Active IQ Unified Manager中的 REST API 存取權和身分驗證

Active IQ Unified Manager

NetApp
October 15, 2025

目錄

Active IQ Unified Manager中的 REST API 存取權和身分驗證 . . . 1
- 驗證 . . . 3
- Active IQ Unified Manager中使用的 HTTP 狀態碼 . . . 3
- 使用Active IQ Unified Manager API 的建議 . . . 4
- 故障排除日誌 . . . 5
- Job物件異步進程 . . . 5
  - 使用 Job 物件描述的非同步請求 . . . 5
  - 查詢與 API 請求關聯的 Job 對象 . . . 5
  - 非同步請求中的步驟 . . . 6
- 你好 API 伺服器 . . . 6

Active IQ Unified Manager中的 REST API 存取權和身分驗證

可以使用任何能夠透過基本 HTTP 驗證機制發出 HTTP 要求的 REST 用戶端或程式設計平台存取Active IQ Unified Manager REST API。

範例請求和回應：

- 要求

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

- 回覆

```
{
  "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*`是 API 伺服器的 IP 位址或完全限定網域名稱 (FQDN)。
- 埠 443

  443 是預設 HTTPS 連接埠。如果需要，您可以自訂 HTTPS 連接埠。

若要從 Web 瀏覽器發出 HTTP 請求，您必須使用 REST API 瀏覽器外掛程式。您也可以使用 cURL 和 Perl 等腳本平台存取 REST API。

驗證

Unified Manager 支援 API 的基本 HTTP 驗證方案。為了確保資訊流（請求和回應）的安全，REST API 只能透過 HTTPS 存取。 API 伺服器向所有用戶端提供自簽名 SSL 證書，用於伺服器驗證。此證書可以用自訂證書（或CA證書）替換。

您必須設定使用者對 API 伺服器的存取權限才能呼叫 REST API。使用者可以是本機使用者（儲存在本機資料庫中的使用者設定檔）或 LDAP 使用者（如果您已將 API 伺服器設定為透過 LDAP 進行驗證）。您可以透過登入 Unified Manager 管理控制台使用者介面來管理使用者存取。

Active IQ Unified Manager中使用的 HTTP 狀態碼

在執行 API 或解決問題時，您應該注意Active IQ Unified Manager API 使用的各種 HTTP 狀態碼和錯誤碼。

下表列出了與認證相關的錯誤代碼：

HTTP 狀態碼	狀態代碼標題	描述
200	好的	同步 API 呼叫成功執行後傳回。
201	創建	透過同步呼叫建立新資源，例如 Active Directory 的配置。
202	公認	成功執行設定功能（例如建立 LUN 和檔案共用）的非同步呼叫後返回。
400	無效請求	表示輸入驗證失敗。使用者必須更正輸入，例如請求正文中的有效金鑰。
401	未經授權的請求	您無權查看資源/未經授權。

HTTP 狀態碼	狀態代碼標題	描述
403	禁止請求	禁止存取您嘗試存取的資源。
404	未找到資源	未找到您嘗試存取的資源。
405	方法不允許	方法不允許。
429	請求過多	當用戶在特定時間內發送過多請求時返回。
500	內部伺服器錯誤	內部伺服器錯誤。無法從伺服器取得回應。此內部伺服器錯誤可能是永久性的，也可能不是。例如，如果你運行 `GET`或者 `GET ALL`操作並收到此錯誤，建議您重複此操作至少五次。如果是永久性錯誤，則傳回的狀態代碼繼續為 500。如果操作成功，則回傳的狀態碼為 200。

使用Active IQ Unified Manager API 的建議

使用Active IQ Unified Manager中的 API 時，您應該遵循某些建議的做法。

- 為了有效執行，所有回應內容類型必須採用以下格式：

