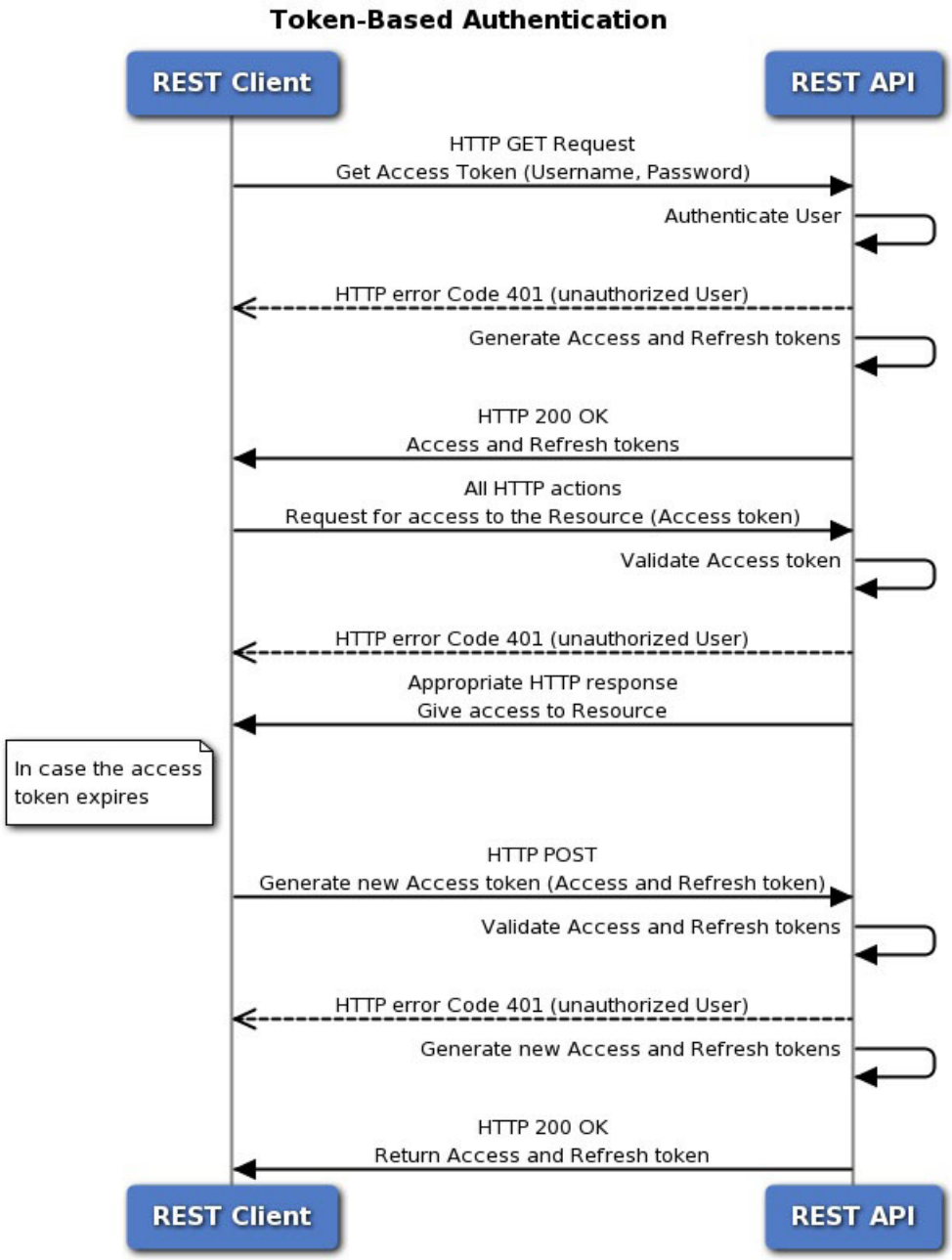
위협 방어 REST API에서는 API 클라이언트의 호출을 인증하는 데 Oauth 2.0을 사용합니다. OAuth는 액세스 토큰 기반 메서드이며, 위협 방어에서는 스키마에 JSON 웹 토큰을 사용합니다. 관련 표준은 다음과 같습니다.

- RFC6749, OAuth 2.0 Authorization Framework, https://tools.ietf.org/html/rfc6749
- RFC7519, JWT(JSON Web Token), https://tools.ietf.org/html/rfc7519

다음 주제에서는 필수 토큰을 획득하고 사용하는 방법에 대해 설명합니다.

API 클라이언트 인증 프로세스의 개요

다음은 API 클라이언트를 위협 방어 디바이스로 인증하는 방법에 대한 엔드 투 엔드 보기입니다.



시작하기 전에

각 토큰은 API 세션 및 device manager 세션에 포함되는 HTTPS 로그인 세션을 나타냅니다. 액티브 HTTPS 세션은 최대 5개까지 가능합니다. 이 제한을 초과하면 가장 오래된 세션인 device manager 로그인 또는 API 토큰이 만료되어 새 세션을 허용합니다. 따라서 필요한 토큰만 가져와 각 토큰이 만료될 때까지 재사용한 다음 갱신해야 합니다. 각 API 호출에 대해 새 토큰을 가져오면 심각한 세션 이탈이 발생하여 사용자를 device manager에서 잠글 수 있습니다. 이러한 제한은 SSH 세션에 적용되지 않습니다.

프로시저

단계 **1** 필요한 방법을 사용하여 API 클라이언트 사용자를 인증합니다.

클라이언트는 사용자를 인증해야 할 의무가 있으며 위협 방어 디바이스에 대한 액세스 및 수정 권한이 있어야 합니다. 부여 권한에 따라 차별적으로 기능을 제공하려면 해당 기능을 클라이언트에 구축해야 합니다.

예를 들어, 읽기 전용 액세스를 허용하려면 필요한 인증 서버, 사용자 어카운트 등을 설정해야 합니다. 그리고 나서 읽기 전용 권한을 가진 사용자가 클라이언트에 로그인할 때 GET 호출만 실행해야 합니다. API v1에서 이러한 유형의 변수 액세스는 위협 방어 디바이스에서 자체적으로 제어할 수 없습니다. API v2부터는 외부 사용자를 사용 중이며 사용자 권한 부여를 기준으로 호출을 정리하지 않는 경우 시도하는 호출과 사용자 권한 부여가 불일치하면 오류가 발생합니다.

v1의 경우 디바이스와 통신할 때 위협 방어 디바이스의 관리 사용자 계정을 사용해야 합니다. **admin** 어카운트에는 사용자가 구성 가능한 모든 개체에 대한 전체 읽기/쓰기 권한이 있습니다.

단계 **2** **admin(**관리자**)** 어카운트를 사용하여 사용자 이름/비밀번호를 기반으로 비밀번호 부여 액세스 토큰을 요청합니다.

비밀번호 부여 액세스 토큰 요청, 15 페이지의 내용을 참조하십시오.

단계 **3** 선택적으로 클라이언트용 맞춤형 액세스 토큰을 요청합니다.

맞춤형 토큰을 사용하면 유효 기간을 명시적으로 요청하고 토큰에 주체 이름을 할당할 수 있습니다. 맞춤형 액세스 토큰 요청, 17 페이지의 내용을 참조하십시오.

단계 **4** Authorization: Bearer(권한 부여: 전달자) 헤더의 API 호출에 액세스 토큰을 사용합니다.

API 호출에서 액세스 토큰 사용, 19 페이지의 내용을 참조하십시오.

단계 **5** 액세스 토큰이 만료되기 전에 토큰을 새로 고칩니다.

액세스 토큰 새로 고침, 20 페이지의 내용을 참조하십시오.

단계 **6** 작업을 완료한 경우 액세스 토큰이 아직 만료되지 않았다면 토큰을 취소합니다.

액세스 토큰 취소, 21 페이지의 내용을 참조하십시오.

비밀번호 부여 액세스 토큰 요청

모든 REST API 호출에는 호출자가 요청한 작업을 수행할 수 있는 권한이 있는지 확인하는 인증 토큰이 포함되어 있어야 합니다. 처음에는 **admin** 사용자 이름/비밀번호를 제공하여 액세스 토큰을 획득해야 합니다. 이를 비밀번호 부여 액세스 토큰(즉, grant_type = password)이라고 합니다.

프로시저

단계 **1** 비밀번호 부여 액세스 토큰을 부여받기 위해 JSON 개체를 생성합니다.

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

admin 사용자 이름 및 올바른 비밀번호를 지정합니다. 아래에 예시가 나와 있습니다.

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

단계 **2** POST /fdm/token을 사용하여 액세스 토큰을 획득합니다.

예를 들어 **curl** 명령은 다음과 같이 표시될 수 있습니다.

```
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/최신/fdm/token'
```

단계 **3** 액세스를 검색하고 응답에서 토큰을 새로 고칩니다.

양호한 응답(상태 코드 200)은 다음과 같이 표시됩니다.

```
{
  "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
}
```

여기서 각 항목은 다음을 나타냅니다.

- **access_token** API 호출에서 포함해야 하는 전달자 토큰입니다. API 호출에서 액세스 토큰 사용, 19 페이지의 내용을 참조하십시오.

- **expires_in** 토큰이 발행된 시간 이후부터 액세스 토큰이 유효한 시간(초)입니다.
- **refresh_token** 새로 고침 요청에서 사용할 토큰입니다. 액세스 토큰 새로 고침, 20 페이지의 내용을 참조하십시오.
- **refresh_expires_**in 새로 고침 토큰이 유효한 시간(초)입니다. 이는 항상 액세스 토큰 유효 기간보다 깁니다.

맞춤형 액세스 토큰 요청

비밀번호 부여 액세스 토큰을 사용할 수 있습니다. 그러나 맞춤형 액세스 토큰도 요청할 수 있습니다. 맞춤형 토큰을 사용하면 주체 이름을 제공하여 고유한 목적에 대해 차별적으로 토큰을 사용하는 데 도움이 될 수 있습니다. 비밀번호 토큰에 반환된 기본값이 요건에 맞지 않는 경우에는 특정한 유효 기간을 요청할 수도 있습니다.

시작하기 전에

맞춤형 토큰을 획득하기 전에 먼저 비밀번호 부여 액세스 토큰을 획득해야 합니다. 비밀번호 부여 액세스 토큰 요청, 15 페이지의 내용을 참조하십시오.

