Active IQ Unified Manager의 REST API 액세스 및 인증

Active IQ Unified Manager

NetApp
November 13, 2026

This PDF was generated from https://docs.netapp.com/ko-kr/active-iq-unified-manager-918/api-automation/concept_rest_api_access_and_authentication_in_um_apis.html on November 13, 2026. Always check docs.netapp.com for the latest.

목차

Active IQ Unified Manager의 REST API 액세스 및 인증 . . . 1
- Active IQ Unified Manager에서 REST API 액세스 및 인증 . . . 1
  - 인증 . . . 3
- Active IQ Unified Manager HTTP 상태 코드에 대해 알아보기 . . . 3
- Active IQ Unified Manager API 사용에 대한 권장 사례를 따르십시오. . . . 4
- Active IQ Unified Manager에서 시스템 로그로 문제 해결 . . . 5
- Active IQ Unified Manager가 비동기 처리를 위해 작업 오브젝트를 사용하는 방식 이해 . . . 5
  - Job 객체를 사용하여 설명된 비동기 요청 . . . 5
  - API 요청과 연결된 Job 객체를 쿼리합니다 . . . 6
  - 비동기 요청의 단계 . . . 6
- Active IQ Unified Manager의 Hello API 서버 샘플 프로그램에 대해 알아보기 . . . 6

Active IQ Unified Manager의 REST API 액세스 및 인증

Active IQ Unified Manager에서 REST API 액세스 및 인증

Active IQ Unified Manager REST API는 기본 HTTP 인증 메커니즘을 사용하여 HTTP 요청을 보낼 수 있는 모든 REST 클라이언트 또는 프로그래밍 플랫폼을 통해 액세스할 수 있습니다.

요청 및 응답 예시:

- 요청

```
GET
https://<IP
address/hostname>:<port_number>/api/v2/datacenter/cluster/clusters
```

- 응답

```
{
  "records": [
    {
      "key": "4c6bf721-2e3f-11e9-a3e2-
00a0985badbb:type=cluster,uuid=4c6bf721-2e3f-11e9-a3e2-00a0985badbb",
      "name": "fas8040-206-21",
      "uuid": "4c6bf721-2e3f-11e9-a3e2-00a0985badbb",
      "contact": null,
      "location": null,
      "version": {
        "full": "NetApp Release Dayblazer__9.5.0: Thu Jan 17 10:28:33
UTC 2019",
        "generation": 9,
        "major": 5,
        "minor": 0
      },
      "isSanOptimized": false,
      "management_ip": "10.226.207.25",
      "nodes": [
        {
          "key": "4c6bf721-2e3f-11e9-a3e2-
00a0985badbb:type=cluster_node,uuid=12cf06cc-2e3a-11e9-b9b4-
00a0985badbb",
          "uuid": "12cf06cc-2e3a-11e9-b9b4-00a0985badbb",
          "name": "fas8040-206-21-01",
          "_links": {
```

```
            "self": {
              "href": "/api/datacenter/cluster/nodes/4c6bf721-2e3f-11e9-
a3e2-00a0985badbb:type=cluster_node,uuid=12cf06cc-2e3a-11e9-b9b4-
00a0985badbb"
            }
          },
          "location": null,
          "version": {
            "full": "NetApp Release Dayblazer__9.5.0: Thu Jan 17
10:28:33 UTC 2019",
            "generation": 9,
            "major": 5,
            "minor": 0
          },
          "model": "FAS8040",
          "uptime": 13924095,
          "serial_number": "701424000157"
        },
        {
          "key": "4c6bf721-2e3f-11e9-a3e2-
00a0985badbb:type=cluster_node,uuid=1ed606ed-2e3a-11e9-a270-
00a0985bb9b7",
          "uuid": "1ed606ed-2e3a-11e9-a270-00a0985bb9b7",
          "name": "fas8040-206-21-02",
          "_links": {
            "self": {
              "href": "/api/datacenter/cluster/nodes/4c6bf721-2e3f-11e9-
a3e2-00a0985badbb:type=cluster_node,uuid=1ed606ed-2e3a-11e9-a270-
00a0985bb9b7"
            }
          },
          "location": null,
          "version": {
            "full": "NetApp Release Dayblazer__9.5.0: Thu Jan 17
10:28:33 UTC 2019",
            "generation": 9,
            "major": 5,
            "minor": 0
          },
          "model": "FAS8040",
          "uptime": 14012386,
          "serial_number": "701424000564"
        }
      ],
      "_links": {
        "self": {
```

```
          "href": "/api/datacenter/cluster/clusters/4c6bf721-2e3f-11e9-
a3e2-00a0985badbb:type=cluster,uuid=4c6bf721-2e3f-11e9-a3e2-
00a0985badbb"
        }
      }
    },
```

