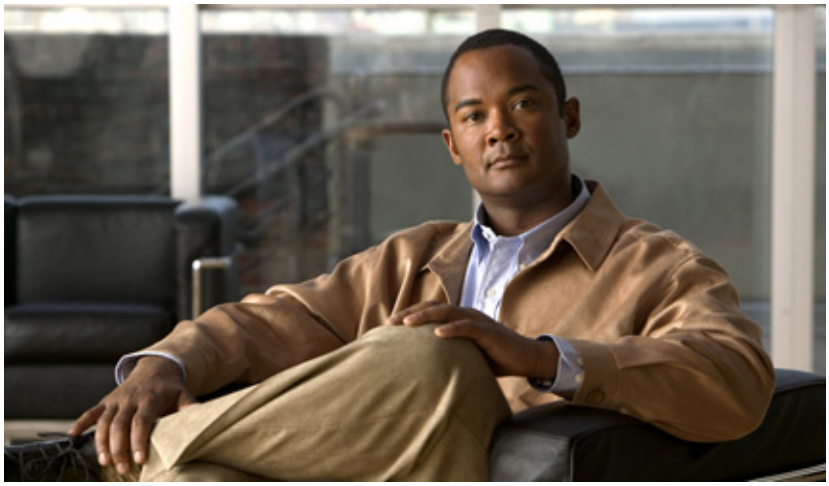

NB-API SDK for Cisco Secure Access Control Server View

Release 4.0

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Text Part Number:

NB-API SDK for Cisco Secure Access Control Server View

CONTENTS

Preface

This document describes the ACS View northbound API and provides instructions for integrating it with the client API's.

Audience

This document is for the experienced network administrator with expertise in managing networks for a company.

Conventions

This document uses the following conventions:

Item	Convention
Commands and keywords	**boldface** font
Variables for which you supply values	*italic* font
Displayed session and system information	`screen` font
Information you enter	**`boldface screen`** `font`
Variables you enter	*`italic screen`* font
Menu items and button names	**boldface** font
Selecting a menu item in paragraphs	**Option > Network Preferences**
Selecting a menu item in tables	Option > Network Preferences

Note Means *reader take note*. Notes contain helpful suggestions or references to material not covered in the publication.

Caution Means *reader be careful*. In this situation, you might do something that could result in equipment damage or loss of data.

EDIT REVIEW DRAFT—CISCO CONFIDENTIAL

Warning **This symbol means danger. You are in a situation that could cause bodily injury.**

Product Documentation

Note We sometimes update the printed and electronic documentation after original publication. Therefore, you should also review the documentation on Cisco.com for any updates.

Table 1 describes the product documentation that is available.

Table 1 Product Documentation

Document Title	Available Formats
Release Notes for ACS View 4.0	• Printed document that was included with the product. • On Cisco.com at this URL: TBD
Quick Start Guide for ACS View 4.0	• Printed document that was included with the product. • PDF on the product CD-ROM. • On Cisco.com at this URL: TBD
Installation [and Configuration Guide] for ACS View 4.0	• PDF on the product CD-ROM. • On Cisco.com at this URL: TBD • Printed document available by order (part number DOC-nnnnnnn=).[1]
User Guide for ACS View 4.0	• PDF on the product CD-ROM. • On Cisco.com at this URL: TBD • Printed document available by order (part number DOC-nnnnnnn=).
Regulatory Compliance and Safety Information for ACS View 4.0	• Printed document that was included with the product. • PDF on the product CD-ROM. • On Cisco.com at this URL: TBD
Context-sensitive online help	• Select an option from the navigation tree, then click **Help**. • Click the Help button in the dialog box.

1.

Obtaining Documentation, Obtaining Support, and Security Guidelines

For information on obtaining documentation, obtaining support, providing documentation feedback, security guidelines, and also recommended aliases and general Cisco documents, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

CHAPTER 1

Overview

The Cisco® Secure Access Control Server (ACS) View provides a major evolutionary change and enhancement for Access Control Server's (ACS) reporting capabilities.