그 외에도,

- 로컬 사용자인 경우 맞춤형 토큰을 요청할 수 있습니다. 외부 사용자는 맞춤형 토큰을 요청할 수 없습니다.
- 토큰을 가져올 수 있는 유닛에서만 맞춤형 토큰을 사용할 수 있습니다. 고가용성 그룹의 피어 디바이스에서는 토큰을 사용할 수 없습니다.

프로시저

단계 **1** 맞춤형 액세스 토큰을 부여받기 위해 JSON 개체를 생성합니다.

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

여기서 각 항목은 다음을 나타냅니다.

- **access_token** 유효한 비밀번호 부여 액세스 토큰입니다.

- **desired_expires_in** 맞춤형 액세스 토큰이 유효한 시간(초)을 나타내는 정수입니다. 한편, 비밀번호 부여 토큰은 1800초 동안 유효합니다.
- **desired_refresh_expires_in** 맞춤형 새로 고침 토큰이 유효한 시간(초)을 나타내는 정수입니다. 새로고침 토큰을 가져오는 경우, 이 값이 **desired_expires_in** 값보다 커야 합니다. 한편, 비밀번호 부여 새로 고침 토큰은 2400초 동안 유효합니다. **desired_refresh_count**에 대해 0을 지정하는 경우에는 이 파라미터가 필요 없습니다.
- **desired_subject** 맞춤형 토큰에 지정한 이름입니다.
- **desired_refresh_count** 토큰을 새로 고칠 수 있는 횟수입니다. 새로 고침 토큰을 획득하지 않으려면 0으로 지정합니다. 새로 고침 토큰이 없는 경우 기존 토큰이 만료되면 새 액세스 토큰을 획득해야 합니다.

예를 들어, 다음은 2400초 후 만료되는 api-client에 대한 맞춤형 토큰을 요청하는 내용입니다. 이 예에서 새로 고침 토큰은 3000초 후 만료됩니다. 토큰은 3번을 새로 고칠 수 있습니다.

```
{{
  "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
}
```

단계 **2** POST /fdm/token을 사용하여 액세스 토큰을 획득합니다.

예를 들어 **curl** 명령은 다음과 같이 표시될 수 있습니다.

```
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/최신/fdm/token'
```

단계 **3** 액세스를 검색하고 응답에서 토큰을 새로 고칩니다.

양호한 응답(상태 코드 200)은 다음과 같이 표시됩니다.

```
{
```

```
  "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
}
```


여기서 각 항목은 다음을 나타냅니다.

- **access_token** API 호출에서 포함해야 하는 전달자 토큰입니다. API 호출에서 액세스 토큰 사용, 19 페이지의 내용을 참조하십시오.
- **expires_in** 토큰이 발행된 시간 이후부터 액세스 토큰이 유효한 시간(초)입니다.
- **refresh_token** 새로 고침 요청에서 사용할 토큰입니다. 액세스 토큰 새로 고침, 20 페이지의 내용을 참조하십시오.
- **refresh_expires**_in 새로 고침 토큰이 유효한 시간(초)입니다. 이는 항상 액세스 토큰 유효 기간보다 깁니다.

API 호출에서 액세스 토큰 사용

비밀번호 부여 액세스 토큰 또는 맞춤형 액세스 토큰 중 하나를 획득한 후에는 HTTPS 요청에 대한 **Authorization: Bearer(**권한 부여**:** 전달자**)** 헤더의 각 API 호출에 이 토큰을 포함해야 합니다.

예를 들어 GET /object/networks 작업을 수행하기 위한 **curl** 명령은 다음과 같이 표시될 수 있습니다.


```
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/최신/object/networks'
```

참고 API Explorer를 사용하여 메서드 및 리소스를 시도하는 경우, 표시되는 **curl** 명령에는 **Authorization: Bearer(**권한 부여**:** 전달자**)** 헤더가 포함되어 있지 않습니다. 그러나 API 클라이언트에서 호출할 때는 이 헤더를 추가해야 합니다.

액세스 토큰 새로 고침

액세스 토큰이 만료되고 나면 기존에 부여받았던 새로 고침 토큰을 사용하여 해당 토큰을 새로 고쳐야 합니다. 새로 고친 액세스 토큰은 실제로 기존의 액세스 토큰과 다릅니다. "새로 고침"을 수행하면 단순히 기존의 액세스 토큰의 수명이 연장되는 것이 아니라 실제로 새로운 쌍의 액세스 토큰과 새로 고침 토큰이 제공됩니다.

프로시저

단계 **1** 새로 고침 토큰을 부여받기 위해 JSON 개체를 생성합니다.

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

refresh_token은 암호 부여 액세스 토큰 또는 맞춤형 액세스 토큰에서 부여받을 수 있습니다.

예를 들면 다음과 같습니다.

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

단계 **2** POST /fdm/token을 사용하여 새로 고쳐진 액세스 토큰을 획득합니다.

예를 들어 **curl** 명령은 다음과 같이 표시될 수 있습니다.

```
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/최신/fdm/token'
```

단계 **3** 액세스를 검색하고 응답에서 토큰을 새로 고칩니다.

양호한 응답(상태 코드 200)은 다음과 같이 표시됩니다. 이 예에서는 새로 고침 토큰이 맞춤형 토큰에 사용되었습니다. 만료 기간은 기존 맞춤형 액세스 토큰 요청의 값을 기반으로 합니다.

```
{
  "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
}
```

여기서 각 항목은 다음을 나타냅니다.

- **access_token** API 호출에서 포함해야 하는 전달자 토큰입니다. API 호출에서 액세스 토큰 사용, 19 페이지의 내용을 참조하십시오.
- **expires_in** 토큰이 발행된 시간 이후부터 액세스 토큰이 유효한 시간(초)입니다.
- **refresh_token** 새로 고침 요청에서 사용할 토큰입니다.
- **refresh_expires**_in 새로 고침 토큰이 유효한 시간(초)입니다. 이는 항상 액세스 토큰 유효 기간보다 깁니다.

액세스 토큰 취소

액세스 토큰은 특정 기간 동안 유효하므로 API 클라이언트에서 로그아웃할 때 토큰을 취소하여 정리해야 합니다. 이렇게 하면 백도어가 위협 방어 디바이스로 열려 있지 않게 할 수 있습니다.

프로시저

단계 **1** 취소 토큰을 부여받기 위해 JSON 개체를 생성합니다.

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

여기서 각 항목은 다음을 나타냅니다.

- **access_token** 비밀번호 부여 액세스 토큰이어야 합니다. 맞춤형 액세스 토큰을 사용해서는 토큰을 취소할 수 없습니다.
- 다음 중 하나만 지정해야 합니다.
  - **token_to_revoke** 취소하려는 비밀번호 부여 토큰 또는 맞춤형 토큰입니다. 이것은 **access_token**과 동일한 토큰일 수 있으며, 이 경우에는 암호 부여 토큰을 사용하여 토큰을 자체적으로 취소할 수 있습니다.
  - (사용하지 않음) **custom_token_id_to_revoke**에서는 내부 고유 ID로 맞춤형 액세스 토큰을 식별합니다. 그러나 이 값을 얻을 수 있는 직접적인 방법은 없으므로 다른 옵션을 대신 사용합니다.
  - **custom_token_subject_to_revoke**는 취소하려는 맞춤형 액세스 토큰에 대한 **desired_subject** 값입니다.

예를 들면 다음과 같습니다.

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

단계 **2** POST /fdm/token을 사용하여 액세스 토큰을 취소합니다.

예를 들어 **curl** 명령은 다음과 같이 표시될 수 있습니다.

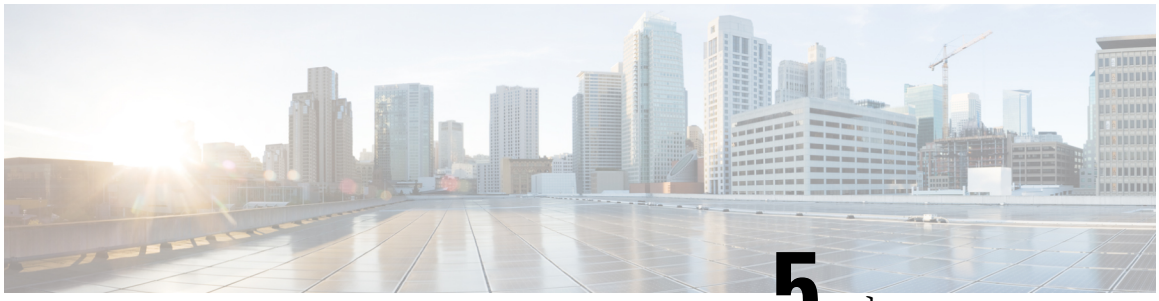
```
curl -X POST --header 'Content-Type: application/json'
--header 'Accept: application/json' -d '{
   "grant_type": "revoke_token",
   "access_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQ
```

```
iOjE1MDI5MDQzMjQsInN1YiI6ImFkbWluIiwianRpIjoiZTM
zNGIxOWYtODJhNy0xMWU3LWE4MWMtNGQ3NzY2ZTExMzVkIiw
ibmJmIjoxNTAyOTA0MzI0LCJleHAiOjE1MDI5MDYxMjQsInJ
lZnJlc2hUb2tlbkV4cGlyZXNBdCI6MTUwMjkwNjcyNDExMiw
idG9rZW5UeXBlIjoiSldUX0FjY2VzcyIsIm9yaWdpbiI6InB
hc3N3b3JkIn0.OVZBT9yVZc4zxZfZiiLH4SZcFclaHyCPbZJ
C_Gyd5FE",
   "custom_token_subject_to_revoke": "api-client"
 }' 'https://ftd.example.com/api/fdm/최신/fdm/token'
```

단계 **3** 응답을 평가하여 토큰이 취소되었는지 확인합니다.

양호한 응답(상태 코드 200)은 다음과 같이 표시됩니다.

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



5 장

API에 대한 외부 사용자 구성

버전 요구 사항: 외부 AAA를 사용하려면 threat defense 버전 6.3(0) 이상 및 threat defense REST API v2 이상을 실행해야 합니다.

외부 RADIUS AAA 서버를 사용하여 threat defense REST API에 대한 사용자 액세스를 인증하고 권한을 부여하도록 디바이스를 구성할 수 있습니다. 내장된 로컬 관리 사용자 어카운트 대신 또는 해당 어카운트와 함께 RADIUS 사용자 어카운트를 사용할 수 있습니다.