◦ `*IP address/hostname*`은(는) API 서버의 IP 주소 또는 정규화된 도메인 이름(FQDN)입니다.

◦ 포트 443

443은 기본 HTTPS 포트입니다. 필요한 경우 HTTPS 포트를 사용자 지정할 수 있습니다.
웹 브라우저에서 HTTP 요청을 보내려면 REST API 브라우저 플러그인을 사용해야 합니다. 또한 cURL 및 Perl 등 스크립팅 플랫폼을 사용하여 REST API에 액세스할 수도 있습니다.

인증

Unified Manager는 API에 대한 기본 HTTP 인증 방식을 지원합니다. 안전한 정보 흐름(요청 및 응답)을 위해 REST API는 HTTPS를 통해서만 접근할 수 있습니다. API 서버는 모든 클라이언트에게 서버 인증을 위한 자체 서명 SSL 인증서를 제공합니다. 이 인증서는 사용자 지정 인증서(또는 CA 인증서)로 교체할 수 있습니다.

REST API를 호출하려면 API 서버에 대한 사용자 액세스를 구성해야 합니다. 사용자는 로컬 사용자(로컬 데이터베이스에 저장된 사용자 프로필) 또는 LDAP 사용자(LDAP를 통해 인증하도록 API 서버를 구성한 경우)일 수 있습니다. Unified Manager 관리 콘솔 사용자 인터페이스에 로그인하여 사용자 액세스를 관리할 수 있습니다.

Active IQ Unified Manager HTTP 상태 코드에 대해 알아보기

API를 실행하거나 문제를 해결하는 동안 Active IQ Unified Manager API에서 사용하는 다양한 HTTP 상태 코드 및 오류 코드를 숙지해야 합니다.

다음 표는 인증과 관련된 오류 코드 목록입니다:

HTTP 상태 코드	상태 코드 제목	설명
200	OK	동기 API 호출이 성공적으로 실행되면 반환됩니다.
201	생성됨	Active Directory 구성 등 동기 호출을 통한 새 리소스 생성.
202	수락됨	LUN 및 파일 공유 생성 등 프로비저닝 기능에 대한 비동기 호출이 성공적으로 실행되면 반환됩니다.
400	잘못된 요청입니다	입력 유효성 검사 실패를 나타냅니다. 사용자는 요청 본문의 유효한 키 등 입력값을 수정해야 합니다.
401	무단 요청	해당 리소스를 볼 권한이 없습니다 /권한 없음.
403	금지된 요청	액세스하려는 리소스에 대한 액세스가 금지되었습니다.

HTTP 상태 코드	상태 코드 제목	설명
404	리소스를 찾을 수 없습니다	찾으시려던 리소스를 찾을 수 없습니다.
405	허용되지 않는 메서드	해당 메서드는 허용되지 않습니다.
429	요청이 너무 많습니다	사용자가 특정 시간 내에 너무 많은 요청을 보낼 때 반환됩니다.
500	내부 서버 오류	내부 서버 오류입니다. 서버에서 응답을 받지 못했습니다. 이 내부 서버 오류는 영구적일 수도 있고 그렇지 않을 수도 있습니다. 예를 들어, `GET` 또는 `GET ALL` 작업을 실행하고 이 오류가 발생하는 경우 최소 5회 이상 이 작업을 반복하는 것이 좋습니다. 영구적인 오류인 경우 반환되는 상태 코드는 계속 500입니다. 작업이 성공하면 반환되는 상태 코드는 200입니다.