ACS View focuses on providing a rich set of reports and alerts, leveraging log and configuration data collected from ACS 4.x servers, using available mechanisms in ACS, namely Syslog and *package.cab* downloaded using HTTPS.

Cisco Secure ACS View provides a set of NB (northbound) APIs that allow programmatic access to certain key features and functions. These APIs, which can integrate legacy or custom management solutions with different network management applications, vendors, OS platforms, and programming languages, allow Cisco Secure ACS View customers to program their client APIs using the vendor and platform of their choice.

Cisco Secure ACS View NB-API uses XML and SOAP encoding technology with HTTPS as the underlying transport mechanism for the northbound services. The northbound services are described using the WSDL (Web Service Definition Language), which is also exported to the client. The NB-APIs use the JAX-WS Web Services paradigm and Jboss WS web services to use client applications.

The following sections provide a high-level description of these components:

- Understanding SOAP
- Understanding WSDL
- Understanding the Cisco Secure ACS View Server
- Understanding the ACS View NB-API Elements
- Client System and Server Requirements
- Required .jar files

Understanding SOAP

SOAP (Simple Object Access Protocol) provides a simple, lightweight mechanism for exchanging structured and typed information between peers in a decentralized, distributed environment using XML. SOAP does not itself define any application semantics such as a programming model or implementation-specific semantics; rather, it defines a simple mechanism for expressing application semantics by providing a modular packaging model and

encoding mechanisms for encoding data within modules. This allows SOAP to be used in a large variety of systems ranging from messaging systems to RPC.

SOAP consists of three parts:

- The SOAP envelope construct defines an overall framework for expressing what is in a message, who should deal with it, and whether it is optional or mandatory.
- The SOAP encoding rules define a serialization mechanism that can be used to exchange instances of application-defined data types.
- The SOAP RPC representation defines a convention that can be used to represent remote procedure calls and responses.

Although these parts are described together as part of SOAP, they are functionally orthogonal. In particular, to promote simplicity through modularity, the envelope and the encoding rules are defined in different namespaces.

One of the design goals of SOAP is to encapsulate and exchange RPC calls using the extensibility and flexibility of XML. This section defines a uniform representation of remote procedure calls and responses. Using SOAP for RPC is orthogonal to the SOAP protocol binding. When using HTTPS as the protocol binding, an RPC call maps naturally to an HTTPS request and an RPC response maps to an HTTPS response. However, using SOAP for RPC is not limited to the HTTPS protocol binding.

To make a method call, the following information is required:

- The URI of the target object
- A method name
- An optional method signature
- The parameters to the method
- Optional header data

SOAP relies on the protocol binding to provide a mechanism for carrying the URI. For example, for HTTPS the request URI indicates the resource that the invocation is being made against. Other than requiring that it be a valid URI, SOAP places no restriction on the form of an address.

For more SOAP programming information and code samples, see the World Wide Web Consortium web site, http://www.w3.org/TR/SOAP/.

Understanding WSDL

The northbound services are described using WSDL (Web Services Definition Language). WSDL is an XML format that describes network services as a collection of network endpoints (ports) that operate on messages. WSDL is extensible to allow the description of endpoints and their messages regardless of what message formats or network protocols are used.

See Understanding the WSDL Files, page 3-1 for more detailed information on WSDL files.

Related Topics

The World Wide Web Consortium web site, http://www.w3.org/TR/wsdl, contains WSDL documentation, software downloads, and code samples.

Understanding the Cisco Secure ACS View Server

The primary function of the complete ACS View solution is:

- To provide administrators a consolidated product that would enable them to extract the core logging and troubleshooting/diagnostic information from ACS and correlate that data across ACS server(s).
- To provide advanced Reporting, Diagnostics and Troubleshooting for a single or multiple ACS server(s) deployed in the field.

