Active IQ Unified Manager 中的 REST API 存取和驗證

Active IQ Unified Manager

NetApp
November 13, 2026

This PDF was generated from https://docs.netapp.com/zh-tw/active-iq-unified-manager-918/api-automation/concept_rest_api_access_and_authentication_in_um_apis.html on November 13, 2026. Always check docs.netapp.com for the latest.

目錄

Active IQ Unified Manager 中的 REST API 存取和驗證 . . . 1
在 Active IQ Unified Manager 中存取和驗證 REST API . . . 1
驗證 . . . 3
了解 Active IQ Unified Manager HTTP 狀態碼 . . . 3
遵循使用 Active IQ Unified Manager API 的建議做法 . . . 4
使用 Active IQ Unified Manager 中的系統日誌進行故障排除 . . . 4
了解 Active IQ Unified Manager 如何使用作業物件進行非同步處理 . . . 5
使用 Job 物件描述的非同步請求 . . . 5
查詢與 API 請求相關聯的 Job 物件 . . . 5
非同步請求中的步驟 . . . 6
了解 Active IQ Unified Manager 中的 Hello API 伺服器範例方案 . . . 6

Active IQ Unified Manager 中的 REST API 存取和驗證

在 Active IQ Unified Manager 中存取和驗證 REST API

可以使用任何能夠發出具有基本 HTTP 驗證機制之 HTTP 請求的 REST 用戶端或程式設計平台，來存取 Active IQ Unified Manager REST API。

請求和回應範例：

- 要求

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

- 回應

```
{
  "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 位址或完整網域名稱 （FQDN）。
- 連接埠 443

443 是預設的 HTTPS 連接埠。如有需要，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 等指令碼平台存取 REST API。

驗證

Unified Manager 支援 API 的基本 HTTP 驗證方案。對於安全的資訊流（請求和回應），REST API 僅可透過 HTTPS 存取。API 伺服器會向所有用戶端提供自簽的 SSL 憑證以進行伺服器驗證。此憑證可以替換為自訂憑證（或 CA 憑證）。

您必須設定使用者對 API 伺服器的存取才能呼叫 REST API。使用者可以是本機使用者（使用者設定檔儲存在本機資料庫中），也可以是 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 管理主控台使用者介面來管理使用者存取。

了解 Active IQ Unified Manager HTTP 狀態碼

在執行 API 或疑難排解問題時，您應該感知 Active IQ Unified Manager API 使用的各種 HTTP 狀態碼和錯誤代碼。

下表列出與驗證相關的錯誤代碼：

HTTP 狀態碼	狀態碼標題	說明
200	確定	成功執行同步 API 呼叫時傳回。
201	已建立	透過同步呼叫建立新資源，例如組態 Active Directory。
202	已接受	在成功執行資源配置功能（例如建立 LUN 和檔案共享）的非同步呼叫時傳回。
400	無效請求	表示輸入驗證故障。使用者必須更正輸入內容，例如要求主體中的有效金鑰。
401	未經授權的請求	您無權檢視此資源/未授權。
403	禁止的請求	您嘗試存取的資源已被禁止。
404	找不到資源	找不到您嘗試存取的資源。

HTTP 狀態碼	狀態碼標題	說明
405	方法不允許	方法不允許。
429	請求過多	當使用者在特定時間內發送過多請求時傳回此錯誤。
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，如果您執行 `GET` 或 `GET ALL` 作業並收到此錯誤，建議您至少重試此作業五次。如果是永久性錯誤，則傳回的狀態碼將持續為 500。如果作業成功，則傳回的狀態碼為 200。

遵循使用 Active IQ Unified Manager API 的建議做法

使用 Active IQ Unified Manager 中的 API 時，應遵循一些建議做法。

- 為了確保有效執行，所有回應內容類型必須採用以下格式：