Active IQ Unified Manager API 사용에 대한 권장 사례를 따르십시오.

Active IQ Unified Manager에서 API를 사용할 때 특정 권장 사항을 따라야 합니다.

- 유효한 실행을 위해서는 모든 응답 콘텐츠 유형이 다음 형식을 따라야 합니다:

```
application/json
```

- API 버전 번호는 제품 버전 번호와 관련이 없습니다. Unified Manager 인스턴스에서 사용 가능한 최신 API 버전을 사용해야 합니다. Unified Manager API 버전에 대한 자세한 내용은 “REST API versioning in Active IQ Unified Manager” 섹션을 참조하십시오.
- Unified Manager API를 사용하여 배열 값을 업데이트할 때는 전체 값 문자열을 업데이트해야 합니다. 배열에 값을 추가할 수는 없습니다. 기존 배열만 교체할 수 있습니다.
- 메트릭 API의 IOPS 및 성능과 같은 double 값을 제외한 모든 쿼리 매개변수에 파이프(|) 및 와일드카드(*) 등의 필터 연산자를 사용할 수 있습니다.
- 필터 연산자 와일드카드(*)와 파이프(|)를 조합하여 오브젝트를 쿼리하는 것은 피하십시오. 잘못된 수의 오브젝트가 검색될 수 있습니다.
- 필터에 값을 사용할 때는 값에 ? 문자가 포함되지 않도록 하십시오. 이는 SQL 인젝션 위험을 완화하기 위한 것입니다.
- 모든 API에 대한 `GET` (all) 요청은 최대 1000개의 레코드를 반환한다는 점에 유의하십시오. `max_records` 매개변수를 1000보다 높은 값으로 설정하여 쿼리를 실행하더라도 1000개의 레코드만 반환됩니다.
- 관리 기능을 수행하려면 Unified Manager UI를 사용하는 것이 좋습니다.

Active IQ Unified Manager에서 시스템 로그로 문제 해결

시스템 로그를 사용하면 API 실행 중에 발생할 수 있는 오류 원인을 분석하고 문제를 해결할 수 있습니다.

API 호출 관련 문제 해결을 위해 다음 위치에서 로그를 가져오십시오.

로그 위치	사용
`/var/log/ocie/access_log.log`	API를 호출하는 사용자의 이름, 시작 시간, 실행 시간, 상태 및 URL 등 모든 API 호출 세부 정보가 포함됩니다. 이 로그 파일을 사용하여 자주 사용하는 API를 확인하거나 GUI 워크플로의 문제를 해결할 수 있습니다. 또한 실행 시간에 따라 분석 규모를 확장하는 데에도 사용할 수 있습니다.
`/var/log/ocum/ocumserver.log`	모든 API 실행 로그가 포함되어 있습니다. 이 로그 파일을 사용하여 API 호출 문제를 해결하고 디버깅할 수 있습니다.
`/var/log/ocie/server.log`	Wildfly 서버 배포 및 서비스 시작/중지 관련 로그가 모두 포함되어 있습니다. 이 로그 파일을 사용하여 Wildfly 서버의 시작, 중지 또는 배포 중에 발생하는 문제의 근본 원인을 찾을 수 있습니다.
`/var/log/ocie/au.log`	획득 장치 관련 로그가 포함되어 있습니다. ONTAP에서 오브젝트를 생성, 수정 또는 삭제했지만 Active IQ Unified Manager REST API에 반영되지 않는 경우 이 로그 파일을 사용할 수 있습니다.

Active IQ Unified Manager가 비동기 처리를 위해 작업 오브젝트를 사용하는 방식 이해

Active IQ Unified Manager는 다른 API를 실행하는 동안 수행된 작업에 대한 정보를 검색하는 `jobs` API를 제공합니다. Job 오브젝트를 사용하여 비동기 처리가 어떻게 작동하는지 알아야 합니다.

일부 API 호출, 특히 리소스를 추가하거나 수정하는 데 사용되는 호출은 다른 호출보다 완료하는 데 시간이 더 오래 걸릴 수 있습니다. Unified Manager는 이러한 장시간 실행되는 요청을 비동기적으로 처리합니다.

