Accès à l'API REST et authentification dans Active IQ Unified Manager

Active IQ Unified Manager 9.13

NetApp
December 18, 2023

Sommaire

Accès à l'API REST et authentification dans Active IQ Unified Manager

L'API REST Active IQ Unified Manager est accessible depuis n'importe quel client REST ou plateforme de programmation pouvant émettre des requêtes HTTP avec un mécanisme d'authentification HTTP de base.

Exemple de demande et de réponse :

- **Demande**

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

- **Réponse**

```
{
  "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* Est l'adresse IP ou le nom de domaine complet (FQDN) du serveur d'API.
- Orifice 443

  Le port HTTPS par défaut est défini sur 443. Vous pouvez personnaliser le port HTTPS si nécessaire.

Pour émettre des requêtes HTTP à partir d'un navigateur Web, vous devez utiliser des plug-ins de navigateur d'API REST. Vous pouvez également accéder à l'API REST à l'aide de plateformes de script telles que curl et Perl.

Authentification

Unified Manager prend en charge le schéma d'authentification HTTP de base pour les API. Pour sécuriser les flux d'informations (demande et réponse), les API REST sont accessibles uniquement via HTTPS. Le serveur d'API fournit un certificat SSL auto-signé à tous les clients pour la vérification du serveur. Ce certificat peut être remplacé par un certificat personnalisé (ou un certificat CA).

Vous devez configurer l'accès utilisateur au serveur d'API pour appeler les API REST. Les utilisateurs peuvent être des utilisateurs locaux (profils utilisateur stockés dans la base de données locale) ou des utilisateurs LDAP (si vous avez configuré le serveur d'API pour s'authentifier via LDAP). Vous pouvez gérer l'accès des utilisateurs en vous connectant à l'interface utilisateur de la console d'administration de Unified Manager.

Codes d'état HTTP utilisés dans Active IQ Unified Manager

Lors de l'exécution des API ou de la résolution des problèmes, vous devez connaître les divers codes d'état et codes d'erreur HTTP utilisés par les API Active IQ Unified Manager.

Le tableau suivant répertorie les codes d'erreur liés à l'authentification :

Code d'état HTTP	Titre du code d'état	Description
200	OK	Renvoyé lors de l'exécution réussie des appels d'API synchrone.
201	Créé	Création de nouvelles ressources par des appels synchrones, tels que la configuration d'Active Directory.

Code d'état HTTP	Titre du code d'état	Description
202	Accepté	Renvoyé lors de l'exécution réussie d'appels asynchrones pour les fonctions de provisionnement, telles que la création de LUN et de partages de fichiers.
400	Demande non valide	Indique un échec de validation de l'entrée. L'utilisateur doit corriger les entrées, par exemple les clés valides dans un corps de demande.
401	Demande non autorisée	Vous n'êtes pas autorisé à afficher la ressource/non autorisé.
403	Demande interdite	Il est interdit d'accéder à la ressource que vous tentez d'atteindre.
404	Ressource introuvable	La ressource que vous avez essayé de joindre est introuvable.
405	Méthode non autorisée	Méthode non autorisée.
429	Nombre de demandes trop important	Renvoyé lorsque l'utilisateur envoie trop de demandes dans un délai spécifique.
500	Erreur interne du serveur	Erreur interne du serveur. Impossible d'obtenir la réponse du serveur. Cette erreur interne du serveur peut être permanente ou non. Par exemple, si vous exécutez un `GET` ou `GET ALL` fonctionnement et recevez cette erreur. nous vous recommandons de répéter cette opération pour un minimum de cinq tentatives. S'il s'agit d'une erreur permanente, le code d'état renvoyé continue à être 500. Si l'opération réussit, le code d'état renvoyé est 200.

Recommandations pour l'utilisation des API pour Active IQ Unified Manager

Lorsque vous utilisez des API dans Active IQ Unified Manager, vous devez respecter certaines pratiques recommandées.

- Tous les types de contenu de réponse doivent être au format suivant pour une exécution valide :

