CHAPTER 3

Monitoring and Report Viewer Web サービスの使用方法

この章では、ACS 5.5 の Monitoring and Report Viewer コンポーネントで提供される Web サービスを使用するために設定する必要がある環境について説明します。以降、Viewer Web サービスと記載します。これらの Web サービスを使用して、ACS イベントのトラッキングおよびトラブルシューティングを行うカスタム アプリケーションを作成できます。

Viewer Web サービスは、次のメソッドで構成されています。

- getVersion()：Monitoring and Report Viewer サーバのバージョンを返します。
- getAuthenticationStatusByDate()：日付別のユーザの認証ステータスを返します。
- getAuthenticationStatusByTimeUnit()：時間別のユーザの認証ステータスを返します。
- getFailureReasons()：失敗の理由のリストを返します。
- getRadiusAccounting()：RADIUS アカウンティング レコードのリストを返します。
- getAPIVersion()：Viewer Web サービスのバージョンを返します。

ACS CLI での Web インターフェイスのイネーブル化

Viewer Web サービスを使用する前に、ACS で Web インターフェイスをイネーブルにする必要があります。ACS で Web インターフェイスをイネーブルにするには、ACS CLI から次のように入力します。

```
acs config web-interface view enable
```

acs config web-interface コマンドの詳細については、次の URL を参照してください。

http://www.cisco.com/en/US/docs/net_mgmt/cisco_secure_access_control_system/5.5/command/reference/cli_app_a.html#wp1887278

ACS CLI からの Web インターフェイスのステータス表示

Web インターフェイスのステータスを表示するには、ACS CLI から次のように入力します。

```
show acs-config-web-interface
```

show acs-config-web-interface コマンドの詳細については、次の URL を参照してください。

http://www.cisco.com/en/US/docs/net_mgmt/cisco_secure_access_control_system/5.5/command/reference/cli_app_a.html#wp1890877

ここでは、Monitoring and Report Viewer Web サービスの使用方法について説明します。

Viewer Web サービスのメソッドについて

ここでは、Viewer Web サービスで使用できるメソッドについて説明します。

表 3-1 で、Viewer Web サービスで使用されるクラスについて説明します。

表 3-1　Viewer Web サービス クラスの情報

クラス	説明
ACSViewWebServices	クライアント アプリケーションで参照できるすべての Web サービスが含まれます。
UserContext	Monitoring and Report Viewer サーバがユーザの認証に使用する ACS ユーザ名とユーザ パスワードが含まれます。
AuthenticationParam	照会されたレコードと返されるレコードに基づいて、認証クエリー パラメータをカプセル化します。
AuthenticationStatus	ACS から受け取るクエリー出力の認証ステータス レコードが含まれます。
AccountingParam	照会されたレコードと返されるレコードに基づいて、アカウンティング クエリー パラメータをカプセル化します。
AccountingStatus	ACS から受け取るクエリー出力のアカウンティング ステータス レコードが含まれます。
AccountingDetail	ACS から受け取るクエリー出力を構成する属性値のリストが含まれます。
ACSViewNBException	Web サービスで問題が発生した場合に Monitoring and Report Viewer が表示するエラーが含まれます。

（注）　Monitoring and Report Viewer では、すべての Web サービス クラスが com.cisco.acsview.nbapi パッケージに含まれます。

バージョンの取得

入力パラメータ

userCtx：（必須）ユーザ コンテキスト オブジェクト

目的

getVersion メソッドを使用して、ACS サーバにインストールされている Monitoring and Report Viewer のバージョンを表示します。このコマンドを CLI で入力して Web サービスを呼び出すと、Monitoring and Report Viewer のバージョンを表示できます。

出力パラメータ

Monitoring and Report Viewer サーバのバージョン。

例外

このメソッドは、次の場合にエラーを表示します。

- ユーザが無効
- 入力が無効
- ACS インスタンスが Monitoring and Report Viewer サーバとして実行されていない

日付別認証ステータスの取得

入力パラメータ

- userCtx：（必須）ユーザ コンテキスト オブジェクト
- authParam：（必須）AuthenticationParam オブジェクト
- startDate：（必須）認証ステータスを取得する日付範囲の開始日付
- endDate：（必須）認証ステータスを取得する日付範囲の終了日付

