Active IQ Unified Manager 中的 REST API 访问和身份验证

Active IQ Unified Manager

NetApp
November 13, 2026

This PDF was generated from https://docs.netapp.com/zh-cn/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.

目录

Active IQ Unified Manager 中的 REST API 访问和身份验证 . . . 1
在 Active IQ Unified Manager 中访问和验证 REST API . . . 1
身份验证 . . . 3
了解 Active IQ Unified Manager HTTP 状态代码 . . . 3
请遵循使用 Active IQ Unified Manager API 的推荐做法 . . . 4
使用 Active IQ Unified Manager 中的系统日志进行故障排除 . . . 4
了解 Active IQ Unified Manager 如何使用作业对象进行异步处理 . . . 5
使用 Job 对象描述的异步请求 . . . 5
查询与 API 请求关联的作业对象 . . . 5
异步请求中的步骤 . . . 6
了解 Active IQ Unified Manager 中的 Hello API server 示例程序 . . . 6

Active IQ Unified Manager 中的 REST API 访问和身份验证

在 Active IQ Unified Manager 中访问和验证 REST API

可以使用任何能够通过基本 HTTP 身份验证机制发出 HTTP 请求的 REST 客户端或编程平台来访问 Active IQ Unified Manager REST API。

请求和响应示例：

- 请求

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

- 响应

```
{
  "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 服务器的 IP 地址或完全限定的域名（FQDN）。
- 端口 443

443 是默认的 HTTPS 端口。如果需要，您可以自定义 HTTPS 端口。

要从 Web 浏览器发出 HTTP 请求，您必须使用 REST API 浏览器插件。您还可以使用脚本平台（如 cURL 和 Perl）访问 REST API。

身份验证

Unified Manager 支持 API 的基本 HTTP 身份验证方案。为了确保信息流（请求和响应）的安全，REST API 只能通过 HTTPS 访问。API 服务器向所有客户端提供自签名 SSL 证书以进行服务器验证。此证书可由自定义证书（或 CA 证书）替换。

必须配置用户对 API 服务器的访问权限以调用 REST API。用户可以是本地用户（存储在本地数据库中的用户配置文件）或 LDAP 用户（如果已将 API 服务器配置为通过 LDAP 进行身份验证）。您可以通过登录到 Unified Manager 管理控制台用户界面来管理用户访问。

了解 Active IQ Unified Manager HTTP 状态代码

在运行 API 或排除问题时，应注意 Active IQ Unified Manager API 使用的各种 HTTP 状态代码和错误代码。

下表列出了与身份验证相关的错误代码：

HTTP 状态代码	状态代码标题	问题描述
200	确定	成功执行同步 API 调用时返回。
201	已创建	通过同步调用创建新资源，例如配置 Active Directory。
202	已接受	成功执行配置功能的异步调用时返回，例如创建 LUN 和文件共享。
400	无效请求	表示输入验证失败。用户必须更正输入，例如请求正文中的有效键。
401	未经授权的请求	您无权查看资源/未授权。
403	禁止的请求	禁止访问您尝试访问的资源。
404	未找到资源	未找到您尝试访问的资源。
405	方法不允许	方法不允许。

HTTP 状态代码	状态代码标题	问题描述
429	请求过多	当用户在特定时间内发送太多请求时返回。
500	内部服务器错误	内部服务器错误。无法从服务器获取响应。此内部服务器错误可能是也可能不是永久性的。例如，如果您运行 `GET`或 `GET ALL`操作并收到此错误，建议您至少重试此操作五次。如果是永久性错误，则返回的状态代码仍为 500。如果操作成功，则返回的状态代码为 200。

请遵循使用 Active IQ Unified Manager API 的推荐做法

在 Active IQ Unified Manager 中使用 API 时，应遵循某些推荐做法。

- 所有响应内容类型必须采用以下格式才能有效执行：

