Active IQ Unified Manager에서 REST API 액세스 및 인증

Active IQ Unified Manager 9.14

NetApp
October 16, 2025

목차

Active IQ Unified Manager에서 REST API 액세스 및 인증

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

샘플 요청 및 응답:

- * 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* API 서버의 IP 주소 또는 FQDN(정규화된 도메인 이름)입니다.
- 포트 443

  443은 기본 HTTPS 포트입니다. 필요한 경우 HTTPS 포트를 사용자 지정할 수 있습니다.

웹 브라우저에서 HTTP 요청을 실행하려면 REST API 브라우저 플러그인을 사용해야 합니다. CURL 및 Perl과 같은 스크립팅 플랫폼을 사용하여 REST API에 액세스할 수도 있습니다.

인증

Unified Manager는 API에 대한 기본 HTTP 인증 체계를 지원합니다. 보안 정보 흐름(요청 및 응답)을 위해 REST API는 HTTPS를 통해서만 액세스할 수 있습니다. API 서버는 서버 확인을 위해 모든 클라이언트에 자체 서명된 SSL 인증서를 제공합니다. 이 인증서는 사용자 지정 인증서(또는 CA 인증서)로 대체될 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 Administration Console 사용자 인터페이스에 로그인하여 사용자 액세스를 관리할 수 있습니다.

Active IQ Unified Manager에서 사용되는 HTTP 상태 코드입니다

API를 실행하거나 문제를 해결하는 동안 Active IQ Unified Manager API에서 사용되는 다양한 HTTP 상태 코드 및 오류 코드를 알고 있어야 합니다.

다음 표에는 인증과 관련된 오류 코드가 정리되어 있습니다.

HTTP 상태 코드입니다	상태 코드 제목	설명
200	좋습니다	동기 API 호출이 성공적으로 실행될 때 반환됩니다.
201	작성됨	Active Directory 구성과 같은 동기 호출을 통해 새 리소스 생성
202	수락됨	LUN 및 파일 공유 생성과 같은 프로비저닝 기능에 대한 비동기 호출을 성공적으로 실행할 때 반환됩니다.

HTTP 상태 코드입니다	상태 코드 제목	설명
400	잘못된 요청입니다	입력 유효성 검사 실패를 나타냅니다. 사용자는 요청 본문의 유효한 키와 같은 입력을 수정해야 합니다.
401	권한이 없는 요청입니다	리소스를 볼 수 있는 권한이 없습니다.
403	금지된 요청입니다	연결하려는 리소스에 대한 액세스는 금지됩니다.
404	리소스를 찾을 수 없습니다	연결하려는 리소스를 찾을 수 없습니다.
405	메서드가 허용되지 않습니다	메서드가 허용되지 않습니다.
429	요청이 너무 많습니다	사용자가 특정 시간 내에 너무 많은 요청을 보낼 때 반환됩니다.
500	내부 서버 오류입니다	내부 서버 오류입니다. 서버에서 응답을 가져오지 못했습니다. 이 내부 서버 오류는 영구적인 오류일 수도 있고 그렇지 않을 수도 있습니다. 예를 들어 또는 `GET ALL` 작업을 실행하고 이 오류가 발생하는 경우 `GET` 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 Manager용 API 사용을 위한 권장 사항

Active IQ Unified Manager에서 API를 사용하는 경우 특정 권장 방법을 따라야 합니다.

- 유효한 실행을 위해서는 모든 응답 콘텐츠 형식이 다음 형식이어야 합니다.

```
application/json
```

- API 버전 번호가 제품 버전 번호와 관련이 없습니다. Unified Manager 인스턴스에 사용할 수 있는 최신 버전의 API를 사용해야 합니다. Unified Manager API 버전에 대한 자세한 내용은 "Active IQ Unified Manager의 'reST API vers버전 관리' 섹션을 참조하십시오.
- Unified Manager API를 사용하여 어레이 값을 업데이트하는 동안 값의 전체 문자열을 업데이트해야 합니다. 배열에 값을 추가할 수 없습니다. 기존 어레이만 교체할 수 있습니다.
- 메트릭 API의 IOPS 및 성능 등 이중 값을 제외한 모든 쿼리 매개 변수에 파이프(|) 및 와일드 카드(+ *+)와 같은 필터 연산자를 사용할 수 있습니다.

- 필터 연산자 와일드 카드(+ *+)와 파이프(|)를 함께 사용하여 개체를 쿼리하지 마십시오. 잘못된 개체 수를 검색할 수 있습니다.
- 필터에 값을 사용할 때는 해당 값에 문자가 포함되어 있지 ? 않은지 확인합니다. 이는 SQL 주입의 위험을 완화하기 위한 것입니다.
- `GET`모든 `API`에 대한 `(ALL)` 요청은 최대 `1000`개의 레코드를 반환합니다. 매개 변수를 `1000`보다 높은 값으로 설정하여 쿼리를 실행하더라도 `max_records` 1000개의 레코드만 반환됩니다.
- 관리 기능을 수행하려면 Unified Manager UI를 사용하는 것이 좋습니다.