目的

getAuthenticationStatusByDate メソッドを使用して、指定された期間のユーザの認証ステータスを日付の古い方から順に表示します。

出力パラメータ

時間順に並べられた、指定された期間のユーザの認証ステータス。

例外

このメソッドは、次の場合にエラーを表示します。

- ユーザ コンテキストの値が入力されたが、ヌルだった
- ユーザ名およびパスワードが入力されたが、ヌルだった
- 日付の値が入力されたが、ヌルだった

時間単位ごとの認証ステータスの取得

入力パラメータ

- userCtx：（必須）ユーザ コンテキスト オブジェクト
- authParam：（必須）AuthenticationParam オブジェクト
- lastX：（必須）認証ステータスを取得する時間範囲の終了時刻
- timeUnit：（必須）分、時間、または日数で指定した時間単位

目的

getAuthenticationStatusByTimeUnit メソッドを使用して、指定された期間のユーザの認証ステータスを時刻の古い方から順に表示します。

出力パラメータ
時間順に並べられた、指定された期間のユーザの認証ステータス リスト。

例外
このメソッドは、次の場合にエラーを表示します。

- ユーザ コンテキストの値が入力されたが、ヌルだった
- ユーザ名およびパスワードが入力されたが、ヌルだった
- 日付の値が入力されたが、ヌルだった

障害理由の取得

入力パラメータ
userCtx：（必須）ユーザ コンテキスト オブジェクト

目的
getFailureReasons メソッドを使用して、失敗の理由が含まれているレコードのリストを取得します。

出力パラメータ
障害理由が含まれているレコードのリスト。

例外
このメソッドは、クレデンシャルが無効な場合にエラーを表示します。

RADIUS アカウンティングの取得

入力パラメータ

- userCtx：（必須）ユーザ コンテキスト オブジェクト
- acctParam：（必須）アカウンティング検索パラメータ。有効な matchOperator の値は、valueLIKE、valueEQ、valueNE、valueGE、valueLE、valueGT、valueLT、attrEQ、valueIN、valueINNOT。式の形式は、次のいずれかになります。
  - AttributeName, MatchArgument, MatchOp=[valueLIKE | valueEQ | valueNE | valueGE | valueLE | valueGT | valueLT | attrEQ]
  - AttributeName, MultipleValueMatchArgument, MatchOp=[valueIN | valueINNOT]

  Attribute Name：標準 RADIUS/Cisco A-V ペア名で定義されているとおり。属性名では大文字と小文字が区別されません。ただし、値では大文字と小文字が区別されます。

  valueLIKE：ワイルドカードによる一致（%）を検索。%foo% など。

  valueEQ：完全一致を検索。

  valueNE：「等しくない」の比較を実行。

  valueGE：「以上」の比較を実行。

  valueLE：「以下」の比較を実行。

  valueGT：「よりも大きい」の比較を実行。

  valueLT：「よりも小さい」の比較を実行。

attrEQ：指定された属性と別の属性を比較し、true または false を返す。

valueIN：matchOperator が valueIN の場合、複数の値を許可する。

valueINNOT：matchOperator が valueINNOT の場合、複数の値を許可しない。

- returnAttributes：（必須）要求された属性の戻りリスト。
- startDate：（必須）RADIUS アカウンティング レコードを取得する日付範囲の開始日付。
- endDate：（必須）RADIUS アカウンティング レコードを取得する日付範囲の終了日付。

目的

getRADIUSAccounting メソッドを使用して、RADIUS アカウンティング レコードのリストを取得します。

出力パラメータ

RADIUS アカウンティング レコードのリスト。

例外

このメソッドは、次の場合にエラーを表示します。

- ユーザ クレデンシャルが無効
- acctParam パラメータの matchOperator に無効な値が含まれている
- acctParam パラメータの matchValues に無効な値が含まれている
- データベース選択エラーが発生

API バージョンの取得

入力パラメータ

userCtx：（必須）ユーザ コンテキスト オブジェクト

目的

getAPIVersion メソッドを使用して、Viewer Web サービスのバージョンを取得します。

出力パラメータ

Viewer Web サービスのバージョン。

例外

このメソッドは、認証失敗が発生した場合にエラーを表示します。

WSDL ファイルについて

ここでは、WSDL ファイル、WSDL ファイルをダウンロードできる場所、クラス ファイル、および Viewer Web サービスで使用できるクエリーについて説明します。この項の構成は、次のとおりです。