```
application/json
```

- API 版本號碼與產品版本號碼無關。您應該使用適用於您 Unified Manager 執行個體的最新 API 版本。如需有關 Unified Manager API 版本的詳細資訊，請參閱「`Active IQ Unified Manager` 中的 `REST API` 版本控制」一節。
- 使用 Unified Manager API 更新陣列值時，您必須更新整個值字串。您無法將值附加至陣列。您只能取代現有的陣列。
- 您可以對所有查詢參數使用篩選運算子，例如管線（|）和萬用字元（*），但雙精確度值除外，例如計量 API 中的 IOPS 和效能。
- 避免結合使用篩選運算子萬用字元（*）和管線（|）來查詢物件。這可能會擷取錯誤數量的物件。
- 使用篩選的值時，請確保該值不包含任何 `?`字元。這是為了降低 SQL 注入的風險。
- 請注意，任何 API 的 `GET` （all）請求最多傳回 1000 筆記錄。即使您將 `max_records` 參數值設為大於 1000 的值來執行查詢，也只會傳回 1000 筆記錄。
- 若要執行管理功能，建議您使用 Unified Manager 使用者介面。

使用 Active IQ Unified Manager 中的系統日誌進行故障排除

系統日誌可以幫助您分析執行 API 時可能出現的故障原因和疑難排解問題。

請從下列位置擷取日誌，以便疑難排解與 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、狀態和 URL。 您可以使用此日誌檔來檢查常用的 API，或對任何 GUI 工作流程進行疑難排解。您也可以根據執行時間使用它來調整分析規模。
`/var/log/ocum/ocumserver.log`	包含所有 API 執行日誌。 您可以使用此日誌檔來疑難排解和偵錯 API 呼叫。
`/var/log/ocie/server.log`	包含所有 Wildfly 伺服器部署和啟動/停止服務相關的日誌。 您可以使用此日誌檔來尋找 Wildfly 伺服器啟動、停止或部署期間發生之任何問題的根本原因。
`/var/log/ocie/au.log`	包含擷取單元相關的日誌。 當您在 ONTAP 中建立、修改或刪除任何物件，但這些變更未反映在 Active IQ Unified Manager REST API 中時，您可以使用此日誌檔。

了解 Active IQ Unified Manager 如何使用作業物件進行非同步處理

Active IQ Unified Manager 提供 `jobs` API，用於擷取執行其他 API 時所執行之作業的相關資訊。您必須了解如何使用作業物件進行非同步處理。

某些 API 呼叫，特別是用於新增或修改資源的呼叫，可能需要比其他呼叫更長的時間才能完成。Unified Manager 會非同步處理這些長時間執行的要求。

使用 Job 物件描述的非同步請求

在進行非同步執行的 API 呼叫後，HTTP 回應代碼 202 表示請求已成功驗證並被接受，但尚未完成。此請求會作為背景工作進行處理，在向用戶端發出初始 HTTP 回應後仍會繼續執行。回應中包含錨定該請求的 Job 物件，以及其唯一識別碼。

查詢與 API 請求相關聯的 Job 物件

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成功完成。Job 物件可以處於以下狀態之一：

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

在輪詢 Job 物件以偵測任務的終止狀態（成功或失敗）時，可以使用兩種技術：

- 標準輪詢要求：立即傳回目前的作業狀態。
- 長輪詢請求：當作業狀態變成 `NORMAL, ERROR,`或 `PARTIAL_FAILURES.`

非同步請求中的步驟

您可以使用下列高階程序來完成非同步 API 呼叫：

1. 發出非同步 API 呼叫。
2. 收到 HTTP 回應 202，表示請求已成功接受。
3. 從回應正文中提取 Job 物件的識別碼。
4. 在迴圈中，等待 Job 物件達到終端狀態 `NORMAL, ERROR,`或 `PARTIAL_FAILURES.`
5. 驗證作業的終止狀態並擷取作業結果。

了解 Active IQ Unified Manager 中的 Hello API 伺服器範例方案

Hello API 伺服器 是一個範例方案，示範如何使用簡單的 REST 用戶端在 Active IQ Unified Manager 中呼叫 REST API。此範例方案以 JSON 格式提供有關 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 為： `https://<hostname>/api/datacenter/svm/svms`. 此範例程式碼接受以下輸入參數：

- API 伺服器 IP 位址或 FQDN
- 選用：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 Jersey 和 RESTEasy 等其他腳本為 Active IQ Unified Manager 寫入 Java REST 用戶端。關於範例程式碼，您應該感知以下事項：

- 使用 HTTPS 連線到 Active IQ Unified Manager 以呼叫指定的 REST URI
- 忽略 Active IQ Unified Manager 提供的憑證
- 信號交換過程中跳過主機名稱驗證
- 使用 `javax.net.ssl.HttpsURLConnection`進行 URI 連線
- 使用第三方程式庫 (`org.apache.commons.codec.binary.Base64`）來建立 HTTP 基本身分驗證中使用的 Base64 編碼字串

若要編譯和執行範例程式碼，您必須使用 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

Copyright © 2026 NetApp, Inc. 版權所有。台灣印製。非經版權所有人事先書面同意，不得將本受版權保護文件的任何部分以任何形式或任何方法（圖形、電子或機械）重製，包括影印、錄影、錄音或儲存至電子檢索系統中。

由 NetApp 版權資料衍伸之軟體必須遵守下列授權和免責聲明：

此軟體以 NETAPP「原樣」提供，不含任何明示或暗示的擔保，包括但不限於有關適售性或特定目的適用性之擔保，特此聲明。於任何情況下，就任何已造成或基於任何理論上責任之直接性、間接性、附隨性、特殊性、懲罰性或衍生性損害（包括但不限於替代商品或服務之採購；使用、資料或利潤上的損失；或企業營運中斷），無論是在使用此軟體時以任何方式所產生的契約、嚴格責任或侵權行為（包括疏忽或其他）等方面，NetApp 概不負責，即使已被告知有前述損害存在之可能性亦然。

NetApp 保留隨時變更本文所述之任何產品的權利，恕不另行通知。NetApp 不承擔因使用本文所述之產品而產生的責任或義務，除非明確經過 NetApp 書面同意。使用或購買此產品並不會在依據任何專利權、商標權或任何其他 NetApp 智慧財產權的情況下轉讓授權。

本手冊所述之產品受到一項（含）以上的美國專利、國外專利或申請中專利所保障。

有限權利說明：政府機關的使用、複製或公開揭露須受 DFARS 252.227-7013（2014 年 2 月）和 FAR 52.227-19（2007 年 12 月）中的「技術資料權利 - 非商業項目」條款 (b)(3) 小段所述之限制。

此處所含屬於商業產品和 / 或商業服務（如 FAR 2.101 所定義）的資料均為 NetApp, Inc. 所有。根據本協議提供的所有 NetApp 技術資料和電腦軟體皆屬於商業性質，並且完全由私人出資開發。 美國政府對於該資料具有非專屬、非轉讓、非轉授權、全球性、有限且不可撤銷的使用權限，僅限於美國政府為傳輸此資料所訂合約所允許之範圍，並基於履行該合約之目的方可使用。除非本文另有規定，否則未經 NetApp Inc. 事前書面許可，不得逕行使用、揭露、重製、修改、履行或展示該資料。美國政府授予國防部之許可權利，僅適用於 DFARS 條款 252.227-7015(b)（2014 年 2 月）所述權利。

商標資訊

NETAPP、NETAPP 標誌及 http://www.netapp.com/TM 所列之標章均為 NetApp, Inc. 的商標。文中所涉及的所有其他公司或產品名稱，均為其各自所有者的商標，不得侵犯。