```
application/json
```

- API 版本号与产品版本号无关。您应使用适用于您的 Unified Manager 实例的最新可用 API 版本。有关 Unified Manager API 版本的详细信息，请参见“`REST API versioning in Active IQ Unified Manager`”部分。
- 在使用 Unified Manager API 更新数组值时，您必须更新整个值字符串。您不能向数组附加值。您只能替换现有的数组。
- 您可以对所有查询参数使用筛选器运算符，例如管道（|）和通配符（*），但双精度值（例如指标 API 中的 IOPS 和性能）除外。
- 避免使用过滤器运算符通配符（*）和管道（|）的组合来查询对象。这可能会检索到数量不正确的对象。
- 使用筛选器值时，请确保此值不包含任何 `?`字符。这是为了降低 SQL 注入的风险。
- 请注意，任何 API 的 `GET` （all）请求最多返回 1000 条记录。即使通过将 `max_records` 参数设置为大于 1000 的值来运行查询，也只会返回 1000 条记录。
- 要执行管理功能，建议使用 Unified Manager UI。

使用 Active IQ Unified Manager 中的系统日志进行故障排除

系统日志可用于分析运行 API 时可能出现的故障原因和故障排除问题。

从以下位置检索日志，以解决与 API 调用相关的问题。

日志位置	用途
`/var/log/ocie/access_log.log`	包含所有 API 调用详细信息，例如调用 API 的用户的用户名、开始时间、执行时间、状态和 URL。 您可以使用此日志文件检查常用 API，或对任何 GUI 工作流进行故障排除。您还可以根据执行时间使用它进行缩放分析。
`/var/log/ocum/ocumserver.log`	包含所有 API 执行日志。 您可以使用此日志文件对 API 调用进行故障排除和调试。
`/var/log/ocie/server.log`	包含所有 Wildfly 服务器部署和启动/停止服务相关日志。 您可以使用此日志文件查找在 Wildfly 服务器的启动、停止或部署期间发生的任何问题的根本原因。
`/var/log/ocie/au.log`	包含与采集单元相关的日志。 当您在 ONTAP 中创建、修改或删除了任何对象，但它们未反映在 Active IQ Unified Manager REST API 中时，您可以使用此日志文件。

了解 Active IQ Unified Manager 如何使用作业对象进行异步处理

Active IQ Unified Manager 提供了 `jobs` API，用于检索有关在运行其他 API 时执行的作业的信息。您必须了解如何使用作业对象进行异步处理。

某些 API 调用，特别是用于添加或修改资源的 API 调用，可能需要比其他调用更长的时间才能完成。Unified Manager 异步处理这些长期运行的请求。

使用 Job 对象描述的异步请求

在进行异步运行的 API 调用后，HTTP 响应代码 202 表示请求已成功验证和接受，但尚未完成。此请求作为后台任务处理，此后台任务在对客户端的初始 HTTP 响应后继续运行。响应包括锚定请求的 Job 对象，包括其唯一标识符。

查询与 API 请求关联的作业对象

HTTP 响应中返回的 Job 对象包含多个属性。您可以查询 state 属性以确定请求是否成功完成。Job 对象可以处于以下状态之一：

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

在轮询 Job 对象以检测任务的终端状态时，您可以使用两种技术，无论是成功还是失败：

- 标准轮询请求：立即返回当前作业状态。
- 长轮询请求：当作业状态变为 `NORMAL, ERROR,`或 `PARTIAL_FAILURES.`

异步请求中的步骤

您可以使用以下高级过程完成异步 API 调用：

1. 发出异步 API 调用。
2. 接收 HTTP 响应 202，指示成功接受请求。
3. 从响应正文中提取 Job 对象的标识符。
4. 在循环中，等待 Job 对象到达终止状态 `NORMAL, ERROR,`或 `PARTIAL_FAILURES.`
5. 验证作业的最终状态并获取作业结果。

了解 Active IQ Unified Manager 中的 Hello API server 示例程序

Hello API server 是一个示例程序，演示了如何使用简单的 REST 客户端在 Active IQ Unified Manager 中调用 REST API。该示例程序以 JSON 格式（服务器仅支持 `application/json`格式）为您提供有关 API 服务器的基本详细信息。

使用的 URI 为： `https://<hostname>/api/datacenter/svm/svms.`此示例代码采用以下输入参数：

- API 服务器 IP 地址或 FQDN
- 可选：端口号（默认：443）
- 用户名
- 密码
- 响应格式 (`application/json`)

要调用 REST API，您还可以使用其他脚本（如 Jersey 和 RESTEasy）为 Active IQ Unified Manager 编写 Java REST 客户端。您应该了解有关示例代码的以下注意事项：

- 使用到 Active IQ Unified Manager 的 HTTPS 连接来调用指定的 REST URI
- 忽略 Active IQ Unified Manager 提供的证书
- 在握手过程中跳过主机名验证
- 使用 `javax.net.ssl.HttpsURLConnection`进行 URI 连接
- 使用第三方库(`org.apache.commons.codec.binary.Base64`）构造 HTTP 基本身份验证中使用的 Base64 编码字符串

要编译和执行示例代码，您必须使用 Java 编译器 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;
    }

}
```