Job 객체를 사용하여 설명된 비동기 요청

비동기적으로 실행되는 API 호출 후 HTTP 응답 코드 202는 요청이 성공적으로 검증되고 수락되었지만 아직 완료되지 않았음을 나타냅니다. 요청은 백그라운드 작업으로 처리되며, 클라이언트에 대한 초기 HTTP 응답 이후에도 계속 실행됩니다. 응답에는 요청을 연결하는 Job 객체와 해당 고유 식별자가 포함됩니다.

API 요청과 연결된 Job 객체를 쿼리합니다

HTTP 응답에 반환되는 Job 객체에는 여러 속성이 포함되어 있습니다. state 속성을 조회하여 요청이 성공적으로 완료되었는지 확인할 수 있습니다. Job 객체는 다음 상태 중 하나일 수 있습니다.

- `NORMAL`
- `WARNING`
- `PARTIAL_FAILURES`
- `ERROR`

Job 객체를 폴링하여 작업의 종료 상태(성공 또는 실패)를 감지하는 데 사용할 수 있는 두 가지 기술이 있습니다.

- 표준 폴링 요청: 현재 작업 상태가 즉시 반환됩니다.
- 장시간 폴링 요청: 작업 상태가 `NORMAL`, `ERROR`, 또는 `PARTIAL_FAILURES.`로 변경될 때

비동기 요청의 단계

다음과 같은 상위 수준 절차를 사용하여 비동기 API 호출을 완료할 수 있습니다.

1. 비동기 API 호출을 실행합니다.
2. 요청이 성공적으로 수락되었음을 나타내는 HTTP 응답 202를 수신합니다.
3. 응답 본문에서 Job 객체의 식별자를 추출합니다.
4. 반복문 내에서 Job 오브젝트가 종료 상태 `NORMAL`, `ERROR`, 또는 `PARTIAL_FAILURES.`에 도달할 때까지 기다립니다
5. 작업의 최종 상태를 확인하고 작업 결과를 검색합니다.

Active IQ Unified Manager의 Hello API 서버 샘플 프로그램에 대해 알아보기

_Hello API 서버_는 간단한 REST 클라이언트를 사용하여 Active IQ Unified Manager에서 REST API를 호출하는 방법을 보여주는 샘플 프로그램입니다. 이 샘플 프로그램은 API 서버에 대한 기본 정보를 JSON 형식(서버는 `application/json` 형식만 지원)으로 제공합니다.

사용된 URI는 다음과 같습니다. `https://<hostname>/api/datacenter/svm/svms`. 이 샘플 코드는 다음 입력 매개변수를 받습니다.

- API 서버 IP 주소 또는 FQDN
- 선택 사항: 포트 번호(기본값: 443)
- 사용자 이름
- 비밀번호
- 응답 형식 (`application/json`)

REST API를 호출하려면 Jersey 및 RESTEasy 등 다른 스크립트를 사용하여 Active IQ Unified Manager용 Java REST 클라이언트를 작성할 수도 있습니다. 샘플 코드와 관련하여 다음 사항을 고려해야 합니다:

- HTTPS 연결을 사용하여 Active IQ Unified Manager에 접속하고 지정된 REST URI를 호출합니다
- Active IQ Unified Manager에서 제공하는 인증서를 무시합니다
- 핸드셰이크 중 호스트 이름 확인을 건너뜁니다.
- URI 연결에 `javax.net.ssl.HttpsURLConnection` 사용
- HTTP 기본 인증에 사용되는 Base64 인코딩된 문자열을 구성하기 위해 타사 라이브러리 (`org.apache.commons.codec.binary.Base64`)를 사용합니다.

샘플 코드를 컴파일하고 실행하려면 Java 컴파일러 1.8 이상 버전을 사용해야 합니다.