WSDL ファイルのダウンロード

WSDL ファイルは、次の場所からダウンロードできます。

https://*ip address or hostname*/ACSViewWebServices/ACSViewWebServices?wsdl。ここで、*ip address* or *hostname* は、ACS サーバの IP アドレスまたはホスト名です。

Viewer の WSDL ファイル

WSDL は、Web サービス、サービスの場所、およびサービスが公開する操作を記述する XML ドキュメントです。

```
<definitions name="ACSViewWebServicesService"
targetNamespace="http://nbapi.acsview.cisco.com/jaws"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://nbapi.acsview.cisco.com/jaws"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <types>
  <schema elementFormDefault="qualified"
targetNamespace="http://nbapi.acsview.cisco.com/jaws"
xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:soap11-enc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://nbapi.acsview.cisco.com/jaws"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <complexType name="getFailureReasons">
    <sequence>
     <element name="userCtx" nillable="true" type="tns:UserContext"/>
    </sequence>
   </complexType>
   <complexType name="getAuthenticationStatusByDate">
    <sequence>
     <element name="userCtx" nillable="true" type="tns:UserContext"/>
     <element name="authParam" nillable="true" type="tns:AuthenticationParam"/>
     <element name="startDate" nillable="true" type="dateTime"/>
     <element name="endDate" nillable="true" type="dateTime"/>
    </sequence>
   </complexType>
   <complexType name="getAuthenticationStatusByDateResponse">
    <sequence>
     <element maxOccurs="unbounded" minOccurs="0" name="result" nillable="true"
type="tns:AuthenticationStatus"/>
    </sequence>
   </complexType>
   <complexType name="getAuthenticationStatusByTimeUnit">
    <sequence>
     <element name="userCtx" nillable="true" type="tns:UserContext"/>
     <element name="authParam1" nillable="true" type="tns:AuthenticationParam"/>
     <element name="lastX" type="int"/>
     <element name="timeUnit" nillable="true" type="string"/>
    </sequence>
   </complexType>
   <complexType name="getVersion">
    <sequence>
     <element name="userCtx" nillable="true" type="tns:UserContext"/>
    </sequence>
   </complexType>
   <complexType name="ACSViewNBException">
    <sequence>
     <element name="message" nillable="true" type="string"/>
    </sequence>
```

```
   </complexType>
   <complexType name="FailureReason">
    <sequence>
     <element name="authenFailureCode" nillable="true" type="string"/>
     <element name="possibleRootCause" nillable="true" type="string"/>
     <element name="resolution" nillable="true" type="string"/>
    </sequence>
   </complexType>
   <complexType name="AuthenticationParam">
    <sequence>
     <element name="AAAClient" nillable="true" type="string"/>
     <element name="clientIPAddress" nillable="true" type="string"/>
     <element name="clientMACAddress" nillable="true" type="string"/>
     <element name="userName" nillable="true" type="string"/>
    </sequence>
   </complexType>
   <complexType name="AuthenticationStatus">
    <sequence>
     <element name="authStatus" nillable="true" type="string"/>
     <element name="date" nillable="true" type="dateTime"/>
     <element name="errorCode" nillable="true" type="string"/>
     <element maxOccurs="unbounded" minOccurs="0" name="moreDetails" nillable="true"
type="string"/>
    </sequence>
   </complexType>
   <complexType name="getAuthenticationStatusByTimeUnitResponse">
    <sequence>
     <element maxOccurs="unbounded" minOccurs="0" name="result" nillable="true"
type="tns:AuthenticationStatus"/>
    </sequence>
   </complexType>
   <complexType name="getVersionResponse">
    <sequence>
     <element name="result" nillable="true" type="string"/>
    </sequence>
   </complexType>
   <complexType name="getFailureReasonsResponse">
    <sequence>
     <element maxOccurs="unbounded" minOccurs="0" name="result" nillable="true"
type="tns:FailureReason"/>
    </sequence>
   </complexType>
   <complexType name="UserContext">
    <sequence>
     <element name="password" nillable="true" type="string"/>
     <element name="userName" nillable="true" type="string"/>
    </sequence>
   </complexType>
   <element name="getAuthenticationStatusByDate"
type="tns:getAuthenticationStatusByDate"/>
   <element name="getAuthenticationStatusByDateResponse"
type="tns:getAuthenticationStatusByDateResponse"/>
   <element name="getAuthenticationStatusByTimeUnit"
type="tns:getAuthenticationStatusByTimeUnit"/>
   <element name="getAuthenticationStatusByTimeUnitResponse"