Understanding the ACS View NB-API Elements

The ACS View NB-API layers contain the following elements that are involved to perform its northbound operations:

- WCS clients—This is the client application (you) who will use SOAP NB-API provided by ACS View.
- SOAP Engine—JBossWS will be used as the SOAP Engine for the NB-API.
- Authentication Check—Before you access the NB-API, an authentication check is done before serving the request.
- NB-API—The API which is implemented in ACS View through JAX-WS Web services will be shared and exposed to you.
- ACS View internal API—ACS-View internal API will read the data from database using the data access layer and will give the data to the NB-API whenever it requests. This API will give the populated values to the Web Services class which is exposed to you.

Client System and Server Requirements

You must check if you satisfy the following system and server requirements to be able to use the northbound API services of ACS View.

Component	Requirement
Hard Diskette Drive (HDD)	500 GB
System Memory	4 GB
I/O Ports	USB and Serial ports
Media	Appliance
Server Requirement	N/A - Shipped with Appliance
Client Operating System	• Windows XP • Windows Vista Business Edition

Component	Requirement
Java Application Servers	JBoss 4.0.5 GA or later version
	JRE 1.5 or later version

Required .jar files

You must have jax-ws 2.0 tools as the client side conversion tool as jax-ws 2.0 is supported by this NB-API.

You also need to have jdk1.5 support.

You can download the .jar files and the related tools from the following location:

https://jax-ws.dev.java.net/ri-download.html#2.0

The .jar files you are required to have are:

- 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

CHAPTER 2

System Flow

This chapter helps you to better understand the system workflow of ACS View NB-API.

EDIT REVIEW DRAFT—CISCO CONFIDENTIAL

Deploying Web Services

To deploy the web services, you must create a .war file with the existing services directory and then copy the .war file into the following directory:

```
JBOSS_HOME/server/default/deploy
```

When you start the JBoss it will deploy the web services and will be ready for your use provided you have the valid WSDL and credentials.

You can view the successfully deployed web services in the following location:

```
/jbossws/services
```

Note Jboss WS will be used as the SOAP/Engine.

Starting and Stopping Web Service

All the Web Services will be running under JBoss. Starting and stopping the JBoss process will start and stop all the services. There is no utility for stopping and starting any individual web services.

Creating WSDL from Java Class

Web Services Description Language (WSDL) serves to describe Web Services in a structured way. A WSDL description of a service describes, in a machine understandable way, the interface to the service, the data types it uses, and where the service is located.

To generate the WSDL file you must use the **wsprovide** command line as mentioned below:

```
wsprovide -o generated -w cisco.ACSViewServices
```

Authentication

NB-API framework will perform authentication (valid session) checks for all the NB-API calls.

If the session is not valid or if the authorization fails then the framework will not proceed. It will return the appropriate SOAP Fault to the NB-API caller.

Exception Handling

All the Exceptions thrown by the NB-API methods should use ACSViewNBException web service method.

For more detailed information on the web services, see Web Services Information, page 3-6.

UserContext and SourceContext Definition

To access the NB-API, you must have a valid session.

Note The login and logout mechanisms are controlled and taken care of by the individiual services shared in your system.

NB-API methods can directly access the JAX-WS SOAP Server specific code to access the session and get these details.

For more detailed information on the web services, see Web Services Information, page 3-6.

CHAPTER 3

Understanding the WSDL Files

In this chapter, you can better understand about the WSDL files, the class files and the available web services in this NB-API.

WSDL Files

WSDL stands for Web Services Description Language. WSDL is a document written in XML. The document describes a web service. It specifies the location of the service and the operations (or methods) the service exposes.

The WSDL files are listed below.

```
<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='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='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='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'/>
  </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>
 <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>
</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>
  </binding>
  <service name='ACSViewWebServicesService'>
   <port binding='tns:ACSViewWebServicesBinding' name='ACSViewWebServicesPort'>
    <soap:address
location='http://acs-view-cars1:8080/ACSViewWebServices/ACSViewWebServices'/>
   </port>
  </service>
</definitions>
```

Northbound Service Classes Information

ACSViewWebServices

This class contains all the web services that are exposed to the Client Applications.