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.URL;
import java.security.SecureRandom;
import java.security.cert.X509Certificate;
import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSession;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;
import org.apache.commons.codec.binary.Base64;

public class HelloApiServer {

    private static String server;
    private static String user;
    private static String password;
    private static String response_format = "json";
    private static String server_url;
    private static String port = null;

    /*
     * * The main method which takes user inputs and performs the *
necessary steps
     * to invoke the REST URI and show the response
     */ public static void main(String[] args) {
        if (args.length < 2 || args.length > 3) {
            printUsage();
            System.exit(1);
        }
        setUserArguments(args);
        String serverBaseUrl = "https://" + server;
        if (null != port) {
            serverBaseUrl = serverBaseUrl + ":" + port;
        }
```

```
        server_url = serverBaseUrl + "/api/datacenter/svm/svms";
        try {
            HttpsURLConnection connection =
getAllTrustingHttpsUrlConnection();
            if (connection == null) {
                System.err.println("FATAL: Failed to create HTTPS
connection to URL: " + server_url);
                System.exit(1);
            }
            System.out.println("Invoking API: " + server_url);
            connection.setRequestMethod("GET");
            connection.setRequestProperty("Accept", "application/" +
response_format);
            String authString = getAuthorizationString();
            connection.setRequestProperty("Authorization", "Basic " +
authString);
            if (connection.getResponseCode() != 200) {
                System.err.println("API Invocation Failed : HTTP error
code : " + connection.getResponseCode() + " : "
                        + connection.getResponseMessage());
                System.exit(1);
            }
            BufferedReader br = new BufferedReader(new
InputStreamReader((connection.getInputStream())));
            String response;
            System.out.println("Response:");
            while ((response = br.readLine()) != null) {
                System.out.println(response);
            }
            connection.disconnect();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /* Print the usage of this sample code */ private static void
printUsage() {
        System.out.println("\nUsage:\n\tHelloApiServer <hostname> <user>
<password>\n");
        System.out.println("\nExamples:\n\tHelloApiServer localhost admin
mypassword");
        System.out.println("\tHelloApiServer 10.22.12.34:8320 admin
password");
        System.out.println("\tHelloApiServer 10.22.12.34 admin password
");
        System.out.println("\tHelloApiServer 10.22.12.34:8212 admin
```

```
password \n");
        System.out.println("\nNote:\n\t(1) When port number is not
provided, 443 is chosen by default.");
    }

    /* * Set the server, port, username and password * based on user
inputs. */ private static void setUserArguments(
            String[] args) {
        server = args[0];
        user = args[1];
        password = args[2];
        if (server.contains(":")) {
            String[] parts = server.split(":");
            server = parts[0];
            port = parts[1];
        }
    }

    /*
     * * Create a trust manager which accepts all certificates and * use
this trust
     * manager to initialize the SSL Context. * Create a
HttpsURLConnection for this
     * SSL Context and skip * server hostname verification during SSL
handshake. * *
     * Note: Trusting all certificates or skipping hostname verification *
is not
     * required for API Services to work. These are done here to * keep
this sample
     * REST Client code as simple as possible.
     */ private static HttpsURLConnection
getAllTrustingHttpsUrlConnection()    {        HttpsURLConnection conn =
null;        try {            /* Creating a trust manager that does not
validate certificate chains */            TrustManager[]
trustAllCertificatesManager = new                    TrustManager[]{new
X509TrustManager(){
     public X509Certificate[] getAcceptedIssuers(){return  null;}
     public void checkClientTrusted(X509Certificate[]
certs, String authType){}
     public void checkServerTrusted(X509Certificate[]
certs, String authType){}            }};            /* Initialize the
SSLContext with the all-trusting trust manager */
     SSLContext sslContext = SSLContext.getInstance("TLS");
sslContext.init(null, trustAllCertificatesManager, new
SecureRandom());
HttpsURLConnection.setDefaultSSLSocketFactory(sslContext.getSocketFactory(
```

```
));            URL url = new URL(server_url);            conn =
(HttpsURLConnection) url.openConnection();            /* Do not perform an
actual hostname verification during SSL Handshake.            Let all
hostname pass through as verified.*/
conn.setHostnameVerifier(new HostnameVerifier() {               public
boolean verify(String host, SSLSession                        session) {
return true;               }            });        } catch (Exception e)
{            e.printStackTrace();        }        return conn;    }