type="tns:getAuthenticationStatusByTimeUnitResponse"/>
   <element name="getVersion" type="tns:getVersion"/>
   <element name="ACSViewNBException" type="tns:ACSViewNBException"/>
   <element name="getVersionResponse" type="tns:getVersionResponse"/>
   <element name="getFailureReasons" type="tns:getFailureReasons"/>
   <element name="getFailureReasonsResponse" type="tns:getFailureReasonsResponse"/>
  </schema>
 </types>
 <message name="ACSViewNBException">
```

```
 <part element="tns:ACSViewNBException" name="ACSViewNBException"/>
</message>
<message name="ACSViewWebServices_getAuthenticationStatusByDate">
 <part element="tns:getAuthenticationStatusByDate" name="parameters"/>
</message>
<message name="ACSViewWebServices_getAuthenticationStatusByTimeUnitResponse">
 <part element="tns:getAuthenticationStatusByTimeUnitResponse" name="result"/>
</message>
<message name="ACSViewWebServices_getAuthenticationStatusByDateResponse">
 <part element="tns:getAuthenticationStatusByDateResponse" name="result"/>
</message>
<message name="ACSViewWebServices_getVersionResponse">
 <part element="tns:getVersionResponse" name="result"/>
</message>
<message name="ACSViewWebServices_getAuthenticationStatusByTimeUnit">
 <part element="tns:getAuthenticationStatusByTimeUnit" name="parameters"/>
</message>
<message name="ACSViewWebServices_getVersion">
 <part element="tns:getVersion" name="parameters"/>
</message>
<message name="ACSViewWebServices_getFailureReasons">
 <part element="tns:getFailureReasons" name="parameters"/>
</message>
<message name="ACSViewWebServices_getFailureReasonsResponse">
 <part element="tns:getFailureReasonsResponse" name="result"/>
</message>
<portType name="ACSViewWebServices">
 <operation name="getAuthenticationStatusByDate">
  <input message="tns:ACSViewWebServices_getAuthenticationStatusByDate"/>
  <output message="tns:ACSViewWebServices_getAuthenticationStatusByDateResponse"/>
  <fault message="tns:ACSViewNBException" name="ACSViewNBException"/>
 </operation>
 <operation name="getAuthenticationStatusByTimeUnit">
  <input message="tns:ACSViewWebServices_getAuthenticationStatusByTimeUnit"/>
  <output message="tns:ACSViewWebServices_getAuthenticationStatusByTimeUnitResponse"/>
  <fault message="tns:ACSViewNBException" name="ACSViewNBException"/>
 </operation>
 <operation name="getVersion">
  <input message="tns:ACSViewWebServices_getVersion"/>
  <output message="tns:ACSViewWebServices_getVersionResponse"/>
  <fault message="tns:ACSViewNBException" name="ACSViewNBException"/>
 </operation>
 <operation name="getFailureReasons">
  <input message="tns:ACSViewWebServices_getFailureReasons"/>
  <output message="tns:ACSViewWebServices_getFailureReasonsResponse"/>
  <fault message="tns:ACSViewNBException" name="ACSViewNBException"/>
 </operation>
</portType>
<binding name="ACSViewWebServicesBinding" type="tns:ACSViewWebServices">
 <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
 <operation name="getAuthenticationStatusByDate">
  <soap:operation soapAction=""/>
  <input>
   <soap:body use="literal"/>
  </input>
  <output>
   <soap:body use="literal"/>
  </output>
  <fault name="ACSViewNBException">
   <soap:fault name="ACSViewNBException" use="literal"/>
  </fault>
 </operation>
 <operation name="getAuthenticationStatusByTimeUnit">
  <soap:operation soapAction=""/>
```

```
   <input>
    <soap:body use="literal"/>
   </input>
   <output>
    <soap:body use="literal"/>
   </output>
   <fault name="ACSViewNBException">
    <soap:fault name="ACSViewNBException" use="literal"/>
   </fault>
  </operation>
  <operation name="getVersion">
   <soap:operation soapAction=""/>
   <input>
    <soap:body use="literal"/>
   </input>
   <output>
    <soap:body use="literal"/>
   </output>
   <fault name="ACSViewNBException">
    <soap:fault name="ACSViewNBException" use="literal"/>
   </fault>
  </operation>
  <operation name="getFailureReasons">
   <soap:operation soapAction=""/>
   <input>
    <soap:body use="literal"/>
   </input>
   <output>
    <soap:body use="literal"/>
   </output>
   <fault name="ACSViewNBException">
    <soap:fault name="ACSViewNBException" use="literal"/>
   </fault>
  </operation>
 </binding>
 <service name="ACSViewWebServicesService">
  <port binding="tns:ACSViewWebServicesBinding" name="ACSViewWebServices">
   <soap:address location="http://localhost:8080/ACSViewWebServices/ACSViewWebServices"/>
  </port>
 </service>
</definitions>
```

