Acceso y autenticación de API DE REST en Active IQ Unified Manager

Active IQ Unified Manager 9.13

NetApp
December 18, 2023

Tabla de contenidos

Acceso y autenticación de API DE REST en Active IQ Unified Manager

La API de REST de Active IQ Unified Manager es accesible mediante cualquier cliente REST o plataforma de programación que pueda emitir solicitudes HTTP con un mecanismo de autenticación HTTP básico.

Una solicitud y respuesta de muestra:

- **Solicitud**

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

- **Respuesta**

```
{
  "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* Es la dirección IP o el nombre de dominio completo (FQDN) del servidor API.
- Puerto 443

  443 es el puerto HTTPS predeterminado. Puede personalizar el puerto HTTPS, si fuera necesario.

Para emitir solicitudes HTTP desde un explorador web, tiene que utilizar los complementos de explorador de API DE REST. También podrá acceder a la API DE REST usando plataformas de scripting, como curl y Perl.

Autenticación

Unified Manager admite el esquema de autenticación HTTP básico para las API. Para obtener un flujo de información seguro (solicitud y respuesta), se puede acceder a las API DE REST solo a través de HTTPS. El servidor API proporciona un certificado SSL autofirmado a todos los clientes para la verificación del servidor. Este certificado puede sustituirse por un certificado personalizado (o certificado de CA).

Debe configurar el acceso de usuario al servidor API para invocar las API DE REST. Los usuarios pueden ser usuarios locales (perfiles de usuario almacenados en la base de datos local) o usuarios LDAP (si ha configurado el servidor API para autenticarse en LDAP). Para gestionar el acceso de los usuarios, inicie sesión en la interfaz de usuario de Unified Manager Administration Console.

códigos de estado HTTP utilizados en Active IQ Unified Manager

Al ejecutar las API o solucionar problemas, debe tener en cuenta los distintos códigos de estado HTTP y códigos de error que utilizan las API de Active IQ Unified Manager.

En la siguiente tabla se enumeran los códigos de error relacionados con la autenticación:

Código de estado HTTP	Título del código de estado	Descripción
200	DE ACUERDO	Se devolvió al ejecutar correctamente las llamadas API síncronas.
201	Creado	Creación de recursos nuevos mediante llamadas síncronas, como la configuración de Active Directory.

Código de estado HTTP	Título del código de estado	Descripción
202	Aceptado	Devuelto cuando la ejecución correcta de llamadas asíncronas para funciones de aprovisionamiento, como la creación de LUN y recursos compartidos de archivos.
400	Solicitud no válida	Indica fallo de validación de entrada. El usuario tiene que corregir las entradas, por ejemplo, las claves válidas de un cuerpo de solicitud.
401	Solicitud no autorizada	No está autorizado a ver el recurso o no autorizado.
403	Solicitud prohibida	Está prohibido acceder al recurso que estaba intentando alcanzar.
404	No se encuentra el recurso	No se encuentra el recurso al que estaba intentando acceder.
405	Método no permitido	Método no permitido.
429	Demasiadas solicitudes	Devuelto cuando el usuario envía demasiadas solicitudes dentro de un periodo de tiempo específico.
500	Error interno del servidor	Error interno del servidor. Error al obtener la respuesta del servidor. Este error interno del servidor puede ser permanente o no. Por ejemplo, si ejecuta un `GET` o. `GET ALL` operación y recibir este error, se recomienda repetir esta operación por un mínimo de cinco reintentos. Si se trata de un error permanente, el código de estado devuelto sigue siendo 500. Si la operación se realiza correctamente, el código de estado devuelto es 200.

Recomendaciones para el uso de las API para Active IQ Unified Manager

Al usar las API en Active IQ Unified Manager, debe seguir ciertas prácticas

recomendadas.

- Todos los tipos de contenido de la respuesta deben tener el siguiente formato para una ejecución válida:

```
application/json
```

- El número de versión de la API no está relacionado con el número de versión del producto. Debe utilizar la versión más reciente de la API disponible para la instancia de Unified Manager. Si quiere más información acerca de las versiones de la API de Unified Manager, consulte la sección «ARTÍCULO «Artículo DE la creación de versiones de la API en Active IQ Unified Manager».
- Al actualizar los valores de cabinas mediante una API de Unified Manager, debe actualizar toda la cadena de valores. No se pueden agregar valores a una matriz. Solo es posible reemplazar una cabina existente.
- Puede utilizar operadores de filtro, como pipe (|) y comodines (*) para todos los parámetros de consulta, excepto para valores dobles, por ejemplo, IOPS y rendimiento en las API de métricas.
- Evite consultar objetos mediante una combinación de la comoda (*) y la tubería (|) del operador de filtro. Es posible que recupere una cantidad incorrecta de objetos.
- Cuando utilice valores para el filtro, asegúrese de que el valor no contiene ninguno `?` carácter. Esto es para mitigar los riesgos de la inyección SQL.
- Observe que el `GET` (All) la solicitud de cualquier API devuelve un máximo de 1000 registros. Incluso si ejecuta la consulta estableciendo la `max_records` parámetro a un valor superior a 1000, sólo se devuelven 1000 registros.
- Para realizar funciones administrativas, se recomienda usar la interfaz de usuario de Unified Manager.

Registros para solución de problemas

Los registros del sistema le permiten analizar las causas de los errores y solucionar los problemas que pueden surgir al ejecutar las API.

Recupere los registros de la siguiente ubicación para solucionar problemas relacionados con las llamadas API.

Ubicación del registro	Uso
`/var/log/ocie/access_log.log`	Contiene todos los detalles de llamada de la API, como el nombre de usuario del usuario que invoca la API, la hora de inicio, la hora de ejecución, el estado y la URL. Puede usar este archivo de registro para comprobar las API que se usan con frecuencia o solucionar los problemas de cualquier flujo de trabajo de la interfaz gráfica de usuario. También se puede utilizar para ampliar el análisis en función del tiempo de ejecución.

Ubicación del registro	Uso
`/var/log/ocum/ocumserver.log`	Contiene todos los registros de ejecución de la API. Es posible usar este archivo de registro para solucionar problemas y depurar las llamadas API.
`/var/log/ocie/server.log`	Contiene todas las implementaciones de servidores Wildfly y registros relacionados con el servicio de inicio y parada. Puede utilizar este archivo de registro para encontrar la causa raíz de cualquier problema que se produzca durante el inicio, la detención o la implementación del servidor Wildfly.
`/var/log/ocie/au.log`	Contiene registros relacionados con la unidad de adquisición. Puede utilizar este archivo de registro cuando ha creado, modificado o eliminado cualquier objeto de la ONTAP, pero no se reflejan para las API de REST de Active IQ Unified Manager.

Procesos asincrónicos de objetos de trabajo

Active IQ Unified Manager proporciona la `jobs` API que recupera información sobre los trabajos realizados mientras ejecuta otras API. Debe saber cómo funciona el procesamiento asíncrono mediante el objeto Job.

Algunas de las llamadas API, especialmente las que se utilizan para agregar o modificar recursos, pueden tardar más tiempo en completarse que otras llamadas. Unified Manager procesa estas solicitudes de ejecución prolongada de forma asíncrona.

Solicitudes asincrónicas descritas mediante el objeto Job

Después de realizar una llamada API que se ejecuta de forma asíncrona, el código de respuesta HTTP 202 indica que la solicitud se ha validado y aceptado correctamente, pero que aún no se ha completado. La solicitud se procesa como una tarea en segundo plano que continúa ejecutándose después de la respuesta HTTP inicial al cliente. La respuesta incluye el objeto Job anclando la solicitud, incluyendo su identificador único.

Consulta del objeto Job asociado a una solicitud API

El objeto Job devuelto en la respuesta HTTP contiene varias propiedades. Puede consultar la propiedad state para determinar si la solicitud se completó correctamente. Un objeto Job puede estar en uno de los siguientes estados:

- `NORMAL`
- `WARNING`

- `PARTIAL_FAILURES`
- `ERROR`

Existen dos técnicas que se pueden utilizar al sondear un objeto Job para detectar un estado de terminal para la tarea, ya sea con éxito o con un error:

- Solicitud de sondeo estándar: El estado del trabajo actual se devuelve inmediatamente.
- Solicitud de sondeo largo: Cuando el estado del trabajo pasa a. `NORMAL, ERROR,` o. `PARTIAL_FAILURES.`

Pasos en una solicitud asíncrona

Puede utilizar el siguiente procedimiento de alto nivel para completar una llamada API asíncrona:

1. Emita la llamada de API asíncrona.
2. Reciba una respuesta HTTP 202 que indique la aceptación correcta de la solicitud.
3. Extraiga el identificador del objeto Job del cuerpo de respuesta.
4. Dentro de un bucle, espere a que el objeto Job alcance el estado de terminal `NORMAL, ERROR,` o. `PARTIAL_FAILURES.`
5. Compruebe el estado del terminal del trabajo y recupere el resultado del trabajo.

Hola servidor API

El *Hello API Server* es un programa de muestra que muestra cómo invocar una API REST en Active IQ Unified Manager mediante un simple cliente REST. El programa de ejemplo proporciona detalles básicos sobre el servidor API en formato JSON (el servidor solo admite) `application/json` formato).

El URI utilizado es: `https://<hostname>/api/datacenter/svm/svms`. Este código de ejemplo toma los siguientes parámetros de entrada:

- La dirección IP o el FQDN del servidor API
- Opcional: Número de puerto (predeterminado: 443)
- Nombre de usuario
- Contraseña
- Formato de respuesta (`application/json`)

Para invocar API REST, también puede utilizar otras secuencias de comandos, como Jersey y RESTeasy, para escribir un cliente Java REST para Active IQ Unified Manager. Debe tener en cuenta las siguientes consideraciones sobre el código de ejemplo:

- Utiliza una conexión HTTPS con Active IQ Unified Manager para invocar el URI DE REST especificado
- Ignora el certificado proporcionado por Active IQ Unified Manager
- Omite la verificación del nombre del host durante el apretón de manos
- Utiliza `javax.net.ssl.HttpsURLConnection` Para una conexión URI

- Utiliza una biblioteca de terceros (`org.apache.commons.codec.binary.Base64`) Para construir la cadena codificada Base64 utilizada en la autenticación básica HTTP

Para compilar y ejecutar el código de ejemplo, debe utilizar el compilador Java 1.8 o posterior.

```
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;
    }

}
```

Información de copyright

Información de la marca comercial