    /*
     * * This forms the Base64 encoded string using the username and
password *
     * provided by the user. This is required for HTTP Basic
Authentication.
     */ private static String getAuthorizationString() {
        String userPassword = user + ":" + password;
        byte[] authEncodedBytes =
Base64.encodeBase64(userPassword.getBytes());
        String authString = new String(authEncodedBytes);
        return authString;
    }

}
```

저작권 정보

Copyright © 2026 NetApp, Inc. All Rights Reserved. 미국에서 인쇄됨 본 문서의 어떠한 부분도 저작권 소유자의 사전 서면 승인 없이는 어떠한 형식이나 수단(복사, 녹음, 녹화 또는 전자 검색 시스템에 저장하는 것을 비롯한 그래픽, 전자적 또는 기계적 방법)으로도 복제될 수 없습니다.

NetApp이 저작권을 가진 자료에 있는 소프트웨어에는 아래의 라이센스와 고지사항이 적용됩니다.

본 소프트웨어는 NetApp에 의해 '있는 그대로' 제공되며 상품성 및 특정 목적에의 적합성에 대한 명시적 또는 묵시적 보증을 포함하여(이에 제한되지 않음) 어떠한 보증도 하지 않습니다. NetApp은 대체품 또는 대체 서비스의 조달, 사용 불능, 데이터 손실, 이익 손실, 영업 중단을 포함하여(이에 국한되지 않음), 이 소프트웨어의 사용으로 인해 발생하는 모든 직접 및 간접 손해, 우발적 손해, 특별 손해, 징벌적 손해, 결과적 손해의 발생에 대하여 그 발생 이유, 책임론, 계약 여부, 엄격한 책임, 불법 행위(과실 또는 그렇지 않은 경우)와 관계없이 어떠한 책임도 지지 않으며, 이와 같은 손실의 발생 가능성이 통지되었다 하더라도 마찬가지입니다.

NetApp은 본 문서에 설명된 제품을 언제든지 예고 없이 변경할 권리를 보유합니다. NetApp은 NetApp의 명시적인 서면 동의를 받은 경우를 제외하고 본 문서에 설명된 제품을 사용하여 발생하는 어떠한 문제에도 책임을 지지 않습니다. 본 제품의 사용 또는 구매의 경우 NetApp에서는 어떠한 특허권, 상표권 또는 기타 지적 재산권이 적용되는 라이센스도 제공하지 않습니다.

본 설명서에 설명된 제품은 하나 이상의 미국 특허, 해외 특허 또는 출원 중인 특허로 보호됩니다.

제한적 권리 표시: 정부에 의한 사용, 복제 또는 공개에는 DFARS 252.227-7013(2014년 2월) 및 FAR 52.227-19(2007년 12월)의 기술 데이터-비상업적 품목에 대한 권리(Rights in Technical Data -Noncommercial Items) 조항의 하위 조항 (b)(3)에 설명된 제한사항이 적용됩니다.

여기에 포함된 데이터는 상업용 제품 및/또는 상업용 서비스(FAR 2.101에 정의)에 해당하며 NetApp, Inc.의 독점 자산입니다. 본 계약에 따라 제공되는 모든 NetApp 기술 데이터 및 컴퓨터 소프트웨어는 본질적으로 상업용이며 개인 비용만으로 개발되었습니다. 미국 정부는 데이터가 제공된 미국 계약과 관련하여 해당 계약을 지원하는 데에만 데이터에 대한 전 세계적으로 비독점적이고 양도할 수 없으며 재사용이 불가능하며 취소 불가능한 라이센스를 제한적으로 가집니다. 여기에 제공된 경우를 제외하고 NetApp, Inc.의 사전 서면 승인 없이는 이 데이터를 사용, 공개, 재생산, 수정, 수행 또는 표시할 수 없습니다. 미국 국방부에 대한 정부 라이센스는 DFARS 조항 252.227-7015(b)(2014년 2월)에 명시된 권한으로 제한됩니다.

상표 정보

NETAPP, NETAPP 로고 및 http://www.netapp.com/TM에 나열된 마크는 NetApp, Inc.의 상표입니다. 기타 회사 및 제품 이름은 해당 소유자의 상표일 수 있습니다.