Viewer Web サービスとアプリケーションの統合

ここでは、Viewer Web サービスとアプリケーションを統合する方法について説明します。

コードを Viewer Web サービスと統合し、Web サービスの起動後に応答を受け取るようにするには、次の手順に従います。

ステップ 1　サーバから証明書を取得し、クライアント証明書を作成します。

a. 次の URL から、展開されている Web サービスを確認します。

https://*ip address or hostname*/ACSViewWebServices/ACSViewWebServices?wsdl

Web サービスの詳細については、「Viewer Web サービスのメソッドについて」（P.3-2）を参照してください。

b. [View Certificate] をクリックし、[Details] タブに移動します。

c. [Copy to File] をクリックします。

d. [Welcome] ウィンドウで、[Next] をクリックします。

e. [Export File Format] ウィンドウで、[DER encoded binary X.509(.CER)] を選択し、[Next] をクリックします。

f. [File to Export] ウィンドウで、ファイル名を入力し、[Next] をクリックします。

g. [Completing the Certificate Export Wizard] ウィンドウで、[Finish] をクリックします。

証明書が *server.cer* としてローカル システムに保存されます。

h. 次のコマンドを使用して、サーバ証明書をインポートし、*client.ks*（クライアント証明書）として保存します。

```
keytool -import -file server.cer -keystore client.ks
```

ステップ 2　次の URL から、展開されている Viewer Web サービスを確認します。

https://*IPaddress*（*or*）*HostName*/ACSViewWebServices/ACSViewWebServices?wsdl

Web サービスの詳細については、「Viewer Web サービスのメソッドについて」（P.3-2）を参照してください。

ステップ 3　次のコマンドを使用して、ソースを表示し、ローカル システムに WSDL ファイルをコピーします。

soap:address location='https://*acsview-cars1:443*/ACSViewWebServices/ACSViewWebServices'/

WSDL ファイルの詳細については、「WSDL ファイルについて」（P.3-5）を参照してください。

ステップ 4　Sun Microsystems の Web サイトから JAX-WS 2.0 ライブラリをダウンロードします。

ステップ 5　アーティファクトに関連する情報を表示するために、次の場所で **wsimport -keep** コマンドを入力します。https://*IPAddress:443*/ACSViewWebServ/ACSViewWebServices?wsdl

ローカル側にすべてのライブラリを格納します。

ステップ 6　クライアント コードを作成します。

ステップ 7　クライアント コードをコンパイルして実行します。

Viewer Web サービスの操作

ここでは、Java のクライアント コードの例を示します。ここで説明する要件は、クライアント側の変換ツールとして Java を使用する場合にのみ適用されます。この項の構成は、次のとおりです。

必要なファイル

Java（JAX-WS）2.0 をクライアント側の変換ツールとして使用するには、次の *JAR* ファイルが必要です。*.jar* ファイルおよび関連ツールは、Sun Microsystems の Web サイトからダウンロードできます。

