REST-API-Zugriff und Authentifizierung in Active IQ Unified Manager

Active IQ Unified Manager 9.7

NetApp
April 17, 2024

Inhalt

REST-API-Zugriff und Authentifizierung in Active IQ Unified Manager

Auf die Active IQ Unified Manager REST API kann über jeden Webbrowser oder eine Programmierplattform zugegriffen werden, um HTTP-Anfragen zu stellen. Unified Manager unterstützt den grundlegenden HTTP-Authentifizierungsmechanismus. Bevor Sie die UNIFIED Manager REST API aufrufen, müssen Sie einen Benutzer authentifizieren.

REST-Zugriff

Sie können jeden Webbrowser oder jede Programmierplattform verwenden, die HTTP-Anfragen zum Zugriff auf die Unified Manager REST-API ausgeben. Nach der Anmeldung bei Unified Manager können Sie beispielsweise die URL in einen beliebigen Browser eingeben, um die Attribute aller Management-Stationen, wie beispielsweise der Name der Management Station, die Taste und die IP-Adresse, abzurufen.

- **Anfrage**

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

- **Antwort**

```
{
  "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` Ist die IP-Adresse oder der vollqualifizierte Domain-Name (FQDN) des API-Servers.
- Port 443

  Der Standard-HTTPS-Port 443 ist. Sie können den HTTPS-Port bei Bedarf anpassen.

Um HTTP-Anfragen aus einem Webbrowser zu veröffentlichen, ZU PATCHEN und ZU LÖSCHEN, müssen Sie Browser-Plugins verwenden. Sie können auch über Skripting-Plattformen wie Curl und Perl auf DIE REST-API zugreifen.

Authentifizierung

Unified Manager unterstützt das grundlegende HTTP-Authentifizierungsschema für APIs. Für einen sicheren Informationsfluss (Anfrage und Antwort) sind die REST-APIs nur über HTTPS zugänglich. Der API-Server stellt allen Clients ein selbstsigniertes SSL-Zertifikat zur Server-Überprüfung zur Verfügung. Dieses Zertifikat kann durch ein benutzerdefiniertes Zertifikat (oder ein CA-Zertifikat) ersetzt werden.

Sie müssen den Benutzerzugriff auf den API-Server konfigurieren, um die REST-APIs zu aufrufen. Die Benutzer können lokale Benutzer (Benutzerprofile, die in der lokalen Datenbank gespeichert sind) oder LDAP-Benutzer (wenn Sie den API-Server für die Authentifizierung über LDAP konfiguriert haben) sein. Sie können den Benutzerzugriff verwalten, indem Sie sich an der Benutzeroberfläche der Unified Manager Administration Console anmelden.

In Active IQ Unified Manager verwendete HTTP-Statuscodes

Bei Ausführung der APIs oder bei der Fehlerbehebung sollten Sie die verschiedenen HTTP-Statuscodes und -Fehlercodes kennen, die von Active IQ Unified Manager-APIs verwendet werden.

In der folgenden Tabelle sind die Fehlercodes für die Authentifizierung aufgeführt:

HTTP-Statuscode	Titel des Statuscodes	Beschreibung
200	OK	Wird bei der erfolgreichen Ausführung von synchronen API-Aufrufen zurückgegeben.

HTTP-Statuscode	Titel des Statuscodes	Beschreibung
201	Erstellt	Erstellung neuer Ressourcen durch synchrone Anrufe, wie z. B. Konfiguration von Active Directory.
202	Akzeptiert	Wird bei der erfolgreichen Ausführung von asynchronen Aufrufen für Bereitstellungsfunktionen zurückgegeben, z. B. Erstellen von LUNs und File Shares.
400	Ungültige Anforderung	Zeigt Fehler bei der Eingabevalidierung an. Der Benutzer muss die Eingaben korrigieren, z. B. gültige Schlüssel in einem Anforderungskörper.
401	Nicht autorisierte Anforderung	Sie sind nicht berechtigt, die Ressource/Unbefugte anzuzeigen.
403	Anfrage verweigert	Der Zugriff auf die Ressource, die Sie erreichen wollten, ist verboten.
404	Ressource nicht gefunden	Die Ressource, die Sie erreichen wollten, wurde nicht gefunden.
405	Methode Nicht Zulässig	Methode nicht zulässig.
429	Zu Viele Anfragen	Dieser Wert wird zurückgegeben, wenn der Benutzer zu viele Anfragen innerhalb eines bestimmten Zeitraums sendet.

HTTP-Statuscode	Titel des Statuscodes	Beschreibung
500	Interner Serverfehler	Interner Serverfehler. Fehler beim Abrufen der Antwort vom Server. Dieser interne Serverfehler ist möglicherweise permanent oder nicht permanent. Beispiel: Wenn Sie ein ausführen `GET` Oder `GET ALL` Vorgang und Empfang dieses Fehlers. Es wird empfohlen, diesen Vorgang für mindestens fünf Wiederholungen zu wiederholen. Wenn es sich um einen permanenten Fehler handelt, ist der zurückgegebene Statuscode weiterhin 500. Wenn der Vorgang erfolgreich ist, wird der zurückgegebene Statuscode 200 zurückgegeben.

Empfehlungen für die Verwendung der APIs für Active IQ Unified Manager

Bei Verwendung der APIs in Active IQ Unified Manager sollten Sie bestimmte empfohlene Methoden befolgen.

- Alle Arten von Antwortinhalten müssen für eine gültige Ausführung das folgende Format aufweisen:

```
application/json
```

- Die API-Versionsnummer steht nicht zur Produktversionsnummer. Sie sollten die neueste Version der für Ihre Unified Manager Instanz verfügbaren API verwenden. Weitere Informationen zu den Unified Manager API-Versionen finden Sie im Abschnitt „`reST API Versionierung in Active IQ Unified Manager`“.
- Beim Aktualisieren der Array-Werte mithilfe einer Unified Manager API müssen Sie die gesamte Zeichenfolge von Werten aktualisieren. Sie können einem Array keine Werte anhängen. Sie können nur ein vorhandenes Array ersetzen.
- Vermeiden Sie das Abfragen von Objekten, indem Sie eine Kombination aus Wildcard (*) und Rohr („) des Filterbedieners verwenden. Es kann eine falsche Anzahl von Objekten abrufen.
- Beachten Sie, dass die `GET` (Alle) die Anforderung für eine beliebige API gibt maximal 1000 Datensätze zurück. Auch wenn Sie die Abfrage ausführen, indem Sie die einstellen `max_records` Parameter auf einen Wert über 1000, nur 1000 Datensätze werden zurückgegeben.
- Für administrative Aufgaben wird empfohlen, die Unified Manager-Benutzeroberfläche zu verwenden.

Protokolle für die Fehlerbehebung

Mithilfe von Systemprotokollen können Sie die Ursachen eines Ausfalls und die

Behebung von Problemen analysieren, die bei der Ausführung der APIs auftreten können.

Rufen Sie die Protokolle vom folgenden Speicherort ab, um Probleme im Zusammenhang mit den API-Aufrufen zu beheben.

Speicherort protokollieren	Nutzung
`/var/log/ocie/access_log.log`	Enthält alle API-Anrufdetails, z. B. den Benutzernamen des Benutzers, der die API aufruft, Startzeit, Ausführungszeit, Status und URL. In dieser Protokolldatei können Sie die häufig verwendeten APIs überprüfen oder einen Fehler in jedem GUI-Workflow beheben. Sie können die Analyse auch anhand der Ausführungszeit skalieren.
`/var/log/ocum/ocumserver.log`	Enthält alle API-Ausführungsprotokolle. Sie können diese Protokolldatei zur Fehlerbehebung und Fehlersuche bei API-Aufrufen verwenden.
`/var/log/ocie/server.log`	Enthält alle WildFly-Server-Bereitstellungen und Start/Stop-Service-bezogene Protokolle. Mithilfe dieser Protokolldatei können Sie die Ursache von Problemen ermitteln, die während des Starts, Stoppens oder der Bereitstellung des Wislfly-Servers auftreten.
`/var/log/ocie/au.log`	Enthält Protokolle für die Erfassungseinheit. Sie können diese Protokolldatei verwenden, wenn Sie Objekte in ONTAP erstellt, geändert oder gelöscht haben, sie sich jedoch nicht für die Active IQ Unified Manager REST-APIs wiederspiegeln.

Hallo API Server

Der *Hello API-Server* ist ein Beispielprogramm, das zeigt, wie eine REST-API in Active IQ Unified Manager mit einem einfachen REST-Client aufgerufen wird. Das Beispielprogramm enthält grundlegende Details zum API-Server im JSON-Format (nur der Server unterstützt `application/json` Format).

Der verwendete URI ist: `https://<hostname>/api/datacenter/svm/svms`. Dieser Beispielcode nimmt die folgenden Eingabeparameter auf:

- Die IP-Adresse oder FQDN des API-Servers
- Optional: Portnummer (Standard: 443)

- Benutzername
- Passwort
- Antwortformat (`application/json`)

Um REST-APIs aufzurufen, können Sie auch andere Skripte wie Jersey und RESTEasy verwenden, um einen Java REST-Client für Active IQ Unified Manager zu schreiben. Beachten Sie die folgenden Überlegungen zum Beispielcode:

- Verwendet eine HTTPS-Verbindung zu Active IQ Unified Manager, um den angegebenen REST-URI aufzurufen
- Ignoriert das von Active IQ Unified Manager bereitgestellte Zertifikat
- Überspringt die Überprüfung des Host-Namens während des Handshakes
- Verwendet`javax.net.ssl.HttpsURLConnection` Für eine URI-Verbindung
- Verwendet eine Bibliothek eines Drittanbieters (`org.apache.commons.codec.binary.Base64`) Für die Erstellung der Base64 kodierten Zeichenfolge in der HTTP-Grundauthentifizierung verwendet

Um den Beispielcode kompilieren und ausführen zu können, müssen Sie Java Compiler 1.8 oder höher verwenden.

```
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;
    }

}
```