Autenticazione e accesso API REST in Active IQ Unified Manager

Active IQ Unified Manager 9.12

NetApp
December 18, 2023

Sommario

Autenticazione e accesso API REST in Active IQ Unified Manager

L'API REST di Active IQ Unified Manager è accessibile utilizzando qualsiasi client REST o piattaforma di programmazione in grado di emettere richieste HTTP con un meccanismo di autenticazione HTTP di base.

Una richiesta e una risposta di esempio:

- **Richiesta**

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

- **Risposta**

```
{
  "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* È l'indirizzo IP o il nome di dominio completo (FQDN) del server API.
- Porta 443

  443 è la porta HTTPS predefinita. Se necessario, è possibile personalizzare la porta HTTPS.

Per inviare richieste HTTP da un browser Web, è necessario utilizzare i plug-in del browser REST API. È inoltre possibile accedere all'API REST utilizzando piattaforme di scripting come CURL e Perl.

Autenticazione

Unified Manager supporta lo schema di autenticazione HTTP di base per le API. Per un flusso di informazioni sicuro (richiesta e risposta), le API REST sono accessibili solo tramite HTTPS. Il server API fornisce un certificato SSL autofirmato a tutti i client per la verifica del server. Questo certificato può essere sostituito da un certificato personalizzato (o certificato CA).

È necessario configurare l'accesso dell'utente al server API per richiamare le API REST. Gli utenti possono essere utenti locali (profili utente memorizzati nel database locale) o utenti LDAP (se il server API è stato configurato per l'autenticazione su LDAP). È possibile gestire l'accesso degli utenti accedendo all'interfaccia utente di Unified Manager Administration Console.

Codici di stato HTTP utilizzati in Active IQ Unified Manager

Durante l'esecuzione delle API o la risoluzione dei problemi, è necessario conoscere i vari codici di stato HTTP e i codici di errore utilizzati dalle API Active IQ Unified Manager.

La seguente tabella elenca i codici di errore relativi all'autenticazione:

Codice di stato HTTP	Titolo del codice di stato	Descrizione
200	OK	Restituito in caso di esecuzione riuscita di chiamate API sincrone.
201	Creato	Creazione di nuove risorse mediante chiamate sincrone, ad esempio la configurazione di Active Directory.
202	Accettato	Restituito in caso di esecuzione corretta di chiamate asincrone per funzioni di provisioning, come la creazione di LUN e condivisioni di file.

Codice di stato HTTP	Titolo del codice di stato	Descrizione
400	Richiesta non valida	Indica un errore di convalida dell'input. L'utente deve correggere gli input, ad esempio chiavi valide in un corpo di richiesta.
401	Richiesta non autorizzata	Non sei autorizzato a visualizzare la risorsa/non autorizzato.
403	Richiesta non consentita	L'accesso alla risorsa che si stava tentando di raggiungere è vietato.
404	Risorsa non trovata	La risorsa che stavi cercando di raggiungere non è stata trovata.
405	Metodo non consentito	Metodo non consentito.
429	Troppe richieste	Viene restituito quando l'utente invia troppe richieste entro un tempo specifico.
500	Errore interno del server	Errore interno del server. Impossibile ottenere la risposta dal server. Questo errore interno del server potrebbe essere permanente o meno. Ad esempio, se si esegue un `GET` oppure `GET ALL` operazione e si riceve questo errore, si consiglia di ripetere questa operazione per almeno cinque tentativi. Se si tratta di un errore permanente, il codice di stato restituito continua a essere 500. Se l'operazione ha esito positivo, il codice di stato restituito è 200.

Raccomandazioni per l'utilizzo delle API per Active IQ Unified Manager

Quando si utilizzano le API in Active IQ Unified Manager, è necessario seguire alcune procedure consigliate.

- Per un'esecuzione valida, tutti i tipi di contenuto della risposta devono essere nel seguente formato:

```
application/json
```

- Il numero di versione dell'API non è correlato al numero di versione del prodotto. Utilizzare la versione più recente dell'API disponibile per l'istanza di Unified Manager. Per ulteriori informazioni sulle versioni delle API di Unified Manager, vedere la sezione "reversione delle API ST in Active IQ Unified Manager".
- Durante l'aggiornamento dei valori degli array mediante un'API di Unified Manager, è necessario aggiornare l'intera stringa di valori. Non è possibile aggiungere valori a un array. È possibile sostituire solo un array esistente.
- È possibile utilizzare gli operatori di filtro, come pipe (|) e wild card (*) per tutti i parametri di query, ad eccezione dei valori doppi, ad esempio IOPS e performance nelle API delle metriche.
- Evitare di eseguire query sugli oggetti utilizzando una combinazione di wild card (*) e pipe (|) degli operatori di filtro. Potrebbe recuperare un numero di oggetti non corretto.
- Quando si utilizzano i valori per il filtro, assicurarsi che il valore non contenga alcun valore `?` carattere. In questo modo si riducono i rischi di SQL injection.
- Tenere presente che il `GET` (All) la richiesta per qualsiasi API restituisce un massimo di 1000 record. Anche se si esegue la query impostando `max_records` parametro con un valore superiore a 1000, vengono restituiti solo 1000 record.
- Per eseguire le funzioni amministrative, si consiglia di utilizzare l'interfaccia utente di Unified Manager.

Registri per la risoluzione dei problemi

I registri di sistema consentono di analizzare le cause dei guasti e di risolvere i problemi che possono verificarsi durante l'esecuzione delle API.

Recuperare i registri dalla seguente posizione per la risoluzione dei problemi relativi alle chiamate API.

Percorso del log	Utilizzare
`/var/log/ocie/access_log.log`	Contiene tutti i dettagli delle chiamate API, ad esempio il nome utente dell'utente che richiama l'API, l'ora di inizio, l'ora di esecuzione, lo stato e l'URL. È possibile utilizzare questo file di log per controllare le API utilizzate di frequente o per risolvere i problemi di qualsiasi flusso di lavoro GUI. È inoltre possibile utilizzarlo per scalare l'analisi in base al tempo di esecuzione.
`/var/log/ocum/ocumserver.log`	Contiene tutti i log di esecuzione API. È possibile utilizzare questo file di log per risolvere i problemi e eseguire il debug delle chiamate API.
`/var/log/ocie/server.log`	Contiene tutte le implementazioni del server Wildfly e i log relativi al servizio start/stop. È possibile utilizzare questo file di log per individuare la causa principale di eventuali problemi che si verificano durante l'avvio, l'arresto o la distribuzione del server Wildfly.

Percorso del log	Utilizzare
`/var/log/ocie/au.log`	Contiene i log relativi all'unità di acquisizione. È possibile utilizzare questo file di log quando si creano, modificano o eliminano oggetti in ONTAP, ma non vengono riflessi per le API REST di Active IQ Unified Manager.

Processi asincroni degli oggetti di lavoro

Active IQ Unified Manager offre `jobs` API che recupera informazioni sui lavori eseguiti durante l'esecuzione di altre API. È necessario conoscere il funzionamento dell'elaborazione asincrona utilizzando l'oggetto Job.

Alcune delle chiamate API, in particolare quelle utilizzate per l'aggiunta o la modifica delle risorse, possono richiedere più tempo per il completamento rispetto ad altre chiamate. Unified Manager elabora queste richieste a esecuzione prolungata in modo asincrono.

Richieste asincrone descritte utilizzando l'oggetto Job

Dopo aver effettuato una chiamata API eseguita in modo asincrono, il codice di risposta HTTP 202 indica che la richiesta è stata convalidata e accettata correttamente, ma non ancora completata. La richiesta viene elaborata come attività in background che continua a essere eseguita dopo la risposta HTTP iniziale al client. La risposta include l'oggetto Job che ancora la richiesta, incluso il relativo identificatore univoco.

Esecuzione di query sull'oggetto Job associato a una richiesta API

L'oggetto Job restituito nella risposta HTTP contiene diverse proprietà. È possibile eseguire una query sulla proprietà state per determinare se la richiesta è stata completata correttamente. Un oggetto Job può trovarsi in uno dei seguenti stati:

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

Esistono due tecniche che è possibile utilizzare quando si esegue il polling di un oggetto Job per rilevare lo stato di un terminale per l'attività, ovvero riuscito o non riuscito:

- Richiesta di polling standard: Lo stato corrente del processo viene restituito immediatamente.
- Richiesta di polling lunga: Quando lo stato del processo passa a. `NORMAL, ERROR,` oppure `PARTIAL_FAILURES.`

Passaggi in una richiesta asincrona

È possibile utilizzare la seguente procedura di alto livello per completare una chiamata API asincrona:

1. Eseguire la chiamata API asincrona.

2. Ricevere una risposta HTTP 202 che indichi la corretta accettazione della richiesta.
3. Estrarre l'identificatore per l'oggetto Job dal corpo della risposta.
4. All'interno di un loop, attendere che l'oggetto Job raggiunga lo stato terminale `NORMAL, ERROR,` oppure `PARTIAL_FAILURES.`
5. Verificare lo stato terminale del lavoro e recuperare il risultato del lavoro.

Ciao API server

Il *Hello API server* è un programma di esempio che dimostra come richiamare un'API REST in Active IQ Unified Manager utilizzando un semplice client REST. Il programma di esempio fornisce informazioni di base sul server API nel formato JSON (il server supporta solo tale funzione) `application/json` formato).

L'URI utilizzato è: `https://<hostname>/api/datacenter/svm/svms`. Questo codice di esempio utilizza i seguenti parametri di input:

- L'indirizzo IP o FQDN del server API
- Opzionale: Numero di porta (impostazione predefinita: 443)
- Nome utente
- Password
- Formato di risposta (`application/json`)

Per richiamare le API REST, è anche possibile utilizzare altri script come Jersey e RESTEasy per scrivere un client REST Java per Active IQ Unified Manager. Tenere presente le seguenti considerazioni relative al codice di esempio:

- Utilizza una connessione HTTPS a Active IQ Unified Manager per richiamare l'URI REST specificato
- Ignora il certificato fornito da Active IQ Unified Manager
- Ignora la verifica del nome host durante l'handshake
- Utilizzi `javax.net.ssl.HttpsURLConnection` Per una connessione URI
- Utilizza una libreria di terze parti (`org.apache.commons.codec.binary.Base64`) Per la costruzione della stringa codificata Base64 utilizzata nell'autenticazione di base HTTP

Per compilare ed eseguire il codice di esempio, è necessario utilizzare il compilatore Java 1.8 o versione successiva.

```
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;
    }

}
```