- activation.jar
- FastInfoset.jar
- http.jar

- jaxb-api.jar
- jaxb-impl.jar
- jaxb-xjc.jar
- jaxws-api.jar
- jaxws-rt.jar
- jaxws-tools.jar
- jsr173_api.jar
- jsr181-api.jar
- jsr250-api.jar
- resolver.jar
- saaj-api.jar
- saaj-impl.jar
- sjsxp.jar

サポートされる SOAP クライアント

サポートされる SOAP クライアントは、以下のとおりです。

- Apache
- JAX-WS

Viewer Web サービスへの接続

Viewer Web サービスに接続するには、次の手順に従います。

ステップ 1　次の URL から、展開されている Viewer Web サービスを確認します。

https://*ip address or hostname*/ACSViewWebServices/ACSViewWebServices?wsdl

Web サービスの詳細については、「Viewer Web サービスのメソッドについて」（P.3-2）を参照してください。

ステップ 2　右クリックして [View Source/View Page Source] オプションを選択し、ソース情報を表示します。

ソース情報はポップアップ ダイアログボックスに表示されます。

ステップ 3　ソース情報をローカル ディレクトリ <SERVICE_HOME> に、ACSViewWebServices.wsdl という名前で保存します。

ステップ 4　次のコマンドを実行して、クラス ファイルを作成します。

```
wsimport <SERVICE_HOME>/ACSViewWebServices.wsdl -d <SERVICE_HOME>
```

ステップ 5　「クライアント コードの例」（P.3-12）を**コピー**して、Client.java として <SERVICE_HOME> に保存し、次のコマンドを使用してコンパイルします。

```
javac -cp <SERVICE_HOME> <SERVICE_HOME>/Client.java -d <SERVICE_HOME>
```

これでクライアント コードがコンパイルされ、<SERVICE_HOME> ディレクトリにパッケージが保存されます。

ステップ 6　クライアント コードを実行するには、次のコマンドを実行します。

```
java -cp <SERVICE_HOME> com.cisco.acsview.nbapi.jaws.Client.
```

（注）　上記の手順は、Java 1.6.0_25 で実行します。JAVA_HOME は java のインストール ディレクトリで、「path」環境変数に値 <JAVA_HOME>/bin を追加する必要があります。

