Active IQ Unified ManagerでのREST APIへのアクセスおよび認証

Active IQ Unified Manager

NetApp
November 13, 2026

This PDF was generated from https://docs.netapp.com/ja-jp/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.

目次

Active IQ Unified ManagerでのREST APIへのアクセスおよび認証 . . . 1
- Active IQ Unified Manager で REST API にアクセスして認証する . . . 1
  - 認証 . . . 3
- Active IQ Unified Manager の HTTP ステータスコードについて . . . 3
- Active IQ Unified Manager API を使用する際は、推奨されるプラクティスに従ってください。 . . . 4
- Active IQ Unified Manager でシステムログを使用してトラブルシューティングを行う . . . 5
- Active IQ Unified Manager が非同期処理にジョブオブジェクトを使用する方法を理解する . . . 6
  - ジョブ オブジェクトを使用して記述された非同期要求 . . . 6
  - API リクエストに関連付けられた Job オブジェクトを照会する . . . 6
  - 非同期要求の手順 . . . 7
- Active IQ Unified Manager の Hello API サーバーサンプルプログラムについて . . . 7

Active IQ Unified ManagerでのREST APIへのアクセスおよび認証

Active IQ Unified Manager で REST API にアクセスして認証する

Active IQ Unified Manager REST APIには、基本的なHTTP認証メカニズムによるHTTP要求を実行できる任意のRESTクライアントまたはプログラミング プラットフォームを使用してアクセスできます。

要求と応答の例を次に示します。

- リクエスト

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

- 応答

```
{
  "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* は、APIサーバーのIPアドレスまたは完全修飾ドメイン名（FQDN）です。
- ポート443

443は、デフォルトのHTTPSポートです。必要に応じて、HTTPSポートをカスタマイズできます。Webブラウザから HTTP要求を実行するには、REST APIブラウザ プラグインを使用する必要があります。cURLやPerlなどのスクリプト プラットフォームを使用して、REST APIにアクセスすることもできます。

認証

Unified Managerでは、APIのベーシックHTTP認証方式がサポートされます。情報の流れ（要求と応答）をセキュリティで保護するために、REST APIにはHTTPS経由でのみアクセスできます。APIサーバは、サーバ検証のためにすべてのクライアントに自己署名のSSL証明書を提供します。この証明書は、カスタム証明書（またはCA証明書）で置き換えることができます。

REST APIを呼び出すには、APIサーバへのユーザ アクセスを設定する必要があります。ユーザには、ローカル ユーザ（ローカル データベースに格納されているユーザ プロファイル）またはLDAPユーザ（LDAP経由で認証するようにAPIサーバを設定している場合）を指定できます。Unified Manager Administration Consoleのユーザ インターフェイスにログインして、ユーザ アクセスを管理できます。

Active IQ Unified Manager の HTTP ステータスコードについて

API を実行したり、問題をトラブルシューティングしたりする際に、Active IQ Unified Manager API で使用されるさまざまな HTTP ステータス コードとエラー コードについて知っておく必要があります。

次の表は、認証に関連するエラー コードを示しています。

HTTPステータス コード	ステータスコードタイトル	説明
200	OK	同期API呼び出しの実行に成功した場合に返されます。
201	Created	Active Directoryの設定など、同期呼び出しによって新しいリソースが作成されたことを示します。

HTTPステータス コード	ステータスコードタイトル	説明
202	Accepted	LUNやファイル共有の作成など、プロビジョニング機能の非同期呼び出しの実行が成功したときに返されます。
400	Invalid request	入力検証に失敗したことを示します。ユーザは、要求本文に有効なキーを指定するなど、入力を修正する必要があります。
401	Unauthorized request	リソースの表示を許可されていません/権限がありません。
403	Forbidden request	接続しようとしたリソースへのアクセスは禁止されています。
404	Resource not found	アクセスしようとしたリソースが見つかりません。
405	Method Not Allowed	許可されていないメソッドです。
429	Too Many Requests	ユーザから一定の時間内に送信された要求が多すぎる場合に返されます。
500	Internal server error	内部サーバーエラー。サーバーからの応答を取得できませんでした。この内部サーバーエラーは、永続的なものとは限りません。例えば、 `GET`または `GET ALL`操作中にこのエラーが発生した場合は、最低5回は操作を繰り返すことをお勧めします。永続的なエラーの場合は、返されるステータスコードは引き続き500になります。操作が成功した場合、返されるステータスコードは200です。

Active IQ Unified Manager API を使用する際は、推奨されるプラクティスに従ってください。

Active IQ Unified Manager で API を使用する場合は、いくつかの推奨プラクティスに従う必要があります。

- 応答のコンテンツ タイプをすべて次の形式にする必要があります。

