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

Active IQ Unified Manager 9.10

NetApp
October 16, 2025

This PDF was generated from https://docs.netapp.com/zh-cn/active-iq-unified-manager-910/api-automation/reference_authentication_errors.html on October 16, 2025. 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
- 身份验证 . . . 3
- Active IQ Unified Manager 中使用的 HTTP 状态代码 . . . 3
- 有关使用 Active IQ Unified Manager API 的建议 . . . 4
- 用于故障排除的日志 . . . 5
- 作业对象异步进程 . . . 5
  - 使用作业对象描述的异步请求 . . . 5
  - 正在查询与 API 请求关联的作业对象 . . . 6
  - 异步请求中的步骤 . . . 6
- 您好， API 服务器 . . . 6

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

Active IQ Unified Manager REST API可通过任何可以使用基本HTTP身份验证机制对HTTP请求执行问题描述 的REST客户端或编程平台进行访问。

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 地址 / 主机名 _` 是 API 服务器的 IP 地址或完全限定域名（ FQDN ）。
- 端口 443

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

要从Web浏览器对HTTP请求进行问题描述 处理、您必须使用REST API浏览器插件。您也可以使用 curl 和 Perl 等脚本平台访问 REST API 。

身份验证

Unified Manager 支持 API 的基本 HTTP 身份验证方案。对于安全信息流（请求和响应），只能通过 HTTPS 访问 REST API 。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。此证书可替换为自定义证书（或 CA 证书）。

您必须配置用户对 API 服务器的访问权限，以便调用 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。

Active IQ Unified Manager 中使用的 HTTP 状态代码

在运行 API 或解决问题时，您应了解 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	Description
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	成功执行异步调用以执行配置功能（例如创建 LUN 和文件共享）时返回。
400	请求无效	指示输入验证失败。用户必须更正输入，例如，请求正文中的有效密钥。
401	未授权请求	您无权查看此资源 / 未授权。

HTTP 状态代码	状态代码标题	Description
403	已禁止请求	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.	不允许使用此方法	不允许使用此方法
429	请求过多	如果用户在特定时间内发送的请求过多，则返回此消息。
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永久性的，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，建议您重复此操作至少五次重试。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 。

有关使用 Active IQ Unified Manager API 的建议

在 Active IQ Unified Manager 中使用 API 时，应遵循某些建议的做法。

- 要有效执行，所有响应内容类型必须采用以下格式：

```
application/json
```

- API 版本号与产品版本号无关。您应使用 Unified Manager 实例可用的最新版本 API 。有关 Unified Manager API 版本的详细信息，请参见 " `re`在 `Active IQ Unified Manager` 中版本控制 API" 一节。
- 使用 Unified Manager API 更新阵列值时，必须更新整个值字符串。您不能将值附加到数组。您只能替换现有阵列。
- 您可以对所有查询参数使用筛选器运算符，例如管道（我们）和通配符（ + ），但双值除外，例如，指标 API 中的 IOPS 和性能。
- 通过结合使用筛选器运算符通配符（ * ）和管道（ | ）来避免查询对象。它可能检索的对象数不正确。
- 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，对于任何 API ， `get` （ all ）请求最多可返回 1000 条记录。即使您通过将 `max_records` 参数设置为大于 1000 的值来运行查询，也仅返回 1000 条记录。
- 要执行管理功能，建议使用 Unified Manager UI 。

用于故障排除的日志

通过系统日志，您可以分析失败的原因，并对运行 API 时可能出现的问题进行故障排除。

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	使用 ...
` /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 ，或者对任何 GUI 工作流进行故障排除。您还可以使用它根据执行时间扩展分析。
` /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`	包含所有与 Wildfly 服务器部署和启动 / 停止服务相关的日志。 您可以使用此日志文件查找启动，停止或部署 Wildfly 服务器期间发生的任何问题的根发生原因。
` /var/log/ocie/au.log`	包含与采集单元相关的日志。 在 ONTAP 中创建，修改或删除了任何对象，但这些对象未反映在 Active IQ Unified Manager REST API 中时，您可以使用此日志文件。

作业对象异步进程

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。您必须了解使用作业对象时异步处理的工作原理。

某些 API 调用（尤其是用于添加或修改资源的 API 调用）的完成时间可能比其他调用更长。Unified Manager 会异步处理这些长时间运行的请求。

使用作业对象描述的异步请求

发出异步运行的 API 调用后， HTTP 响应代码 202 表示此请求已成功验证并被接受，但尚未完成。此请求将作为后台任务进行处理，在对客户端进行初始 HTTP 响应后，此任务将继续运行。响应包括作业对象锁定请求，包括其唯一标识符。

正在查询与 API 请求关联的作业对象

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

- 正常
- 警告
- `partial_failures`
- 错误

在轮询作业对象以检测任务的终端状态时，可以使用两种方法：成功或失败：

- 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` 时。

异步请求中的步骤

您可以使用以下高级操作步骤完成异步 API 调用。

1. 问题描述异步 API 调用。
2. 接收表示已成功接受请求的 HTTP 响应 202 。
3. 从响应正文中提取作业对象的标识符。
4. 在环路中，等待作业对象达到终端状态 `normal` ， `error` ，或 `partial_failures` 。
5. 验证作业的终端状态并检索作业结果。

您好， API 服务器

Hello API 服务器 _ 是一个示例程序，用于演示如何使用简单的 REST 客户端在 Active IQ Unified Manager 中调用 REST API 。此示例程序以 JSON 格式（此服务器仅支持 `application/json` 格式）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
- Password
- 响应格式（`application/json` ）

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

- 使用与 Active IQ Unified Manager 的 HTTPS 连接调用指定的 REST URI

- 忽略 Active IQ Unified Manager 提供的证书
- 在握手期间跳过主机名验证
- 对 URI 连接使用 `javax.net.ssl.HttpsURLConnection`
- 使用第三方库（`org.apache.commons.codec.binary.Base64` ）构建 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，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 © 2025 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。