クライアント コードの例

ここでは、Viewer Web サービスのクライアント コードの例を示します。

```
package com.cisco.acsview.nbapi.jaws;

import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.ArrayList;
import java.util.List;
import java.util.Iterator;
import com.sun.org.apache.xerces.internal.jaxp.datatype.XMLGregorianCalendarImpl;
import javax.xml.datatype.XMLGregorianCalendar;
import javax.xml.datatype.DatatypeFactory;
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;

public class Client
{
private static void install() throws Exception
{
// Create a trust manager that does not validate certificate chains
TrustManager[] trustAllCerts = new TrustManager[]
{
new X509TrustManager()
{
public X509Certificate[] getAcceptedIssuers()
{
return null;
}

public void checkClientTrusted(X509Certificate[] certs, String authType)
{
// Trust always
}

public void checkServerTrusted(X509Certificate[] certs, String authType)
{
// Trust always
}
}
};
```

```
// Install the all-trusting trust manager
SSLContext sc = SSLContext.getInstance("SSL");
// Create empty HostnameVerifier
HostnameVerifier hv = new HostnameVerifier()
{
public boolean verify(String arg0, SSLSession arg1)
{
return true;
}
};

sc.init(null, trustAllCerts, new java.security.SecureRandom());
HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());
HttpsURLConnection.setDefaultHostnameVerifier(hv);
}

public static void install1() throws Exception
{
// Bypass hostname verification.
HttpsURLConnection.setDefaultHostnameVerifier(
new HostnameVerifier()
{
public boolean verify(String arg0, SSLSession arg1)
{
return true;
}
});
}

public static void main(String args[])
{
try
{
install();
ACSViewWebServicesService serviceObj = new ACSViewWebServicesService();
ACSViewWebServices service = serviceObj.getACSViewWebServices();
UserContext userCtx = new UserContext();
userCtx.setUserName("acsadmin");
userCtx.setPassword("Acs5.5");
getVersion(service,userCtx);
getAPIVersion(service,userCtx);
getAuthBydate(service,userCtx);
getAuthByTime(service,userCtx);
getRadiusAccounting(service,userCtx);
getFailureReasons(service,userCtx);
}
catch (Exception ex)
{
ex.printStackTrace();
}
}


/**
* getVersion provide the application version
*/
public static void getVersion(ACSViewWebServices service, UserContext userCtx)
{
try
{
String result = service.getVersion(userCtx);
System.out.println("-------------------------*** Application Version
***-------------------------"+"\n");
System.out.println("Application Version : "+result);
```

```
System.out.println("------------------------------------------------------------------------
-------"+"\n");
}
catch(Exception e)
{
e.printStackTrace();
}
}
/**
*getAuthByDate provides the data of the authentication success/failure between
the specified date range
*/
private static void getAuthBydate(ACSViewWebServices service, UserContext userCtx)
{
try
{
System.out.println("-------------------------*** Authentication Status by
Date Starts ***-------------------------"+"\n");
AuthenticationParam authParam = new AuthenticationParam();
/**
*** The following Attributes are optional.
** If the parameters are not set, method will return all the authentications
success/failure between the specified date range.
** The Data will be filtered based on the attribute set which is falling
under the specified date range.
** The attributes set are exactly matched for filtering,ie., only the data
which is matching the below attributes
   and with in the specified date range are retrived.
*/
authParam.setAAAClient("MyClient");
authParam.setClientIPAddress("10.77.241.203");
authParam.setClientMACAddress("ABAC00019E05");
authParam.setUserName("user1");
/******* Optional Attributes Ends **************/
DatatypeFactory datatypeFactory = DatatypeFactory.newInstance();
GregorianCalendar gc1 = newGregorianCalendar(2011, Calendar.AUGUST, 4);
XMLGregorianCalendar startDate =
datatypeFactory.newXMLGregorianCalendar(gc1).normalize();
GregorianCalendar gc2 = newGregorianCalendar(2011, Calendar.AUGUST, 6);
XMLGregorianCalendar endDate =
datatypeFactory.newXMLGregorianCalendar(gc2).normalize();
java.util.List authStatusArray =
service.getAuthenticationStatusByDate(userCtx,authParam, startDate, endDate);
System.out.println("No of Records Retrieved : "+authStatusArray.size());
for(int i=0; i<authStatusArray.size();i++)
{
System.out.println("*************** Authentication Status : "+(i+1)+"
***************");
AuthenticationStatus status = (AuthenticationStatus)authStatusArray.get(i);
java.util.List sarray = status.getMoreDetails();
System.out.println(sarray.get(0) +" :: "+sarray.get(1));
for(int j=0;j<sarray.size();j++)
{
System.out.println(sarray.get(j)+" :: "+sarray.get(++j));
}

System.out.println("*********************************************************************");
}
System.out.println("-------------------------*** Authentication Status by
Date Ends ***-------------------------"+"\n");
}
catch (Exception ex)
{
```

```
ex.printStackTrace();
}
}

/**
* getAuthByTime provides the data of the authentication success/failure in the
specified time.
* Time can be provided in Minutes, Hours or Days
*/