문제 해결을 위한 로그

시스템 로그를 사용하면 API를 실행하는 동안 발생할 수 있는 장애 원인과 문제 해결을 분석할 수 있습니다.

다음 위치에서 로그를 검색하여 API 호출과 관련된 문제를 해결합니다.

로그 위치	사용
`/var/log/ocie/access_log.log`	API 호출, 시작 시간, 실행 시간, 상태 및 URL을 호출하는 사용자의 사용자 이름과 같은 모든 API 호출 세부 정보를 포함합니다. 이 로그 파일을 사용하여 자주 사용하는 API를 확인하거나 GUI 워크플로우를 해결할 수 있습니다. 또한 실행 시간을 기준으로 분석을 확장할 수도 있습니다.
`/var/log/ocum/ocumserver.log`	모든 API 실행 로그를 포함합니다. 이 로그 파일을 사용하여 API 호출 문제를 해결하고 디버깅할 수 있습니다.
`/var/log/ocie/server.log`	모든 Wildfly 서버 배포 및 서비스 시작/중지 관련 로그를 포함합니다. 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 로그를 포함합니다. ONTAP에서 개체를 생성, 수정 또는 삭제했지만 Active IQ Unified Manager REST API에 반영되지 않은 경우 이 로그 파일을 사용할 수 있습니다.

작업 객체 비동기 프로세스

Active IQ Unified Manager `jobs` 다른 API를 실행하는 동안 수행된 작업에 대한 정보를 검색하는 API를 제공합니다. 작업 개체를 사용하여 비동기 처리가 작동하는 방식을 알아야

합니다.

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

작업 개체를 사용하여 설명된 비동기 요청

비동기적으로 실행되는 API 호출을 수행한 후 HTTP 응답 코드 202는 요청이 성공적으로 유효성 확인 및 승인되었지만 아직 완료되지 않았음을 나타냅니다. 요청은 클라이언트에 대한 초기 HTTP 응답 후 계속 실행되는 백그라운드 작업으로 처리됩니다. 응답에는 고유한 식별자를 포함하여 요청을 고정하는 작업 객체가 포함됩니다.

API 요청과 관련된 작업 객체를 쿼리합니다

HTTP 응답에서 반환된 작업 개체에는 여러 속성이 포함되어 있습니다. 상태 속성을 쿼리하여 요청이 성공적으로 완료되었는지 확인할 수 있습니다. 작업 오브젝트는 다음 상태 중 하나일 수 있습니다.

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

작업 개체를 폴링하여 작업의 터미널 상태를 감지할 때 성공 또는 실패 등 두 가지 방법을 사용할 수 있습니다.

- 표준 폴링 요청: 현재 작업 상태가 즉시 반환됩니다.
- 긴 폴링 요청: 작업 상태가 또는 `PARTIAL_FAILURES.` 로 이동하는 경우 `NORMAL, ERROR,`

비동기 요청의 단계입니다

다음 고급 절차를 사용하여 비동기 API 호출을 완료할 수 있습니다.

1. 비동기 API 호출을 실행합니다.
2. 요청을 성공적으로 수락했음을 나타내는 HTTP 응답 202 을 수신합니다.
3. 응답 본문에서 작업 객체의 식별자를 추출합니다.
4. 루프 내에서 작업 객체가 터미널 상태 또는 에 도달할 때까지 `NORMAL, ERROR,` 기다립니다 `PARTIAL_FAILURES.`
5. 작업의 터미널 상태를 확인하고 작업 결과를 가져옵니다.

Hello API 서버

Hello API server_는 간단한 REST 클라이언트를 사용하여 Active IQ Unified Manager에서 REST API를 호출하는 방법을 보여 주는 샘플 프로그램입니다. 샘플 프로그램은 API 서버에 대한 기본 세부 정보를 JSON 형식으로 제공합니다(서버는 형식만 지원 `application/json`).

사용되는 URI는 다음과 같습니다. `https://<hostname>/api/datacenter/svm/svms`. 이 샘플 코드는 다음과 같은 입력 매개 변수를 사용합니다.

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

REST API를 호출하려면 Jersey 및 RESTEasy와 같은 다른 스크립트를 사용하여 Active IQ Unified Manager용 Java REST 클라이언트를 작성할 수도 있습니다. 샘플 코드에 대한 다음 고려 사항을 숙지해야 합니다.

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

샘플 코드를 컴파일하고 실행하려면 Java 컴파일러 1.8 이상을 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

상표 정보