```
application/json
```

- APIのバージョン番号は、製品のバージョン番号とは関係ありません。Unified Managerインスタンスで利用可能な最新バージョンのAPIを使用してください。Unified Manager APIのバージョンの詳細については、「REST API versioning in Active IQ Unified Manager」セクションを参照してくださ

い。

- Unified Manager APIを使用して配列値を更新する場合は、値の文字列全体を更新する必要があります。配列に値を付加することはできません。既存の配列のみを置換できます。
- パイプ（|）やワイルドカード（*）などのフィルタ演算子は、double値（たとえば、メトリクスAPIのIOPSやパフォーマンスなど）を除くすべてのクエリパラメータに使用できます。
- フィルタ演算子のワイルドカード（*）とパイプ（|）を組み合わせてオブジェクトをクエリすることは避けてください。誤った数のオブジェクトが取得される可能性があります。
- フィルターに値を使用する場合は、値に `?`文字が含まれていないことを確認してください。これはSQLインジェクションのリスクを軽減するための措置です。
- （すべての）APIに対する `GET`リクエストは、最大1000件のレコードを返すことに注意してください。`max_records`パラメータを1000より大きい値に設定してクエリを実行しても、返されるレコードは1000件のみです。
- 管理機能を実行するには、Unified Manager UIを使用することをお勧めします。

Active IQ Unified Manager でシステムログを使用してトラブルシューティングを行う

システム ログを使用して、APIの実行中に発生する可能性のある障害の原因を分析し、問題のトラブルシューティングを実行できます。

API呼び出しに関連する問題のトラブルシューティングを行うには、次の場所からログを取得します。

ログの場所	用途
`/var/log/ocie/access_log.log`	APIを呼び出しているユーザ名、開始時刻、実行時間、ステータス、URLなど、API呼び出しに関するすべての詳細情報が含まれます。 このログ ファイルを使用して、頻繁に使用されるAPIを確認したり、GUIワークフローのトラブルシューティングを行うことができます。また、実行時間に基づいて分析の規模を調整することもできます。
`/var/log/ocum/ocumserver.log`	すべてのAPI実行ログが含まれます。 このログ ファイルを使用して、API呼び出しのトラブルシューティングとデバッグを行うことができます。
`/var/log/ocie/server.log`	すべてのWildflyサーバ構成と、サービスの開始 / 停止に関連するログが含まれています。 このログ ファイルを使用して、Wildflyサーバの開始、停止、または導入中に発生した問題の根本原因を見つけることができます。

ログの場所	用途
`/var/log/ocie/au.log`	Acquisition Unit関連のログが含まれます。 ONTAP でオブジェクトを作成、変更、または削除したにもかかわらず、Active IQ Unified Manager REST API に反映されない場合に、このログファイルを使用できます。

Active IQ Unified Manager が非同期処理にジョブオブジェクトを使用する方法を理解する

Active IQ Unified Managerは、他のAPIの実行中に実行されたジョブに関する情報を取得する `jobs` APIを提供します。Jobオブジェクトを使用した非同期処理の仕組みを理解しておく必要があります。

一部のAPI呼び出し、特にリソースの追加や変更に使用されるAPI呼び出しは、他の呼び出しよりも完了に時間がかかることがあります。Unified Managerは、これらの長時間実行されるリクエストを非同期的に処理します。

ジョブ オブジェクトを使用して記述された非同期要求

非同期的に実行されるAPI呼び出しを行うと、HTTP応答コード202が返されます。この応答コードは、要求が正常に検証され受け入れられたものの、まだ完了していないことを示します。要求はバックグラウンド タスクとして処理され、クライアントへの最初のHTTP応答後も引き続き実行されます。応答には、要求に対応するジョブ オブジェクトと、その一意の識別子が含まれます。

API リクエストに関連付けられた Job オブジェクトを照会する

HTTP応答で返されるジョブ オブジェクトには、いくつかのプロパティが含まれています。状態プロパティを照会して、要求が正常に完了したかどうかを確認できます。ジョブ オブジェクトは次のいずれかの状態になります。

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

ジョブ オブジェクトをポーリングしてタスクの最終状態（成功または失敗）を検出するには、2つの方法があります。

- 標準のポーリング要求：現在のジョブの状態が即座に返されます。
- ロングポーリングリクエスト：ジョブの状態が `NORMAL, ERROR,`または `PARTIAL_FAILURES.`に移行したとき

非同期要求の手順

以下は、非同期API呼び出しを完了する手順の概要です。

1. 非同期API呼び出しを実行します。
2. 要求が正常に受け取られたことを示すHTTP応答202を受信します。
3. 応答の本文からジョブ オブジェクトの識別子を抽出します。
4. ループ内で、Jobオブジェクトが終了状態に達するまで待機します `NORMAL`, `ERROR`,`または `PARTIAL_FAILURES`.
5. ジョブの最終状態を確認し、ジョブの結果を取得します。

Active IQ Unified Manager の Hello API サーバーサンプルプログラムについて

Hello API server は、シンプルな REST クライアントを使用して Active IQ Unified Manager で REST API を呼び出す方法を示すサンプルプログラムです。サンプルプログラムは、JSON 形式で API サーバーに関する基本的な詳細情報を提供します（サーバーは `application/json`形式のみをサポートします）。

使用する URI は次のとおりです： `https://<hostname>/api/datacenter/svm/svms`. このサンプルコードは、次の入力パラメータを使用します：

- APIサーバのIPアドレスまたはFQDN
- オプション：ポート番号（デフォルト：443）
- ユーザ名
- パスワード
- 回答形式 (`application/json`)

REST API を呼び出すには、Jersey や RESTEasy などの他のスクリプトを使用して、Active IQ Unified Manager 用の Java REST クライアントを作成することもできます。サンプルコードに関して、以下の点にご注意ください。

- HTTPS接続を使用してActive IQ Unified Managerに接続し、指定されたREST URIを呼び出します
- Active IQ Unified Manager によって提供された証明書を無視します
- ハンドシェイク中にホスト名の検証をスキップする
- URI接続に `javax.net.ssl.HttpsURLConnection`を使用します
- サードパーティライブラリ（(`org.apache.commons.codec.binary.Base64`）を使用して、HTTPベーシック認証で使用されるBase64エンコード文字列を構築します

サンプル コードをコンパイルして実行するには、バージョン1.8以降のJavaコンパイラを使用する必要があります。