private static void getAuthByTime(ACSViewWebServices service, UserContext userCtx)
{
try
{
System.out.println("-------------------------*** Authentication Status by
Time Starts ***-------------------------"+"\n");
AuthenticationParam authParam = new AuthenticationParam();
/**
*** The following Attributes are optional.
** If the parameters are not set method will return all the authentications
success/failure between the specified date range.
** The Data will be filtered based on the attribute set which is falling
under the specified date range.
** The attributes set are exactly matched for filtering,ie., only the data
which is matching the below attributes
   and with in the specified date range are retrived.
*/
authParam.setAAAClient("MyClient");
authParam.setClientIPAddress("10.77.241.203");
authParam.setClientMACAddress("ABAC00019E05");
authParam.setUserName("user1");
/******* Optional Attributes Ends **************/
java.util.List authStatusArray =
service.getAuthenticationStatusByTimeUnit(userCtx,authParam, 20, "Hours");
System.out.println("No of Records Retrieved : " + authStatusArray.size());
for(int i=0; i<authStatusArray.size();i++)
{
System.out.println("*************** Authentication Status : "+(i+1)+"
***************");
AuthenticationStatus status = (AuthenticationStatus)authStatusArray.get(i);
java.util.List sarray = status.getMoreDetails();
System.out.println(sarray.get(0) +" :: "+sarray.get(1));
for(int j=0;j<sarray.size();j++)
{
System.out.println(sarray.get(j)+" :: "+sarray.get(++j));
}

System.out.println("*****************************************************************");
}
System.out.println("-------------------------*** Authentication Status by
Time Ends ***-------------------------"+"\n");
}
catch (Exception ex)
{
ex.printStackTrace();
}
}

/**
** getAPIVersion provides the application API Version
*/
public static void getAPIVersion(ACSViewWebServices service, UserContext userCtx)
{
try
{
```

```
System.out.println("-------------------------*** API Version
***-------------------------"+"\n");
String apiresult = service.getAPIVersion(userCtx);
System.out.println("API Version : "+apiresult);

System.out.println("-------------------------------------------------------------------------
-------"+"\n");
}
catch(Exception ex)
{
ex.printStackTrace();
}
}

/**
** getFailureReasons provide the Failure Code, Possible Root Cause and Resolution
*/
public static void getFailureReasons(ACSViewWebServices service, UserContext
userCtx)
{
try
{
// Get Failure reason - Example
System.out.println("-------------------------*** Failure Reasons Starts
***-------------------------"+"\n");
List result1 = service.getFailureReasons(userCtx);
System.out.println("Failure reasons list is : " + result1.size());
for (int i=0;i<result1.size() ;i++ )
{
System.out.println("Authentication Failure Code :
"+((FailureReason)result1.get(i)).getAuthenFailureCode());
System.out.println("Possible Root Cause :
"+((FailureReason)result1.get(i)).getPossibleRootCause());
System.out.println("Resolution :
"+((FailureReason)result1.get(i)).getResolution());
}
System.out.println("-------------------------*** Failure Reasons Ends
***-------------------------"+"\n");
}
catch(Exception ex)
{
ex.printStackTrace();
}
}

/**
** getRadiusAccounting provides the accounting details between the specified
date range.
*/
public static void getRadiusAccounting(ACSViewWebServices service, UserContext
userCtx)
{
try
{
System.out.println("-------------------------*** Radius Accounting Starts
***-------------------------"+"\n");
List acctParam = new ArrayList();
AccountingParam acParam = new AccountingParam();
List valList = acParam.getMatchValues();
valList.add("11");
acParam.setAttributeName("cisco-h323-disconnect-cause/h323-disconnect-cause");
acParam.setMatchOperator("valueINNOT");
acctParam.add(acParam);
List returnAttributes = new ArrayList();
```

```
returnAttributes.add("cisco-h323-disconnect-cause/h323-disconnect-cause");
DatatypeFactory datatypeFactory = DatatypeFactory.newInstance();
GregorianCalendar gc1 = newGregorianCalendar(2011, Calendar.AUGUST, 5);
XMLGregorianCalendar startDate =
datatypeFactory.newXMLGregorianCalendar(gc1).normalize();
GregorianCalendar gc2 = newGregorianCalendar(2011, Calendar.AUGUST, 7);
XMLGregorianCalendar endDate =
datatypeFactory.newXMLGregorianCalendar(gc2).normalize();
AccountingStatus acctStatus = service.getRadiusAccounting(userCtx,acctParam,
startDate, endDate, returnAttributes);
List attrNames = acctStatus.getAttrNames();
for(int x=0 ; x<attrNames.size() ; x++)
{
System.out.println("Attribute Names : "+attrNames.get(x));
}
List acctDetailsList = (ArrayList)acctStatus.getAcctDetails();
Iterator detailIterator = acctDetailsList.iterator();
while(detailIterator.hasNext())
{
AccountingDetail acctDetailObj = (AccountingDetail)detailIterator.next();
List acctDetails = (List)acctDetailObj.getAttrValues();
for (int i=0;i<acctDetails.size() ;i++ )
{
System.out.println("Attribute Details : "+acctDetails.get(i));
}
}
System.out.println("-------------------------*** Radius Accounting Ends
***-------------------------"+"\n");
}
catch(Exception e)
{
e.printStackTrace();
}
}
}
```