외부 AAA 사용할 때 각기 다른 권한 부여 레벨을 사용하도록 어카운트를 정의할 수 있습니다. 그러면 디바이스 컨피그레이션을 변경할 수 있는 사용자를 제한하는 동시에 지원 담당자에게는 읽기 전용 액세스 권한을 계속 제공할 수 있습니다.

다음 절차에서는 RADIUS 어카운트를 설정한 다음 인증과 권한 부여에 외부 AAA를 사용하도록 디바이스를 구성하는 전체 프로세스를 설명합니다.

시작하기 전에

외부 권한 부여를 사용할 때는 다음의 작동 요소에 유의해야 합니다.

- 디바이스의 고가용성이 구성된 경우에는 액티브 유닛에서 외부 권한 부여를 구성합니다. 그런 다음 권한 부여 설정에 대한 구축 작업을 실행하여 스탠바이 디바이스에 대한 사용자 액세스를 허용해야 합니다.
- 새 사용자가 시스템에 액세스할 때마다 해당 사용자에 대한 사용자 리소스가 생성됩니다. 해당 사용자 개체를 저장하려면 컨피그레이션을 구축해야 합니다.

  (threat defense 6.6 이전 버전) HA(고가용성) 모드로 작동하는 경우에는 구성을 구축해야 해당 사용자가 스탠바이 유닛에 로그인할 수 있습니다. 관리자 또는 읽기-쓰기 사용자만 구축 작업을 시작할 수 있으므로 최초 읽기 전용 사용자는 다른 사용자가 컨피그레이션을 구축하여 사용자 개체를 저장하도록 해야 합니다.

  threat defense 6.6부터는 HA 제한 사항이 제거됩니다. 외부 사용자는 먼저 액티브 유닛에 로그인하고 구성을 구축하지 않아도 스탠바이 유닛에 로그인할 수 있습니다. 사용자 개체는 스탠바이 유닛에서 생성되지 않지만, 유효한 사용자 이름/비밀번호가 제공되었다고 가정할 경우 사용자에게는 액세스 권한이 부여되며 사용자 특성은 캐시됩니다.

프로시저

단계 **1** RADIUS 사용자 어카운트에 대한 권한 부여 권한 정의, 26 페이지.

단계 **2** RADIUS 서버 정의, 27 페이지.

단계 **3** RADIUS 서버용 AAA 서버 그룹 생성, 28 페이지.

단계 **4** HTTP 액세스를 위한 인증 소스로 AAA 서버 그룹 설정, 30 페이지.

단계 **5** **POST /operational/deploy**를 사용하여 구축 작업을 시작합니다.

curl 명령은 다음과 같습니다.

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

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

변경 사항 구축에 대한 자세한 내용은 컨피그레이션 변경 사항 구축, 49 페이지의 내용을 참조하십시오.

단계 **6** 외부 사용자 액세스 확인, 34 페이지.

RADIUS 사용자 어카운트에 대한 권한 부여 권한 정의

외부 RADIUS 서버에서 threat defense REST API에 액세스할 수 있는 권한을 제공할 수 있습니다. RADIUS 인증 및 권한 부여를 활성화하면 각기 다른 액세스 권한 레벨을 제공할 수 있으며, 모든 사용자가 로컬 관리자 어카운트를 통해 로그인하지 못하게 할 수 있습니다.

참고 이러한 외부 사용자에게 device manager에 대한 권한도 부여할 수 있습니다.

RBAC(Role-Based Access Control)를 제공하려면 RADIUS 서버에서 사용자 어카운트를 업데이트하여 **cisco-av-pair** 특성을 정의합니다. 사용자 어카운트에 대해 이러한 특성을 정확하게 정의해야 합니다. 그렇지 않으면 해당 사용자의 REST API 액세스가 거부됩니다. cisco-av-pair 특성에 대해 지원되는 값은 다음과 같습니다.

- **fdm.userrole.authority.admin**은 전체 관리자 액세스를 제공합니다. 이러한 사용자는 로컬 관리 사용자가 수행할 수 있는 모든 작업을 수행할 수 있습니다.
- **fdm.userrole.authority.rw**는 읽기-쓰기 액세스를 제공합니다. 이러한 사용자는 읽기 전용 사용자가 수행할 수 있는 모든 작업을 수행할 수 있으며 컨피그레이션 수정 및 구축도 수행할 수 있

습니다. 업그레이드 설치, 백업 생성 및 복원, 감사 로그 확인, 다른 사용자 로그오프를 포함하는 시스템의 중요 작업만 제한됩니다.

- **fdm.userrole.authority.ro**는 읽기 전용 액세스를 제공합니다. 이러한 사용자는 대시보드 및 컨피그레이션을 볼 수는 있지만 변경할 수는 없습니다. 사용자가 변경을 시도하면 권한이 없음을 설명하는 오류 메시지가 표시됩니다.

RADIUS 서버 정의

적절한 권한 부여 권한을 정의하도록 RADIUS 서버의 사용자 어카운트를 구성한 후에는 서버를 사용하여 REST API에 대한 액세스를 인증하고 권한을 부여하도록 디바이스를 구성할 수 있습니다.

POST /object/radiusidentitysources 리소스를 사용하여 정의하려는 각 RADIUS 서버용 개체를 생성합니다.

프로시저

단계 **1** RADIUS 서버용 JSON 개체 본문을 생성합니다.

이 호출에 사용할 JSON 개체의 예는 다음과 같습니다.

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

특성은 다음과 같습니다.

- **name(**이름**)** - 개체 이름입니다. 이 이름은 RADIUS 서버에 정의된 항목과 일치하지 않아도 됩니다.
- **description(**설명**)** - (선택 사항). 개체의 설명입니다.
- **host(**호스트**)** - RADIUS 서버의 IP 주소 또는 정규화된 호스트 이름입니다.
- **timeout(**시간 제한**)** - (선택 사항). 시스템이 다음 서버로 요청을 보내기 전까지 서버의 응답을 기다리는 시간(1~300초)입니다. 이 특성을 포함하지 않는 경우의 기본값은 10초입니다.
- **serverAuthenticationPort** - (선택 사항). RADIUS 인증 및 권한 부여가 수행되는 포트입니다. 이 특성을 포함하지 않는 경우의 기본값은 1812입니다.
- **serverSecretKey** - (선택 사항). threat defense 디바이스와 RADIUS 서버 간의 데이터를 암호화하는 데 사용되는 공유 암호입니다. 이 키는 대/소문자를 구분하며 공백은 포함하지 않는 영숫자 문자열(최대 64자)입니다. 또한 영숫자 문자 또는 밑줄로 시작해야 하며 특수 문자 $ & - _ . + @

는 포함할 수 없습니다. 문자열은 RADIUS 서버에 구성된 것과 일치해야 합니다. 비밀 키를 구성하지 않으면 연결이 암호화되지 않습니다.

단계 **2** 개체를 게시합니다.

예를 들어 **curl** 명령은 다음과 같이 표시됩니다.

```
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/최신/object/radiusidentitysources'
```

단계 **3** 응답을 확인합니다.

응답 코드 200이 표시되어야 합니다. 올바른 응답 본문은 다음과 같이 표시됩니다. 비밀 키와 같은 민감한 정보는 응답에서 마스크 처리됩니다.

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

RADIUS 서버용 AAA 서버 그룹 생성

RADIUS 서버 개체를 생성한 후에는 **POST /object/radiusidentitysourcegroups** 리소스를 사용하여 radiusidentitysource 개체를 포함할 AAA 그룹을 생성합니다.

RADIUS AAA 서버 그룹에는 RADIUS 서버를 16개까지 추가할 수 있습니다. 이러한 서버는 서로의 백업이어야 합니다. 즉, 서버의 사용자 어카운트 목록이 같아야 합니다.

프로시저

단계 **1** RADIUS 서버 그룹용 JSON 개체 본문을 생성합니다.

이 호출에 사용할 JSON 개체의 예는 다음과 같습니다.

```
{
  "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"
}
```

특성은 다음과 같습니다.

- **name(**이름**)** - 개체 이름입니다. 이 이름은 멤버 RADIUS 서버에 정의된 항목과 일치하지 않아도 됩니다.
- **maxFailedAttempts** - (선택 사항). 실패한 서버는 모든 서버가 실패한 후에야 비로소 재활성화됩니다. 비활성 시간이란 마지막 서버가 실패한 이후, 모든 서버를 재활성화하기 이전의 대기 시간(0~1440분)입니다. 이 특성을 포함하지 않는 경우 기본값은 10분입니다.
- **deadTime** - (선택 사항). 다음 서버 사용을 시도하기 전에 그룹의 RADIUS 서버로 전송되었으나 실패한 요청(즉, 응답을 받지 못한 요청)의 수입니다. 1~5 사이의 값을 지정할 수 있으며 기본값은 3입니다. 최대 실패 시도 횟수가 초과되면 시스템에서 해당 서버를 Failed(장애 발생)로 표시합니다.

  특정 기능에 대해 로컬 데이터베이스를 사용하여 대체 방법을 구성했는데 그룹의 모든 서버가 응답하지 않으면 해당 그룹은 응답이 없는 것으로 간주되고 대체 방법을 시도합니다. 서버 그룹은 비활성 시간 동안 응답이 없는 것으로 표시됩니다. 따라서 이 기간 내에는 추가 AAA 요청이 서버 그룹 연결을 시도하지 않으며 대체 방법이 즉시 사용됩니다.
- **description(**설명**)** - (선택 사항). 개체의 설명입니다.
- **radiusIdentitySources** - 그룹에 포함할 RADIUS 서버를 정의하는 각 radiusidentitysource 개체를 정의하는 항목의 그룹입니다. [괄호] 안에 항목을 포함합니다. 아래에는 각 개체의 특성과 구문이 나와 있습니다. 개별 개체에서 **id**, **version(**버전**)** 및 **name(**이름**)** 속성의 값을 가져옵니다. 개체 생성 시 정보는 응답 본문에 있습니다. **GET /object/radiusidentitysources** 호출에서 정보를 가져올 수도 있습니다. **type(**유형**)**은 **radiusidentitysource**이어야 합니다.