```
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;
    }

}
```

著作権に関する情報

Copyright © 2026 NetApp, Inc. All Rights Reserved. Printed in the U.S.このドキュメントは著作権によって保護されています。著作権所有者の書面による事前承諾がある場合を除き、画像媒体、電子媒体、および写真複写、記録媒体、テープ媒体、電子検索システムへの組み込みを含む機械媒体など、いかなる形式および方法による複製も禁止します。

ネットアップの著作物から派生したソフトウェアは、次に示す使用許諾条項および免責条項の対象となります。

このソフトウェアは、ネットアップによって「現状のまま」提供されています。ネットアップは明示的な保証、または商品性および特定目的に対する適合性の暗示的保証を含み、かつこれに限定されないいかなる暗示的な保証も行いません。ネットアップは、代替品または代替サービスの調達、使用不能、データ損失、利益損失、業務中断を含み、かつこれに限定されない、このソフトウェアの使用により生じたすべての直接的損害、間接的損害、偶発的損害、特別損害、懲罰的損害、必然的損害の発生に対して、損失の発生の可能性が通知されていたとしても、その発生理由、根拠とする責任論、契約の有無、厳格責任、不法行為（過失またはそうでない場合を含む）にかかわらず、一切の責任を負いません。

ネットアップは、ここに記載されているすべての製品に対する変更を随時、予告なく行う権利を保有します。ネットアップによる明示的な書面による合意がある場合を除き、ここに記載されている製品の使用により生じる責任および義務に対して、ネットアップは責任を負いません。この製品の使用または購入は、ネットアップの特許権、商標権、または他の知的所有権に基づくライセンスの供与とはみなされません。

このマニュアルに記載されている製品は、1つ以上の米国特許、その他の国の特許、および出願中の特許によって保護されている場合があります。

権利の制限について：政府による使用、複製、開示は、DFARS 252.227-7013（2014年2月）およびFAR 5252.227-19（2007年12月）のRights in Technical Data -Noncommercial Items（技術データ - 非商用品目に関する諸権利）条項の(b)(3)項、に規定された制限が適用されます。

本書に含まれるデータは商用製品および / または商用サービス（FAR 2.101の定義に基づく）に関係し、データの所有権はNetApp, Inc.にあります。本契約に基づき提供されるすべてのネットアップの技術データおよびコンピュータ ソフトウェアは、商用目的であり、私費のみで開発されたものです。米国政府は本データに対し、非独占的かつ移転およびサブライセンス不可で、全世界を対象とする取り消し不能の制限付き使用権を有し、本データの提供の根拠となった米国政府契約に関連し、当該契約の裏付けとする場合にのみ本データを使用できます。前述の場合を除き、NetApp, Inc.の書面による許可を事前に得ることなく、本データを使用、開示、転載、改変するほか、上演または展示することはできません。国防総省にかかる米国政府のデータ使用権については、DFARS 252.227-7015(b)項（2014年2月）で定められた権利のみが認められます。

商標に関する情報

NetApp、NetAppのロゴ、http://www.netapp.com/TMに記載されているマークは、NetApp, Inc.の商標です。その他の会社名と製品名は、それを所有する各社の商標である場合があります。