```
application/json
```

- API版本號與產品版本號無關。您應該使用適用於 Unified Manager 實例的最新版本的 API。有關 Unified Manager API 版本的更多信息，請參閱「Active IQ Unified Manager中的 REST API 版本控制」部分。
- 使用 Unified Manager API 更新陣列值時，必須更新整個值字串。您不能將值附加到數組。您只能替換現有陣列。
- 您可以對所有查詢參數使用篩選運算符，例如垂直線 (|) 和通配符 (*)，但雙精確度值除外，例如指標 API 中的 IOPS 和效能。
- 避免使用篩選運算子通配符 (*) 和管道符 (|) 的組合來查詢物件。它可能會檢索到不正確數量的物件。
- 使用值進行過濾時，請確保值不包含任何 `?`特點。這是為了減輕 SQL 注入的風險。
- 請注意 `GET`任何 API 的 (all) 請求最多傳回 1000 筆記錄。即使您透過設定 `max_records`參數設定為高於 1000 的值，則僅傳回 1000 筆記錄。
- 為了執行管理功能，建議您使用 Unified Manager UI。

故障排除日誌

系統日誌可讓您分析故障原因並解決執行 API 時可能出現的問題。

從下列位置檢索日誌以解決與 API 呼叫相關的問題。

日誌位置	使用
`/var/log/ocie/access_log.log`	包含所有API呼叫詳情，例如呼叫API的使用者的使用者名稱、開始時間、執行時間、狀態、URL等。 您可以使用此日誌檔案檢查常用的 API，或排除任何 GUI 工作流程的故障。您還可以根據執行時間使用它來擴展分析。
`/var/log/ocum/ocumserver.log`	包含所有 API 執行日誌。 您可以使用此日誌檔案來排除故障和偵錯 API 呼叫。
`/var/log/ocie/server.log`	包含所有 Wildfly 伺服器部署和啟動/停止服務相關記錄。 您可以使用此日誌檔案來尋找 Wildfly 伺服器啟動、停止或部署期間發生的任何問題的根本原因。
`/var/log/ocie/au.log`	包含採集單元相關日誌。 當您在ONTAP中建立、修改或刪除任何物件但這些物件不會反映在Active IQ Unified Manager REST API 中時，您可以使用此記錄檔。

Job物件異步進程

Active IQ Unified Manager提供 `jobs`API 會擷取有關在執行其他 API 時執行的作業的資訊。您必須知道如何使用作業對象進行非同步處理。

某些 API 呼叫（尤其是用於新增或修改資源的呼叫）可能比其他呼叫需要更長的時間才能完成。 Unified Manager 非同步處理這些長時間運行的要求。

使用 **Job** 物件描述的非同步請求

進行非同步運行的 API 呼叫後，HTTP 回應代碼 202 表示請求已成功驗證並接受，但尚未完成。該請求將作為後台任務處理，並在客戶端收到初始 HTTP 回應後繼續執行。回應中包含錨定該請求的 Job 物件及其唯一識別碼。

查詢與 **API** 請求關聯的 **Job** 對象

HTTP 回應中傳回的 Job 物件包含多個屬性。您可以查詢 state 屬性來決定請求是否已成功完成。 Job物件可以

處於以下狀態之一：

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

輪詢 Job 物件來偵測任務的最終狀態（成功或失敗）時，可以使用兩種技術：

- 標準輪詢請求：立即返回目前作業狀態。
- 長輪詢請求：當作業狀態轉移到 `NORMAL, ERROR,`或者 `PARTIAL_FAILURES.`

非同步請求中的步驟

您可以使用下列進階程序來完成非同步 API 呼叫：

1. 發出異步 API 呼叫。
2. 收到 HTTP 回應 202，表示成功接受請求。
3. 從回應主體中提取 Job 物件的識別碼。
4. 在循環中，等待 Job 物件達到終止狀態 `NORMAL, ERROR,`或者 `PARTIAL_FAILURES.`
5. 驗證作業的最終狀態並擷取作業結果。

你好 **API** 伺服器

Hello API server 是一個範例程序，示範如何使用簡單的 REST 用戶端呼叫Active IQ Unified Manager中的 REST API。範例程式以 JSON 格式向您提供有關 API 伺服器的基本詳細資訊（伺服器僅支援 `application/json`格式）。

使用的 URI 是：`https://<hostname>/api/datacenter/svm/svms.`此範例程式碼採用以下輸入參數：

- API 伺服器 IP 位址或 FQDN
- 可選：連接埠號碼（預設值：443）
- 使用者名稱
- 密碼
- 回應格式(`application/json`)

若要呼叫 REST API，您也可以使用其他腳本（例如 Jersey 和 RESTEasy）為Active IQ Unified Manager編寫 Java REST 用戶端。您應該了解有關範例程式碼的以下注意事項：

- 使用與Active IQ Unified Manager 的HTTPS 連線來呼叫指定的 REST URI
- 忽略Active IQ Unified Manager提供的憑證
- 握手期間跳過主機名稱驗證

- 用途 `javax.net.ssl.HttpsURLConnection`用於 URI 連接
- 使用第三方函式庫(`org.apache.commons.codec.binary.Base64`) 用於建立 HTTP 基本驗證中使用的 Base64 編碼字串

若要編譯和執行範例程式碼，必須使用 Java 編譯器 1.8 或更高版本。

```
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;
    }

}
```

版權資訊

商標資訊