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

```
      "name": "aaa-server-1"
    }
```

단계 **2** 개체를 게시합니다.

예를 들어 **curl** 명령은 다음과 같이 표시됩니다.

```
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/최신/object/radiusidentitysourcegroups'
```

단계 **3** 응답을 확인합니다.

응답 코드 200이 표시되어야 합니다. 올바른 응답 본문은 다음과 같이 표시됩니다.

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

HTTP 액세스를 위한 인증 소스로 AAA 서버 그룹 설정

PUT /devicesettings/default/aaasettings/{objId} 리소스를 사용하여 RADIUS AAA 서버 그룹을 사용자 인증용 ID 소스로 식별합니다.

POST 메서드는 없으며 시스템 인증에 필요한 개체는 이미 있습니다. 먼저 GET을 수행하여 관련 ID 및 버전 값을 확인해야 합니다.

프로시저

단계 **1** **GET /devicesettings/default/aaasettings**를 사용하여 aaasettings 개체의 특성을 확인합니다.

curl 명령은 다음과 같습니다.

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

예를 들어 응답 본문은 다음과 같습니다. 이 예시에서는 로컬 ID 소스가 HTTPS 액세스용으로 정의된 소스임을 보여줍니다. 이 소스는 SSH 액세스(REST API와는 관련이 없음)에도 사용됩니다.

```
{
  "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/최신/
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/최신/
devicesettings/default/aaasettings/00000003-0000-0000-0000-000000000008"
      }
    }
  ],
  "paging": {
    "prev": [],
```

```
    "next": [],
    "limit": 10,
    "offset": 0,
    "count": 2,
    "pages": 0
  }
}
```

단계 **2** (선택 사항). **GET /devicesettings/default/aaasettings/{objId}**를 사용하여 HTTPS AAA 설정 개체 복사본을 가져와 보기의 범위를 좁힙니다.

PUT 호출에서는 HTTPS 개체만 업데이트됩니다. SSH 개체는 업데이트할 필요가 없습니다.

이 예시에서 HTTPS 개체의 ID는 00000003-0000-0000-0000-000000000007이므로 **curl** 명령은 다음과 유사합니다.

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

응답 본문은 다음과 유사합니다.

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

단계 **3** AAA 관리 액세스용 JSON 개체 본문을 생성합니다.

이 호출에 사용할 JSON 개체의 예는 다음과 같습니다.

```
{
  "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"
}
```

특성은 다음과 같습니다.

- **version(**버전**)** - HTTPS 개체의 버전입니다. GET 호출에 대한 응답 본문에서 이 값을 찾을 수 있습니다.
- **name(**이름**)** - 개체 이름(**HTTPS**)입니다. GET 호출에 대한 응답 본문에서 이 값을 찾을 수 있습니다.
- **identitySourceGroup** - RADIUS 서버 그룹을 식별하는 특성입니다. 서버 그룹 또는 **GET /object/radiusidentitysourcegroups** 호출을 생성한 경우, 응답 본문에서 **id**, **version(**버전**)** 및 **name(**이름**)** 값을 가져옵니다. 유형은 **radiusidentitysourcegroup**이어야 합니다.
- **description(**설명**)** - (선택 사항). 개체의 설명입니다.
- **protocolType** - 이 소스가 적용되는 프로토콜(**HTTPS**)입니다.
- **useLocal** - 로컬 관리 사용자 어카운트가 포함된 로컬 ID 소스를 사용할 방법입니다. 다음 옵션 중 하나를 입력합니다.
  - **Before(**전**)** - 시스템이 로컬 소스를 먼저 대조하여 사용자 이름과 비밀번호를 확인합니다.
  - **After(**후**)** - 외부 소스를 사용할 수 없거나 외부 소스에 사용자 어카운트가 없는 경우에만 로컬 소스를 확인합니다.
  - **Never(**사용 안 함**)** — (권장되지 않음.) 로컬 소스를 사용하지 않습니다. 따라서 관리 사용자로 로그인할 수 없습니다.

    주의 **Never(**사용 안 함**)**를 선택하는 경우 관리자 어카운트를 사용하여 device manager에 로그인하거나 API를 사용할 수 없습니다. RADIUS 서버를 사용할 수 없게 되거나 RADIUS 서버에서 어카운트를 잘못 구성하는 경우에는 시스템에서 해당 어카운트를 차단합니다.

- **id** - HTTPS 개체의 ID 값입니다. GET 호출에 대한 응답 본문에서 이 값을 찾을 수 있습니다.
- **type(**유형**)** - 개체 유형(**aaasetting**)입니다.

단계 **4** 개체를 배치합니다.

예를 들어 **curl** 명령은 다음과 같습니다. URL의 {objId}와 JSON 개체의 aaasettings 개체 id는 같습니다.

```
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/최신/devicesettings/
default/aaasettings/00000003-0000-0000-0000-000000000007'
```

단계 **5** 응답을 확인합니다.

응답 코드 200이 표시되어야 합니다. 올바른 응답 본문은 다음과 같이 표시됩니다.

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

외부 사용자 액세스 확인

구축 작업이 완료되고 나면 device manager와 REST API 모두에 대한 외부 사용자 액세스를 테스트할 수 있습니다.

프로시저

단계 **1** 유효한 cisco-av-pair 특성이 포함된 외부 사용자 이름을 사용하여 device manager에 로그인합니다.

로그인이 성공해야 하며 페이지 오른쪽 상단에 사용자 이름과 권한 레벨이 표시되어야 합니다.

단계 **2** 외부 사용자에 대한 REST API 토큰을 획득합니다.

토큰을 획득할 수 있는 사용자는 할당된 권한 레벨에 대해 허용되는 리소스와 메서드를 사용할 수 있습니다.

a) 단순 비밀번호 부여 토큰용 JSON 개체 본문을 생성합니다.

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

b) **POST /fdm/token**을 사용하여 토큰을 획득합니다.

```
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/최신/fdm/token'
```

c) 응답을 평가하여 토큰이 부여되었는지 확인합니다.

응답 코드 200이 표시되어야 합니다. 토큰을 획득했다는 것은 시스템에서 사용자를 인증할 수 있었다는 의미입니다.

```
{
  "access_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1Mjg4MjM
3MTAsInN1YiI6InJhZGl1c3JlYWR3cml0ZXVzZXIxIiwianRpIjoiMjk5Zj
Q5YjYtNmU2NC0xMWU4LWJkNjUtNmY0ZmVmYjY1MzI5IiwibmJmIjoxNTI4O
DIzNzEwLCJleHAiOjE1Mjg4MjU1MTAsInJlZnJlc2hUb2tlbkV4cGlyZXNB
dCI6MTUyODgyNjExMDg4OSwidG9rZW5UeXBlIjoiSldUX0FjY2VzcyIsInV
zZXJVdWlkIjoiMjliMjBlNjctNmU2NC0xMWU4LWJkNjUtMzU4MmUwZjU5Yj
Q4IiwidXNlclJvbGUiOiJST0xFX1JFQURfV1JJVEUiLCJvcmlnaW4iOiJwY
XNzd29yZCJ9.dtKsl9IB4ds3RAktEeaSuQy_Zs2SrzLr976UtblBt28",
  "expires_in": 1800,
  "token_type": "Bearer",
  "refresh_token": "eyJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1Mjg4Mj
M3MTAsInN1YiI6InJhZGl1c3JlYWR3cml0ZXVzZXIxIiwianRpIjoiMjk5Z
jQ5YjYtNmU2NC0xMWU4LWJkNjUtNmY0ZmVmYjY1MzI5IiwibmJmIjoxNTI4
ODIzNzEwLCJleHAiOjE1Mjg4MjYxMTAsImFjY2Vzc1Rva2VuRXhwaXJlc0F
0IjoxNTI4ODI1NTEwODg5LCJyZWZyZXNoQ291bnQiOi0xLCJ0b2tlblR5cG
UiOiJKV1RfUmVmcmVzaCIsInVzZXJVdWlkIjoiMjliMjBlNjctNmU2NC0xM
WU4LWJkNjUtMzU4MmUwZjU5YjQ4IiwidXNlclJvbGUiOiJST0xFX1JFQURf
V1JJVEUiLCJvcmlnaW4iOiJwYXNzd29yZCJ9.Lc7MYmieNMMrjx7XoTiW-x
8Z8qFCnzfNM1apgbwLQvo",
  "refresh_expires_in": 2400
}
```

단계 **3** **GET /object/users**를 사용하여 각 사용자에 대해 사용자 개체가 생성되었는지 확인합니다.

사용자 개체는 device manager에 로그인하거나 액세스 토큰을 획득하는 각 새 사용자에 대해 자동으로 생성됩니다. 이러한 사용자 개체를 저장하려면 구축 작업을 실행해야 합니다. 고가용성 모드에서는 구축 작업을 실행해야 사용자가 스탠바이 유닛에 로그인할 수 있습니다.

예를 들어 **curl** 명령은 다음과 같이 표시됩니다.

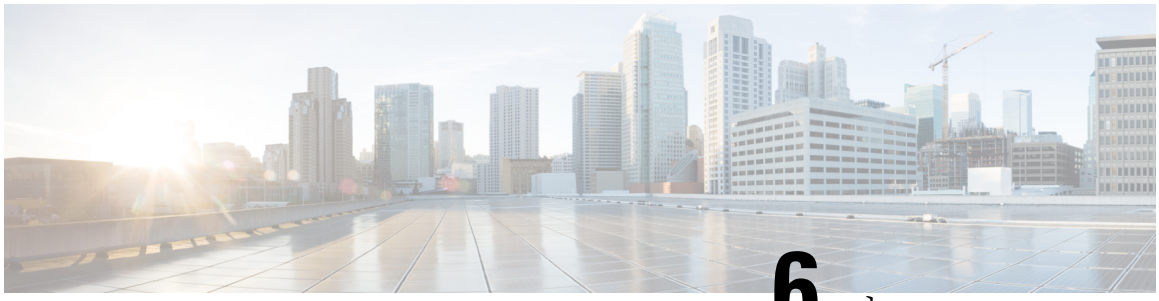
```
curl -X GET --header 'Accept: application/json'
'https://ftd.example.com/api/fdm/최신/object/users'
```

다음 응답 본문에는 두 외부 사용자가 로그인했음이 표시됩니다. **userRole**은 이러한 사용자 어카운트에 대해 RADIUS 서버에 구성된 cisco-av-pair에서 획득한 권한을 표시합니다. 이 정보를 사용하여 RADIUS 사용자 어카운트를 정확하게 구성했는지 확인합니다. 관리 사용자는 로컬에 정의된 사용자입니다.

```
{
  "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/최신/
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/최신/
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/최신/
object/users/29b20e67-6e64-11e8-bd65-3582e0f59b48"
      }
    }
```

```
  ],
  "paging": {
    "prev": [],
    "next": [],
    "limit": 10,
    "offset": 0,
    "count": 3,
    "pages": 0
  }
}
```



6 장

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 찾기, 8 페이지의 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, 41 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 찾기, 8 페이지의 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, 39 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, 39 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, 41 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, 39 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, 41 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.