版权信息

版权所有 © 2026 NetApp, Inc.。保留所有权利。中国印刷。未经版权所有者事先书面许可，本文档中受版权保护的任何部分不得以任何形式或通过任何手段（图片、电子或机械方式，包括影印、录音、录像或存储在电子检索系统中）进行复制。

从受版权保护的 NetApp 资料派生的软件受以下许可和免责声明的约束：

本软件由 NetApp 按“原样”提供，不含任何明示或暗示担保，包括但不限于适销性以及针对特定用途的适用性的隐含担保，特此声明不承担任何责任。在任何情况下，对于因使用本软件而以任何方式造成的任何直接性、间接性、偶然性、特殊性、惩罚性或后果性损失（包括但不限于购买替代商品或服务；使用、数据或利润方面的损失；或者业务中断），无论原因如何以及基于何种责任理论，无论出于合同、严格责任或侵权行为（包括疏忽或其他行为），NetApp 均不承担责任，即使已被告知存在上述损失的可能性。

NetApp 保留在不另行通知的情况下随时对本文档所述的任何产品进行更改的权利。除非 NetApp 以书面形式明确同意，否则 NetApp 不承担因使用本文档所述产品而产生的任何责任或义务。使用或购买本产品不表示获得 NetApp 的任何专利权、商标权或任何其他知识产权许可。

本手册中描述的产品可能受一项或多项美国专利、外国专利或正在申请的专利的保护。

有限权利说明：政府使用、复制或公开本文档受 DFARS 252.227-7013（2014 年 2 月）和 FAR 52.227-19（2007 年 12 月）中“技术数据权利 — 非商用”条款第 (b)(3) 条规定的限制条件的约束。

本文档中所含数据与商业产品和/或商业服务（定义见 FAR 2.101）相关，属于 NetApp, Inc. 的专有信息。根据本协议提供的所有 NetApp 技术数据和计算机软件具有商业性质，并完全由私人出资开发。 美国政府对这些数据的使用权具有非排他性、全球性、受限且不可撤销的许可，该许可既不可转让，也不可再许可，但仅限在与交付数据所依据的美国政府合同有关且受合同支持的情况下使用。除本文档规定的情形外，未经 NetApp, Inc. 事先书面批准，不得使用、披露、复制、修改、操作或显示这些数据。美国政府对国防部的授权仅限于 DFARS 的第 252.227-7015(b)（2014 年 2 月）条款中明确的权利。

商标信息

NetApp、NetApp 标识和 http://www.netapp.com/TM 上所列的商标是 NetApp, Inc. 的商标。其他公司和产品名称可能是其各自所有者的商标。