AuthenticationParam

This class encapsulates the Authentication query parameters based on which the records will be queried and returned.

AuthenticationStatus

This class contains the Authentication Status record retrieved from the ACS View server.

SourceContext

This class contains details about the NB-API client like the Application name, version and the Hostname from which the NB-APIs are being accessed.

UserContext

This class contains the ACS View user name and the password using which the user will be authenticated with the ACS View server.

ACSViewNBException

This class contains the exception that will be thrown for any issues with the NB-APIs.

Note All the NB-API classes will be placed in the package **com.cisco.acsview.nbapi**.

Web Services Information

The following web services are available in this NB-API:

getVesion()

You can call this method to get the ACS View version information.

getAuthenticationStatusByDate()

This web service method will retun you the Authentication Status array objects that contains all the failed and passed attempts. UserContext, Startdate and endDate are the parameters that you are required to pass.

Following are the parameters in AuthenticationStatusByDate():

```
userCtx - UserContext object passed by the user
authParam - AuthenticationParam object passed by the user
startDate - Start Date to get the Authentication Status
endDate - End Date to get the Authentication Status
```

It throws the exception `com.cisco.acsview.nbapi.ACSViewNBException` if the following validations fail:

- If the userContext value is entered but passed as null
- If the username and password are entered but passed as null
- If the Date value is entered but passed as null.

getAuthenticationStatusByTimeUnit()

This web service method will return you with the AuthenticationStatus array objects that contains all the failed and passed attempts. You must pass the UserContext,AuthenticationParam,lasttime, and time unit parameters.

Following are the parameters in AuthenticationStatusByTimeUnit():

```
userCtx - UserContext object passed by the user
authParam1 - AuthenticationParam object passed by the user
lastX - last time to get the Atuthenication Status
timeUnit - Valid Time Unit values are 'Minutes', 'Hours', 'Days'
```

It throws the exception `com.cisco.acsview.nbapi.ACSViewNBException` if the following validations fail:

- If the userContext value is entered but passed as null
- If the username and password are entered but passed as null
- If the Date value is entered but passed as null.

Below are the name and value pairs that will appear in the List:

```
Name: NAS_IP_Address
Value = IP address of the AAA client that is requesting authentication.

Name:  Framed_IP
Value = IP address of the client.

Name: User_Name
Value = Name of the user being authenticated.

Name: Group_Name
Value = Network access group (the final mapped user group)
```

Name of the User Group to which you are added. If you do not have such a group, it automatically adds to the default group.

```
Name: Network_Device_Group
Value = The network device group to which the access device (AAA client) belongs.

Name: protocol
Value = The protocol used.
```

You can also get the values from the List and use it in the client code in the form:

```
_hMap.get(Network_Device_Group);
```

Queries Used in NB-API

The following is the query used for computing the result for the API:

```
select * from acs_log_failed_attempts  where loggedAt_ACSView between        \
'statdate' and 'enddate'
        where User_Name='User_Name'and NAS_IP_Address ='NAS-IP' and                  \
Framed_IP_Address='framed-ip' and Caller_ID='caller-id'
```

Note The above query is dynamic depending upon your input.

CHAPTER 4

NB-API Integration

This chapter will help you perform the ACS View NB-API integration successfully.

Performing the NB-API Integration

You must integrate your code with the NB-API to get response after you invoke the API's. This can be performed by following the simple steps listed here:

Step 1 You must first verify the deployed web services from the following location:
`http://<IPAddress>:8080/ACSViewWebServ/ACSViewWebServices?wsdl`

For more information on the web services, see Web Services Information, page 3-6

Step 2 Get the jax-ws2.0 libraries from the location https://jax-ws.dev.java.net/ri-download.html

Step 3 To get your related artifacts information, use the command **`wsimport -keep`** at the location:
`http://<IPAddress>:8080/ACSViewWebServ/ACSViewWebServices?wsdl`

Step 4 Include all the libraries in your location.