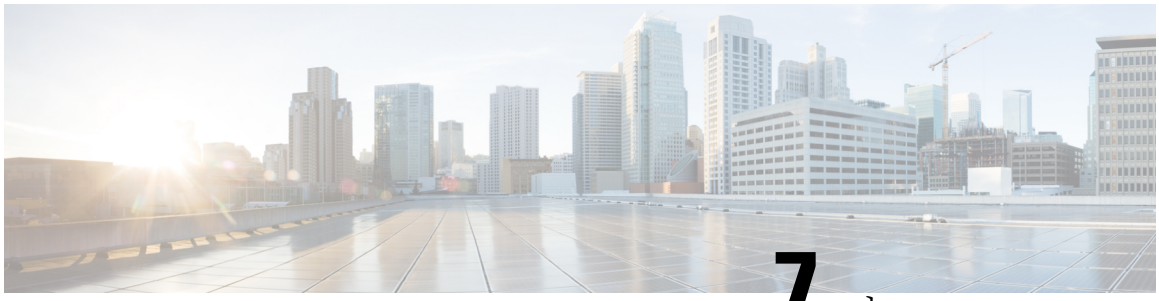
개체가 컨테이너 내에 있는 경우에는 **parentId** 필드에 상위 개체 ID도 입력해야 합니다.

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.



7 장

컨피그레이션 변경 사항 구축

컨피그레이션 변경 사항 구축

POST, PUT, DELETE 호출을 통해 위협 방어 디바이스가 직접 업데이트됩니다. 그러나 즉시 활성화되지는 않습니다. 트래픽을 처리하는 경우, 디바이스에서 새로운 설정을 사용하기 전에 컨피그레이션 변경 사항을 구축해야 합니다.

프로시저

단계 **1** Deployment 그룹의 POST /operational/deploy 리소스를 사용하여 구축을 시작합니다.

예를 들어 **curl** 명령은 다음과 같이 표시될 수 있습니다.

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

단계 **2** 응답을 평가하여 구축 작업이 대기되었는지 확인합니다.

양호한 응답(상태 코드 200)은 다음과 같이 표시됩니다. state(상태)를 참고합니다.

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

컨피그레이션 변경 사항 구축

```
  }
}
```

참고 **cliErrorMessage** 및 **name(**이름**)** 특성은 API v2에 추가되었으며 v1 응답에는 포함되지 않습니다.

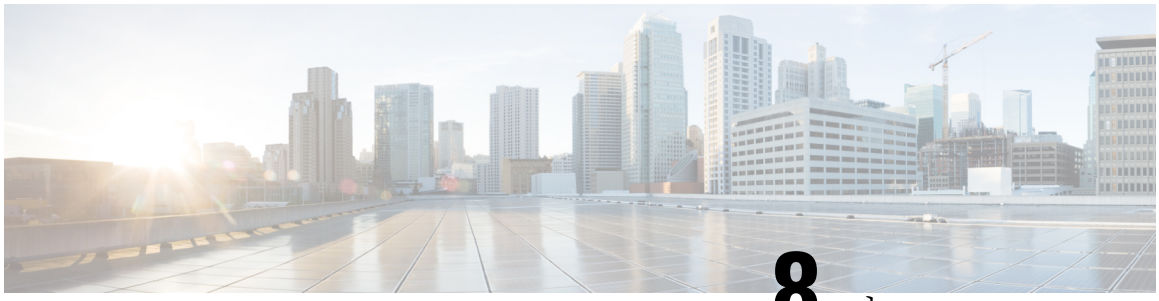
단계 **3** GET /operational/deploy/{objId} 리소스를 사용하여 작업의 상태를 확인합니다.

예를 들어 **curl** 명령은 다음과 같이 표시될 수 있습니다.

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

응답은 다음과 같이 표시될 수 있습니다. state(상태)는 DEPLOYED(구축됨)이며, 이는 작업이 성공적으로 완료되었음을 나타냅니다. ModifiedObjects 파라미터는 구축 작업에서 변경된 개체를 나열합니다. 이 경우, new-network라는 이름의 네트워크 개체에 한 가지 변경 사항이 있습니다.

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



8 장

구성 가져오기/내보내기

버전 요구 사항: 구성 가져오기/내보내기를 사용하려면 위협 방어 6.5(0) 이상 버전 및 위협 방어 REST API v4 이상 버전을 실행해야 합니다.

device manager로 관리되는 디바이스에서 구성을 내보내고 구성을 동일한 디바이스 또는 다른 호환 가능한 디바이스로 가져올 수 있습니다. 예를 들어 구성 가져오기/내보내기를 사용하여 여러 유사한 디바이스 전반에서 베이스라인 구성을 복제한 다음, 각 디바이스의 device manager을 사용하여 각 디바이스에 고유한 특성을 구성할 수 있습니다.

컨피그레이션 가져오기/내보내기 정보

device manager을 사용하거나 CDO를 통해 위협 방어 디바이스를 로컬로 관리하는 경우, 위협 방어 API를 사용하여 디바이스의 구성을 내보낼 수 있습니다. 이 메서드는 Secure Firewall Management Center에서 관리하는 디바이스에서 효과가 없습니다.

구성을 내보내면 zip 파일이 생성됩니다. 그러면 zip 파일을 워크스테이션에 다운로드할 수 있습니다. 이 구성 자체는 JSON 형식 텍스트 파일에서 특성-값 쌍을 사용하여 정의된 개체로 표시됩니다. 동일한 디바이스 또는 다른 디바이스로 파일을 다시 가져오기 전에 파일을 수정할 수 있습니다.

따라서 내보내기 파일을 사용하여 네트워크의 다른 디바이스에 구축 가능한 템플릿을 생성할 수 있습니다.

개체를 가져올 때, 구성 파일이 아닌 import 명령으로 개체를 직접 정의하는 옵션을 사용할 수도 있습니다. 그러나 적은 수의 변경 사항을 가져오는 경우에만 개체를 직접 정의해야 합니다.

다음 주제에서는 구성 가져오기/내보내기에 대해 자세히 설명합니다.

내보내기 파일에 포함된 내용

내보내기를 수행할 때 내보내기 파일에 포함할 구성을 지정합니다. 전체 내보내기의 경우, 내보내기 zip 파일에 모든 항목이 포함됩니다. 내보내기 위해 선택한 항목에 따라 내보내기 zip 파일에는 다음 항목이 포함될 수 있습니다.

- 구성된 각 개체를 정의하는 특성-값 쌍. 구성 가능한 모든 항목은 device manager에서 "개체"라고 부르는 항목이 아니라 개체로서 모델링됩니다.
- 원격 액세스 VPN을 구성한 경우, AnyConnect 패키지 및 기타 참조된 모든 파일(예: 클라이언트 프로필 XML 파일, DAP XML 파일, Hostscan 패키지)
- 맞춤형 파일 정책을 구성한 경우, 모든 참조된 정리 목록 또는 맞춤형 탐지 목록

가져오기/내보내기 및 백업/복원 비교

구성 가져오기/내보내기는 백업/복원과 동일하지 않습니다.

- 백업/복원은 재해 복구를 위한 작업입니다. 디바이스가 동일한 모델이며 백업이 수행된 디바이스와 동일한 소프트웨어 버전을 실행 중인 경우에만 디바이스에 백업을 복원할 수 있습니다. 이는 주로 동일한 디바이스에 대해 "마지막으로 양호했던" 구성을 복구하거나, 구성을 대체 디바이스에 복원하기 위한 것입니다.
- 가져오기/내보내기는 구성의 전체 또는 일부를 유지하기 위한 것입니다. 디바이스를 이미지로 다시 설치한 후에 내보내기 파일을 사용하여 구성을 디바이스에 복원할 수 있습니다. 또는 내보내기 파일을 템플릿으로 사용하여 다른 디바이스로 가져오기 전에 콘텐츠를 수정할 수 있습니다. 가져오기/내보내기를 사용하면 새 디바이스를 특정 베이스라인 구성으로 빠르게 가져올 수 있으므로 이를 네트워크에 더 빠르게 구축할 수 있습니다. 제한 이내에서 파일을 다른 디바이스 모델(예: Firepower 2120~2130)로 가져올 수도 있습니다. 가져오기 파일에 모든 디바이스 모델에서 지원되는 개체만 포함되는 경우, 가져오기에 대한 제한 사항이 거의 없어야 합니다. 한 가지 제한 사항은 디바이스에서 내보내기 파일에 사용된 버전과 동일한 API 버전을 사용해야 한다는 것입니다.

가져오기/내보내기 전략

다음은 가져오기/내보내기를 사용할 수 있는 몇 가지 방법입니다.

- 새 디바이스의 템플릿을 생성합니다**.** 필요한 베이스라인에 맞게 모델 디바이스를 구성한 다음, 전체 구성을 내보냅니다. 그 후에는 해당 구성을 새 디바이스로 가져온 다음, device manager 또는위협 방어 API를 사용하여 필요한 수정 작업을 수행할 수 있습니다. 또한 가져오기 전에 템플릿을 수정하여 수정 작업을 수행할 수 있는데 예를 들어, 각 인터페이스의 IP 주소를 수정할 수 있습니다. 전체 내보내기에는 ManagementIP 개체(type=managementip)가 포함됩니다. 이때 대상 디바이스에서 관리 주소 및 게이트웨이를 이미 구성한 경우 새 디바이스에 대한 템플릿을 생성할 때 내보내기 파일에서 이 개체를 제거해야 하며, 그렇지 않으면 관리 주소 지정 정보를 덮어쓰게 됩니다.
- 한 디바이스의 구성 변경 사항을 유사한 디바이스에 구축합니다**.** 예를 들어, 디바이스 A의 구성을 수정할 때 몇 가지 새로운 네트워크 개체 및 액세스 제어 규칙을 생성합니다. 그런 다음 보류 중인 변경 사항을 내보내고 이러한 변경 사항을 디바이스 B로 가져올 수 있습니다. 두 디바이스에서 구성을 구축하면 디바이스에서 동일한 새 규칙을 실행하게 됩니다.