```
application/json
```

- Le numéro de version de l'API n'est pas lié au numéro de version du produit. Nous vous recommandons d'utiliser la dernière version de l'API disponible pour votre instance Unified Manager. Pour plus d'informations sur les versions de l'API Unified Manager, reportez-vous à la section « GESTION des versions de l'API DE ST dans Active IQ Unified Manager ».
- Lors de la mise à jour des valeurs d'une baie à l'aide d'une API Unified Manager, vous devez mettre à jour l'ensemble de la chaîne de valeurs. Vous ne pouvez pas ajouter de valeurs à un tableau. Vous ne pouvez remplacer qu'une baie existante.
- Vous pouvez utiliser des opérateurs de filtre, tels que pipe (|) et Wild card (*) pour tous les paramètres de requête, à l'exception des valeurs doubles, par exemple, IOPS et performances dans les API de metrics.
- Évitez d'interroger les objets en utilisant une combinaison de caractères génériques (*) et de tuyaux (|) des opérateurs de filtre. Il est possible que le nombre d'objets soit incorrect.
- Lorsque vous utilisez des valeurs pour le filtre, assurez-vous que la valeur ne contient aucune `?` caractère. Ceci est pour atténuer les risques de l'injection SQL.
- Notez que le `GET` (All) la demande d'une API renvoie un maximum de 1000 enregistrements. Même si vous exécutez la requête en définissant l' `max_records` paramètre à une valeur supérieure à 1000, seuls 1000 enregistrements sont renvoyés.
- Pour effectuer des fonctions administratives, il est recommandé d'utiliser l'interface utilisateur de Unified Manager.

Journaux pour le dépannage

Les journaux système vous permettent d'analyser les causes des défaillances et de résoudre les problèmes susceptibles de survenir lors de l'exécution des API.

Récupérez les journaux à partir de l'emplacement suivant pour résoudre les problèmes liés aux appels API.

Emplacement du journal	Utiliser
`/var/log/ocie/access_log.log`	Contient tous les détails d'appel API, tels que le nom d'utilisateur de l'utilisateur appelant l'API, l'heure de début, l'heure d'exécution, l'état et l'URL. Vous pouvez utiliser ce fichier journal pour vérifier les API fréquemment utilisées ou pour dépanner n'importe quel workflow de l'interface graphique. Vous pouvez également l'utiliser pour mettre l'analyse à l'échelle, en fonction du temps d'exécution.
`/var/log/ocum/ocumserver.log`	Contient tous les journaux d'exécution de l'API. Vous pouvez utiliser ce fichier journal pour dépanner et déboguer les appels API.

Emplacement du journal	Utiliser
`/var/log/ocie/server.log`	Contient tous les déploiements de serveur Wildfly et journaux relatifs au service de démarrage/arrêt. Vous pouvez utiliser ce fichier journal pour trouver la cause principale de tout problème survenant au cours du démarrage, de l'arrêt ou du déploiement du serveur Wildfly.
`/var/log/ocie/au.log`	Contient les journaux relatifs à l'unité d'acquisition. Vous pouvez utiliser ce fichier journal lors de la création, de la modification ou de la suppression d'objets dans ONTAP, mais ils ne sont pas répercutés pour les API REST de Active IQ Unified Manager.

Processus asynchrones des objets de travail

Active IQ Unified Manager offre la solution `jobs` API qui récupère des informations sur les travaux effectués lors de l'exécution d'autres API. Vous devez savoir comment le traitement asynchrone fonctionne à l'aide de l'objet travail.

Certains appels API, en particulier ceux utilisés pour ajouter ou modifier des ressources, peuvent prendre plus de temps que d'autres appels. Unified Manager traite ces requêtes à long terme de manière asynchrone.

Demandes asynchrones décrites à l'aide de l'objet travail

Après avoir effectué un appel API qui s'exécute de manière asynchrone, le code de réponse HTTP 202 indique que la demande a été validée et acceptée avec succès, mais pas encore terminée. La requête est traitée comme une tâche d'arrière-plan qui continue à s'exécuter après la réponse HTTP initiale au client. La réponse inclut l'objet Job qui fixe la requête, y compris son identifiant unique.

Interrogation de l'objet travail associé à une requête API

L'objet travail renvoyé dans la réponse HTTP contient plusieurs propriétés. Vous pouvez interroger la propriété d'état pour déterminer si la demande a bien été effectuée. Un objet travail peut être dans l'un des États suivants :

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

Il existe deux techniques que vous pouvez utiliser lors de l'interrogation d'un objet travail pour détecter un état de terminal pour la tâche, succès ou échec :

- Demande d'interrogation standard : l'état actuel du travail est renvoyé immédiatement.
- Demande d'interrogation longue : lorsque l'état du travail passe à `NORMAL, ERROR,` ou

`PARTIAL_FAILURES`.

Étapes d'une demande asynchrone

Vous pouvez utiliser la procédure de haut niveau suivante pour effectuer un appel d'API asynchrone :

1. Lancez l'appel d'API asynchrone.
2. Recevoir une réponse HTTP 202 indiquant que la demande a été acceptée avec succès.
3. Extraire l'identifiant de l'objet travail du corps de réponse.
4. Dans une boucle, attendez que l'objet travail atteigne l'état du terminal `NORMAL`, `ERROR`, ou `PARTIAL_FAILURES`.
5. Vérifiez l'état du terminal du travail et récupérez le résultat du travail.

Bonjour serveur API

Le *Hello API Server* est un exemple de programme qui montre comment appeler une API REST dans Active IQ Unified Manager à l'aide d'un simple client REST. L'exemple de programme vous fournit des détails de base sur le serveur d'API au format JSON (le serveur ne prend en charge que ce dernier `application/json` format).

L'URI utilisé est : `https://<hostname>/api/datacenter/svm/svms`. Ce code d'échantillon utilise les paramètres d'entrée suivants :

- Adresse IP ou FQDN du serveur d'API
- Facultatif : numéro de port (par défaut : 443)
- Nom d'utilisateur
- Mot de passe
- Format de réponse (`application/json`)

Pour appeler des API REST, vous pouvez aussi utiliser d'autres scripts comme Jersey et RESTEasy pour écrire un client JAVA REST pour Active IQ Unified Manager. Vous devez tenir compte des considérations suivantes concernant le code d'échantillon :

- Utilise une connexion HTTPS vers Active IQ Unified Manager pour appeler l'URI REST spécifiée
- Ignore le certificat fourni par Active IQ Unified Manager
- Ignore la vérification du nom de l'hôte lors de l'établissement de la liaison
- Utilisations `javax.net.ssl.HttpsURLConnection` Pour une connexion URI
- Utilise une bibliothèque tierce (`org.apache.commons.codec.binary.Base64`) Pour construire la chaîne encodée Base64 utilisée dans l'authentification de base HTTP

Pour compiler et exécuter l'exemple de code, vous devez utiliser le compilateur Java 1.8 ou ultérieur.

```
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;
    }

}
```