A sample client code integration method is listed here:

```
/*
 * Client.java
 *
 * Created on October 13, 2007, 8:44 PM
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package com.cisco.acsview.nbapi.jaws;

import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import com.cisco.acsview.nbapi.jaws.*;
import com.sun.org.apache.xerces.internal.jaxp.datatype.XMLGregorianCalendarImpl;
import javax.xml.datatype.DatatypeConstants;
import javax.xml.datatype.XMLGregorianCalendar;

/**
 *
 * @author nchilaka
 */
public class Client {
    public static void main(String args[]){
        try {
            ACSViewWebServicesService service = new ACSViewWebServicesService();
            ACSViewWebServices port = service.getACSViewWebServicesPort();
            // TODO initialize WS operation arguments here
            UserContext userCtx = new UserContext();
            userCtx.setUserName("admin");
```

```
            userCtx.setPassword("roZes123");
            // TODO process result here
            java.lang.String result = port.getVersion(userCtx);
            System.out.println("Version = "+result);

            //getAuths();
            getAuthBydate();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }


/**
```

For calling AuthenticationByDate Service:

```
*/

    private static void getAuthBydate(){
        try {

            System.out.println("#####Get Authentication by Date#######");

            ACSViewWebServicesService service = new ACSViewWebServicesService();
            ACSViewWebServices port = service.getACSViewWebServicesPort();
            // TODO initialize WS operation arguments here
            UserContext userCtx = new UserContext();
            userCtx.setUserName("nagesh");
            userCtx.setPassword("roZes123$");
            AuthenticationParam authParam = new AuthenticationParam();
            authParam.setUserName("User5");


            Calendar xmas = new GregorianCalendar(2005, Calendar.OCTOBER, 03);
            Date startDate = xmas.getTime();

            Calendar xmas1 = new GregorianCalendar(2007, Calendar.OCTOBER, 11);
            Date endDate = xmas1.getTime();

            XMLGregorianCalendar gc1 = new XMLGregorianCalendarImpl(new
GregorianCalendar(2005, Calendar.OCTOBER, 3));


            XMLGregorianCalendar gc2 = new XMLGregorianCalendarImpl(new
GregorianCalendar(2007, Calendar.OCTOBER, 19));
```

```
            java.util.List authStatusArray = port.getAuthenticationStatusByDate(userCtx,
authParam, gc1, gc2);
             System.out.println("Size of the List is" + authStatusArray.size());
             for(int i=0; i<authStatusArray.size();i++){
              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("###############ENDDDDDDD######################");
             }
         } catch (Exception ex) {
             ex.printStackTrace();
         }
     }

        private static void getAuths(){
         System.out.println("#####Get Authentication by Time#######");
         try {
             ACSViewWebServicesService service = new ACSViewWebServicesService();
             ACSViewWebServices port = service.getACSViewWebServicesPort();
             // TODO initialize WS operation arguments here
             UserContext userCtx = new UserContext();

             userCtx.setUserName("admin");
             userCtx.setPassword("roZes123");

             AuthenticationParam authParam1 = new AuthenticationParam();
             authParam1.setUserName("User5");
             int lastX = 0;
             java.lang.String timeUnit = "";
             // TODO process result here
             java.util.List  authStatusArray =
port.getAuthenticationStatusByTimeUnit(userCtx,authParam1, 1000, "Days");
             //AuthenticationStatusArray authArray =
port.getAuthenticationStatusByTimeUnit(userCtx,authParam1, 1000, "Days");
             //java.util.List  authStatusArray  = authArray.getValue();

             System.out.println("Size of the List is" + authStatusArray.size());
             for(int i=0; i<authStatusArray.size();i++){
              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("################ENDDDDDDD#######################");
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

}
```

INDEX