- 시스템을 이미지로 다시 설치한 후에 구성을 다시 적용합니다**.** 디바이스를 이미지로 다시 설치하면 구성이 지워집니다. 전체 구성을 처음 내보내는 경우에는 이미지로 다시 설치하기를 완료한 후에 이를 가져올 수 있습니다.
- 대상 구성을 적용합니다**.** 내보내기 파일을 수정하거나 수동으로 생성할 수도 있으므로 다른 디바이스로 가져오려는 개체를 제외하고 모든 개체를 제거할 수 있습니다. 예를 들어 네트워크 개체 집합을 포함하는 구성 파일을 생성하고 이 파일을 사용하여 동일한 네트워크 개체 그룹을 모든 위협 방어 디바이스에 가져올 수 있습니다.

구성 가져오기/내보내기에 대한 지침

- 내보내기 작업 중에는 시스템에서 구성 데이터베이스에 대한 쓰기 잠금을 유지합니다. 작업을 완료할 때까지 API 또는 device manager을 사용하여 구성을 변경할 수 없습니다. 그러나 내보내기 작업 중에 device manager의 구성을 보거나 API에서 GET 호출을 사용할 수는 있습니다.
- 가져오기 작업 중에는 시스템에서 구성 데이터베이스에 대한 읽기 및 쓰기 잠금을 둘 다 유지합니다. 작업을 완료할 때까지 API 또는 device manager을 사용하여 구성을 보거나 변경할 수 없습니다.
- 가져온 구성이 기존 구성에 추가됩니다. 디바이스의 구성을 지우고 가져온 구성으로 교체할 수 없습니다. 가져오기 전에 디바이스 구성을 재설정해야 하는 경우, 디바이스 CLI로 이동하여 **configure manager delete** 명령을 실행한 다음, **configure manager local** 명령을 실행할 수 있습니다. 관리 인터페이스 구성만 유지됩니다.
- 디바이스에서 파일에 포함된 메타데이터 개체의 apiVersion 특성에 정의된 것과 동일한 API 버전을 실행 중인 경우에만 파일을 디바이스로 가져올 수 있습니다.

컨피그레이션 가져오기 및 내보내기

가져오기/내보내기 프로세스는 로컬의 매니지드 디바이스에서 구성을 내보내는 작업부터 시작됩니다. 그런 다음 내보내기 파일을 다운로드하고 필요에 따라 수정하여 동일한 디바이스 또는 호환 가능한 디바이스에 업로드하기 전에 해당 파일을 수정할 수 있습니다. 다음 주제에서는 각 단계에 대해 설명합니다.

구성 내보내기

POST /action/configexport 메서드를 사용하여 구성 내보내기 작업을 생성하고 시작합니다.

프로시저

단계 **1** 내보내기 작업용 JSON 개체 본문을 생성합니다.

이 호출에 사용할 JSON 개체의 예는 다음과 같습니다.

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

특성은 다음과 같습니다.

- **diskFileName** - (선택 사항) 내보내기 zip 파일의 이름입니다. 이름을 지정하지 않으면 시스템에서 대신 이름을 생성합니다. 이름을 지정하는 경우에도 시스템에서 고유성을 보장하기 위해 이름에 문자를 추가할 수 있습니다. 이름의 최대 길이는 60자입니다.
- **encryptionKey** - (선택 사항) zip 파일의 암호화 키입니다. 파일을 암호화하지 않으려면 이 필드를 생략하고 대신 "doNotEncrypt": true를 지정합니다. 키를 지정할 경우 zip 파일을 워크스테이션에 다운로드한 후에 해당 키를 사용하여 열어야 합니다. 내보낸 구성 파일에서는 비밀 키, 비밀번호 및 기타 중요한 데이터를 일반 텍스트에 표시합니다(다른 방법으로 가져올 수 없기 때문). 따라서 중요한 데이터를 보호하기 위해 암호화 키를 적용할 수 있습니다. 시스템에서 AES 256 암호화를 사용합니다.
- **doNotEncrypt** - (선택 사항) 내보내기 파일을 암호화할지(false), 암호화하지 않을지(true) 여부를 지정합니다. 기본값은 false입니다. 즉, 비어 있지 않은 encryptionKey 특성을 지정해야 합니다. true를 지정하면 encryptionKey 특성이 무시됩니다.
- **configExportType** - 다음 열거 값 중 하나입니다.
  - **FULL_EXPORT** - 전체 구성을 내보내기 파일에 포함합니다. 이는 기본값입니다.
  - **PARTIAL_EXPORT** - entityId 목록에서 식별된 개체 및 하위 개체만 포함합니다. 내보낼 수 없는 개체는 ID를 지정하는 경우에도 포함되지 않습니다. 모든 사용자 정의 개체는 내보낼 수 있습니다.
  - **PENDING_CHANGE_EXPORT** - 아직 구축되지 않은 개체 즉, 보류 중인 변경 사항만 포함합니다.
- **deployedObjectsOnly** - (선택 사항) 구축된 경우에만 내보내기 파일에 개체를 포함할지 여부입니다. 즉, 보류 중인 변경 사항은 포함하지 마십시오. 이 특성은 PENDING_CHANGE_EXPORT 작업의 경우 무시되는데 이러한 작업에는 구축되지 않은 개체만 포함되어 있기 때문입니다. 기본값은 false입니다. 이는 모든 보류 중인 변경 사항이 내보내기에 포함되었음을 의미합니다. 보류 중인 변경 사항을 제외하려면 true를 지정합니다.
- **entityIds** - 시작점 개체 집합의 ID가 쉼표로 구분된 목록으로,[대괄호]로 묶여 있습니다. 이 목록은 PARTIAL_EXPORT 작업에 필요합니다. 이 목록의 각 항목은 UUID 값 또는 "**id=***uuid-value*", "**type=***object-type*" 또는 "**name=***object-name*"과 같은 특성-값 쌍이 일치하는 패턴일 수 있습니다. 예를 들어 **"type=networkobject"**입니다.

type은 리프 엔터티(예: networkobject)이거나 리프 유형 집합의 별칭일 수 있습니다. 몇 가지 일반적인 유형의 별칭으로는 네트워크(NetworkObject 및 NetworkObjectGroup), 포트(모든 TCP/UDP/ICMP 포트, 프로토콜 및 그룹 유형), url(URL 개체 및 그룹), ikepolicy(IKE V1/V2 정책), ikeproposal(Ike V1/V2 제안), identitysource(모든 id 소스), 인증서(모든 인증서 유형), 개체(device manager에서 개체 페이지에 나열되는 모든 개체/그룹 유형), 인터페이스(모든 네트워크 인터페이스, s2svpn(모든 Site-to-Site VPN 관련 유형), ravpn(모든 RA VPN 관련 유형), vpn(s2svpn 및 ravpn 둘 다)이 있습니다.

이러한 모든 개체와 해당하는 발신 참조 하위 항목은 PARTIAL_EXPORT 출력 파일에 포함됩니다. 내보낼 수 없는 모든 개체는 ID를 지정한 경우에도 출력에서 제외됩니다. 적절한 리소스 유형에 대해 GET 메서드를 사용하여 대상 개체의 UUID, 유형 또는 이름을 가져옵니다.

예를 들어, 모든 네트워크 개체와 함께 myaccessrule이라는 액세스 규칙 및 UUID로 식별되는 개체 2개를 내보내려면 다음과 같이 지정하면 됩니다.

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

- **jobName** - (선택 사항) 내보내기 작업의 이름입니다. 작업의 이름을 지정하면 작업 상태를 검색할 때 더 쉽게 찾을 수 있습니다.
- **type(**유형**)** - 작업 유형이며 항상 **scheduleconfigexport**입니다.

예제**:**

다음 예에서는 파일 export-config-1에 대한 전체 내보내기를 수행하고 기타 모든 특성에 대한 기본값을 수락합니다.

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

단계 **2** 개체를 게시합니다.

예를 들어 curl 명령은 다음과 같이 표시됩니다.

```
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/최신/action/configexport'
```

단계 **3** 응답을 확인합니다.

응답 코드 200이 표시되어야 합니다. 최소 JSON 개체를 게시한 경우, 성공적인 응답 본문은 다음과 같이 표시됩니다. 암호화 키를 지정하는 경우 응답에서 해당 키가 마스킹됩니다.

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


내보내기 작업의 상태 확인

내보내기 작업을 완료하는 데는 시간이 약간 걸립니다. 구성할 항목이 많을수록 작업을 수행하는 데 더 많은 시간이 필요합니다. 작업 상태를 확인하여 파일 다운로드를 시도하기 전에 작업이 완료되었는지 확인합니다.

상태를 가져오는 가장 간단한 방법은 GET /jobs/configexportstatus를 사용하는 것입니다. 예를 들어 curl 명령은 다음과 같이 표시됩니다.

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

완료된 작업의 경우 다음과 유사한 상태가 반환됩니다.


```
{
  "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/최신
```

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

또는 GET /jobs/configexportstatus/{objId} 메서드를 사용하여 특정 작업에 대한 상태를 검색할 수 있습니다. 응답 개체의 **id** 필드에서 개체 ID를 가져옵니다.

내보내기 파일 다운로드

내보내기 작업이 완료되면 내보내기 파일이 시스템 디스크에 기록되며, 이 파일을 구성 파일이라고 합니다. GET /action/downloadconfigfile/{objId} 메서드를 사용하여 이 내보내기 파일을 워크스테이션에 다운로드할 수 있습니다. 사용 가능한 파일 목록을 가져오려면 GET /action/configfiles 메서드를 사용합니다.

참고 GET /action/downloadconfigfile/{objId}을 사용하여 일반적으로 파일 이름을 개체 ID로 지정합니다. 또는 파일과 관련된 ConfigExportStatus 개체의 ID를 지정할 수 있습니다.

프로시저

단계 **1** 디스크에 있는 구성 파일의 목록을 가져옵니다.

구성 파일의 목록에는 내보내기 파일 및 가져오기 위해 업로드한 모든 파일이 포함됩니다.

curl 명령은 다음과 같습니다.

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

응답에는 항목 목록이 표시되며 각 항목은 구성 파일입니다. 예를 들어, 다음 목록에는 2개의 파일이 표시됩니다. 모든 파일의 **id**는 기본값입니다. ID를 무시하고 **diskFileName**을 대신 사용합니다.

```
{
  "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/최신/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/최신/action/configfiles/default"
```

```
      }
    }
  ],
```

단계 **2** diskFileName을 개체 ID로 사용하여 파일을 다운로드합니다.

curl 명령은 다음과 같습니다.

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

파일은 기본 다운로드 폴더에 다운로드됩니다. API Explorer에서 GET 메서드를 실행하고 브라우저가 다운로드 위치를 확인하도록 구성된 경우 파일을 저장하라는 메시지가 표시됩니다.

다운로드에 성공하면 반환 코드 200이 생성되며 응답 본문이 없습니다.

내보낸 구성 파일 수정

구성 파일을 다운로드한 후에는 해당 파일의 압축을 풀어 개체가 포함된 텍스트 파일을 열 수 있습니다. WordPad에서는 NotePad보다 읽기 쉬운 방식으로 콘텐츠를 포맷합니다. 설치되어 있는 다른 텍스트 편집기를 사용할 수도 있습니다. 고유한 구성 파일을 처음부터 생성할 수도 있지만, 파일 구조를 파악하려면 구성을 내보내야 합니다.

다음 주제에서는 텍스트 파일의 요구 사항에 대해 설명합니다.

최소 구성 파일 요구 사항

구성 파일에는 다음과 같은 최소 요소가 필요합니다.

- 파일의 개체를 [대괄호]로 묶습니다. 전체 파일은 표준 JSON 표시법을 사용하며 개체를 배열한 것입니다.
- 각 개체를 {중괄호}로 묶습니다.
- 쉼표를 사용하여 구성 파일의 개체를 구분합니다. 즉, 개체의 끝 중괄호 뒤에는 마지막 개체를 제외하고 쉼표가 와야 합니다.
- 파일의 첫 번째 개체는 메타데이터 개체여야 합니다. 올바른 개체 특성을 가져오는 가장 쉬운 방법은 원하는 모델의 디바이스에서 구성을 내보내는 것입니다. 예를 들어, 다음은 Secure Firewall Threat Defense Virtual 디바이스의 메타데이터 개체입니다. 디바이스를 가져오기 전에 구성 및 내보내기 유형을 수정할 수 있으며, 원하는 경우 generatedOn 특성을 삭제할 수 있습니다.

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

- 메타데이터 개체에서는 적절한 구성 유형(configType) 값을 지정해야 합니다.

- FULL_CONFIG - 이 텍스트 파일에는 전체 디바이스 구성이 포함됩니다.
- DELTA_CONFIG - 이 텍스트 파일에는 일부 개체만 포함하는 부분 구성이 포함됩니다.

- exportType은 FULL_EXPORT, PARTIAL_EXPORT, PENDING_CHANGE_EXPORT 중 하나입니다.
- 전체 구성 가져오기를 수행 중인 경우, 메타데이터 개체에서는 hardwareModel, softwareVersion, apiVersion의 특성을 지정해야 합니다.
- 개체를 하나 또는 여러 줄에 작성할 수 있지만, 개체의 특성 사이에는 빈 줄 또는 주석 줄을 넣지 마십시오. 파일에서는 주석이 허용되지 않습니다.
- 개체는 다른 개체에서 참조하는 개체가 먼저 정의되는 종속성 순서로 내보내기되지만, 가져오기 구성 파일에서 해당 순서를 유지할 필요는 없습니다. 시스템에서는 개체 이름 및 ID가 종속 개체 간에 올바르게 확인되는 것으로 간주하여 가져오기 중에 관계를 자동으로 확인합니다.

ID 래퍼 개체의 기본 구조

구성 파일에서는 ID 래퍼 개체를 사용하여 내보내거나 가져올 수 있는 ConfigEntity 또는 ManagementEntity 개체를 정의합니다. ID 래퍼 개체의 기본 구조는 다음과 같습니다.

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

개체에는 다음 특성이 포함됩니다.

- **type** - 항상 **identitywrapper**입니다.
- **data** - 네트워크 개체, 액세스 제어 규칙 등의 구성에서 개체를 정의하는 특성-값 쌍의 모음입니다. 이 모음에 필요한 특성은 특정 개체 유형과 수행 중인 작업을 위한 모델에 따라 달라집니다. 특성-값 쌍을 {중괄호}로 묶습니다. 데이터 배열 내의 특성을 쉼표로 구분합니다.
- **parentName** - (필요시) 제한된 수의 개체는 ContainedObjects이며 이를 포함하는 개체와 관계가 있습니다. 예로는 액세스 규칙, 수동 NAT 규칙, 하위 인터페이스가 있습니다. 이러한 항목의 경우 parentName에서은 포함하는 개체(상위 항목)의 이름을 지정합니다. 포함된 개체에 이 특성을 지정하고, 포함되지 않은 개체에는 이 특성을 지정하지 마십시오. 이러한 개체에 대한 인덱스를 지정해야 할 수도 있습니다.

  상위 항목이 단일 개체(즉, 둘 이상 생성할 수 없음, 예: AccessPolicy)이며 시스템에서 참조를 확인할 수 있는 경우 실제로 이 특성을 생략할 수 있습니다.
- **oldName** - (필요시) 기존 개체의 이름을 바꾸는 경우, 이 특성의 이전 이름 및 데이터 특성의 **name** 특성에 새 이름을 지정할 수 있습니다. 이 특성을 사용하려면 action이 EDIT여야 합니다.

- **action** - 정의된 개체와 관련하여 수행할 작업입니다. 전체 내보내기에서는 작업이 항상 **CREATE** 입니다. 보류 중인 변경 또는 부분 내보내기의 경우 다른 작업은 **EDIT** 또는 **DELETE**일 수 있습니다.

  가져오기를 위해 파일을 수정할 때 원하는 작업을 지정합니다. CREATE를 지정했지만 개체가 이미 존재하는 경우, 작업은 EDIT로 변경됩니다. 개체가 없는 경우, EDIT가 CREATE로 변경됩니다. DELETE 작업은 변경되지 않습니다. 개체 참조는 개체 유형 및 이름, 개체 유형 및 이전 이름, 또는 개체 유형 및 상위 이름에 따라 확인됩니다.

  - CREATE - 이는 새 개체입니다. 개체를 게시할 때 필요한 데이터 특성을 지정해야 합니다. **name**이 지정된 유형의 기존 개체와 일치하는 경우, 작업이 EDIT로 자동으로 변경됩니다.

    새 개체를 생성하고 다른 개체에서 해당 개체를 참조하는 경우(예: 네트워크 개체를 정의한 다음, 액세스 규칙에서 이를 사용하는 경우)에는 참조에서 개체 **name**이 정확해야 합니다.

  - EDIT - 개체를 업데이트하는 중입니다. version 및 id를 제외하고 개체를 배치할 때 필요한 데이터 특성을 지정해야 합니다. 이름 및 개체 유형은 업데이트할 개체를 결정하는 데 사용되며 버전 특성은 항상 무시됩니다.

  - DELETE - 개체를 삭제하는 중입니다. 개체 데이터에서 **type** 및 **name** 특성을 지정해야 합니다.

- **index** - (선택 사항, 정수) 액세스 제어 및 수동 NAT 규칙과 같이 순서가 지정된 목록의 일부인 개체의 경우, 정책에서 개체의 위치입니다. 새 규칙을 생성하고 인덱스 값을 지정하지 않은 경우, 규칙은 마지막 규칙으로 정책 끝에 추가됩니다. 규칙을 수정하는 중인 경우, 규칙의 기존 위치가 유지됩니다.

예: 다른 디바이스로 가져오기 위해 네트워크 개체 수정

각 개체는 다음과 같이 구성되며, 이는 syslog 서버의 IP 주소를 정의하는 네트워크 호스트 개체입니다.

```
{"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"}}
```

디바이스에서 이 개체를 내보냈으며 개체를 다른 디바이스로 가져오고 싶지만 새 디바이스가 다른 주소(192.168.5.15)에서 syslog 서버를 사용해야 한다고 가정해 보겠습니다. 새 개체를 생성할 예정이므로 데이터 특성에서 **version** 및 **id** 특성을 제거합니다. **isSystemDefined**(기본값이 false임) 및 **dnsResolution**(FQDN 개체만 해당)도 제거할 수 있습니다. 그 결과 생성되는 새 개체는 다음과 같이 표시됩니다.

```
{"type":"identitywrapper",
 "action":"CREATE",
```

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

파일 상단에서 메타데이터 개체를 유지(또는 추가)해야 합니다. 파일 콘텐츠를 더 쉽게 스캔하고 확인할 수 있도록 줄 반환을 추가할 수도 있습니다. 따라서 전체 구성 파일은 다음과 같이 표시됩니다.

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

가져오기 파일 업로드

구성 파일을 디바이스에 가져오려면 먼저 파일을 디바이스에 업로드해야 합니다. zip 또는 텍스트 파일 중 하나를 업로드할 수 있습니다. zip 파일을 사용하는 경우 AnyConnect 패키지 및 클라이언트 프로필을 포함할 수 있습니다.

POST /action/uploadconfigfile 리소스를 사용하여 파일을 업로드합니다. 이름의 최대 길이는 60자입니다.

- API Explorer에서 이 메서드를 사용하는 경우, **fileToUpload** 특성 옆에 있는 **Choose File(**파일 선택**)** 버튼을 클릭하여 워크스테이션 드라이브에서 파일을 선택합니다.
- 고유한 프로그램에서 이 메서드를 사용 중인 경우, 요청 페이로드는 파일 이름 필드와 함께 단일 파일 항목을 포함해야 합니다. 파일 이름 확장명은 .txt 또는 .zip이어야 하며, 실제 파일 콘텐츠 형식은 파일 확장명과 일치해야 합니다.

curl 명령은 다음과 같이 표시됩니다.

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

전송에 성공하면 반환 코드 200 및 다음과 유사한 응답 본문이 표시됩니다. 여기에는 위협 방어 시스템의 파일 이름(**diskFileName**)이 표시되며, 이는 가져오기 작업에 필요합니다.

```
{
  "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/최신/action/uploadconfigfile/default"
  }
}
```

구성 가져오기 및 작업 상태 확인

위협 방어 시스템에 구성 파일을 업로드한 후에는 구성 파일에 정의된 개체를 위협 방어 구성에 가져올 수 있습니다. POST /action/configimport 메서드를 사용합니다.

개체를 가져올 때, 구성 파일이 아닌 import 명령으로 개체를 직접 정의하는 옵션을 사용할 수도 있습니다. 그러나 한두 개의 네트워크 개체 등 적은 수의 변경 사항을 가져오는 경우에만 개체를 직접 정의해야 합니다.

프로시저

단계 **1** 가져오기 작업용 JSON 개체 본문을 생성합니다.

이 호출에 사용할 JSON 개체의 예는 다음과 같습니다.

```
{
  "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"
}
```

특성은 다음과 같습니다.

- **diskFileName** - 가져올 구성 zip 또는 txt 파일의 이름입니다.
- **encryptionKey** - zip 파일을 암호화하는 데 사용된 키입니다(있는 경우). 구성 파일이 암호화되지 않은 경우 키를 지정하지 마십시오.

- **preserveConfigFile** - (선택 사항) 가져오기 작업을 성공적으로 완료한 후에 가져온 구성 파일의 복사본을 위협 방어 디스크에 보관할지 여부입니다. 파일을 보관하려면 true를 지정하고 위협 방어 디스크에서 파일을 삭제하려면 false를 지정합니다. 기본값은 false입니다.
- **autoDeploy** - (선택 사항) 가져오기에 성공한 경우 구축 작업을 자동으로 시작할지 여부입니다. 가져온 개체는 보류 중인 변경 사항에 해당하며 변경 사항을 성공적으로 구축할 때까지 활성화되지 않습니다. 구축 작업을 자동으로 시작하려면 true를 지정합니다. false를 지정하는 경우 변경 사항을 수동으로 구축해야 합니다. 기본값은 false입니다.
- **allowPendingChange** - (선택 사항) 기존의 보류 중인 변경 사항이 있는 경우 가져오기 작업을 시작할지 여부입니다. 이 특성을 true로 설정하고 autoDeploy를 true로 설정하면 자동 구축 작업에 기존 및 가져온 변경 사항 등 모든 변경 사항이 포함됩니다. 이 특성을 false로 설정하면 보류 중인 변경 사항이 있는 경우 가져오기 작업이 실행되지 않습니다. 기본값은 false입니다.
- **excludeEntities** - (선택 사항) 가져오지 않아야 하는 개체를 식별하는 개체 일치 문자열의 목록입니다. 가져오기 파일에 가져오지 않을 항목이 포함된 경우에만(즉, 업로드한 파일에서 삭제하지 않기로 결정한 경우) 이 특성을 지정해야 합니다. 이 목록의 각 항목은 "**id=***uuid-value*", "**type=***object-type*" 또는 "**name=***object-name*"rhk 같은 패턴을 지닙니다. 이러한 패턴 중 하나와 일치하는 입력 개체는 가져오기에서 제외됩니다.

  type은 리프 엔터티(예: networkobject)이거나 리프 유형 집합의 별칭일 수 있습니다. 몇 가지 일반적인 유형의 별칭으로는 네트워크(NetworkObject 및 NetworkObjectGroup), 포트(모든 TCP/UDP/ICMP 포트, 프로토콜 및 그룹 유형), url(URL 개체 및 그룹), ikepolicy(IKE V1/V2 정책), ikeproposal(Ike V1/V2 제안), identitysource(모든 id 소스), 인증서(모든 인증서 유형), 개체(device manager에서 개체 페이지에 나열되는 모든 개체/그룹 유형), 인터페이스(모든 네트워크 인터페이스, s2svpn(모든 Site-to-Site VPN 관련 유형), ravpn(모든 RA VPN 관련 유형), vpn(s2svpn 및 ravpn 둘 다)이 있습니다.

  예를 들어, 모든 네트워크 개체와 myobj 및 UUID라는 이름으로 식별된 두 개의 다른 개체를 가져오기에서 제외하려면 다음을 지정합니다.

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

- **inputEntities** - 가져올 개체 수가 적은 경우, 구성 파일이 아닌 inputEntities 개체 목록에서 정의할 수 있습니다. 이 특성을 사용하려면 diskFileName 특성을 포함할 수 없으며 이 특성을 null로 설정해야 합니다.
- **jobName** - (선택 사항) 내보내기 작업의 이름입니다. 작업의 이름을 지정하면 작업 상태를 검색할 때 더 쉽게 찾을 수 있습니다.
- **type** - 작업 유형이며 항상 **scheduleconfigimport**입니다.

예제**:**

다음 예에서는 import-1.txt라는 이름의 구성 파일을 가져옵니다.

```
{
  "diskFileName": "import-2.txt",
```

```
  "preserveConfigFile": true,
  "autoDeploy": true,
  "allowPendingChange": true,
  "type": "scheduleconfigimport"
}
```

단계 **2** 개체를 게시합니다.

예를 들어 curl 명령은 다음과 같이 표시됩니다.

```
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/최신/action/configimport'
```

단계 **3** 응답을 확인합니다.

응답 코드 200이 표시되어야 합니다. 최소 JSON 개체를 게시한 경우, 성공적인 응답 본문은 다음과 같이 표시됩니다. 암호화 키를 지정하는 경우 응답에서 해당 키가 마스킹됩니다.

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

단계 **4** GET /jobs/configimportstatus를 사용하여 가져오기 작업의 상태를 확인합니다.

또는 GET /jobs/configimportstatus/{objId}를 사용하여 한 개의 가져오기 작업의 상태를 가져올 수 있습니다. objId의 경우 응답 본문의 jobHistoryUuid 값을 POST /action/configimport 호출에 사용합니다.

curl 명령은 다음과 같이 표시됩니다.

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

성공적인 가져오기에 대한 응답 본문은 다음과 같이 표시될 수 있습니다. 가져오기에 실패하면 형식 지정 또는 콘텐츠 오류를 수정하기 위해 파일을 수정하고 다시 시도해야 할 수 있습니다.

```
{
```

```
      "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/최신
/jobs/configimportstatus/7e360139-6725-11e9-abb5-078014531401"
  }
}
```


다음에 수행할 작업

autoDeploy를 false로 설정한 경우, 가져온 변경 사항을 통합하려면 구축 작업을 실행해야 합니다. POST /operational/deploy 메서드를 사용합니다. 이 메서드를 true로 설정한 경우 구성이 성공적으로 구축되어야 합니다. device manager 또는 API(GET /operational/auditevents)에서 감사 로그를 확인할 수 있으며, 구축 작업의 이름은 "구성 후 가져오기 구축"으로 지정되어 있습니다.

참고 일부 기능의 경우 특정 라이선스가 필요합니다. 예를 들어, 디바이스에는 원격 액세스 VPN 기능에 대한 라이선스가 있어야 합니다. 그러나 가져오기 프로세스에서는 라이선스를 검증하지 않습니다. 따라서 라이선스 제어 기능에 대한 개체를 필수 라이선스가 없는 디바이스로 가져오는 경우, 구축 작업에 실패합니다. 이 문제가 발생하면 디바이스에 필수 라이선스를 할당하거나 개체를 삭제합니다.

필요 없는 가져오기/내보내기 파일 삭제

구성 파일이 더 이상 필요하지 않은 경우, 내보내기 작업을 통해 생성된 파일 또는 구성 가져오기를 위해 업로드한 파일을 삭제할 수 있습니다.

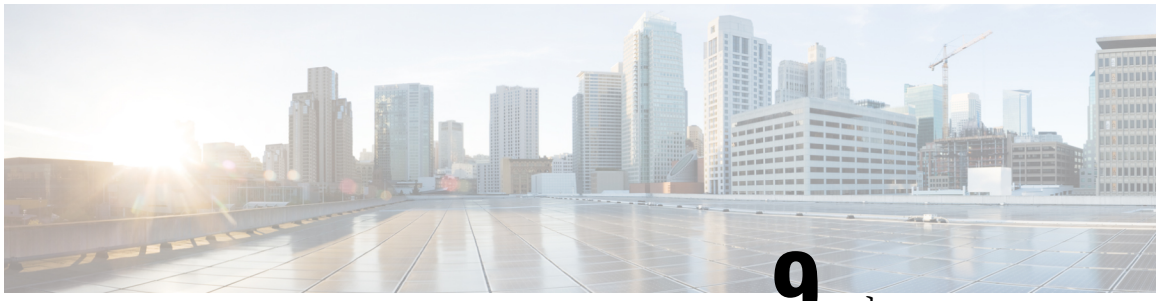
파일 이름을 objId 값으로 사용하는 DELETE /action/configfiles/{objId} 메서드를 사용합니다.

예를 들어, export-config-2.zip이라는 이름의 파일을 삭제하려는 경우 curl 명령은 다음과 같습니다.

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

성공적인 결과는 응답 본문이 없는 204 반환 코드입니다.

GET /action/configfiles를 사용하여 파일이 삭제되었음을 확인할 수 있습니다.



9 장

추가 정보 및 예시

추가 정보 및 예시

다음 사이트에서 API를 사용하는 방법에 대한 추가 정보를 참조할 수 있습니다.

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

  이 사이트에는 Bash 호출 및 Python 코드에 대한 예시를 포함하여 리소스에 대한 참조 정보가 포함되어 있습니다. 여기에는 사용하려는 API 버전을 선택할 수 있는 메뉴가 있습니다. 올바른 참조 정보를 확인하려면 적절한 버전을 선택하십시오. 이 사이트에는 API를 사용할 때 표시될 수 있는 모든 고유 오류 코드 및 메시지의 목록도 있습니다.

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

  이 사이트에는 코드 샘플을 비롯하여, 고가용성 같은 선택 기능을 구성하기 위한 엔드 투 엔드 예시가 있습니다.

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

  이 사이트에는 API 사용 방법을 익히는 데 도움이 되는 비디오, 학습 모듈, 실습이 포함되어 있습니다.

추가 정보 및 예시