



RSA MCF Plug-in Implementation Guide

Last Modified: February 29, 2016

Partner Information

Product Information	
Partner Name	TeleSign
Web Site	http://www.telesign.com/
Product Name	Verify SMS RSA AAOP Multi-Credential Framework Plug-In
Version & Platform	1.3.0
Product Description	TeleSign Verify SMS authentication service verifies users in real time by sending a one-time verification code via SMS to their mobile phones.



Solution Summary

! > Important: The TeleSign Verify SMS plugin v1.3.0 is only compatible with RSA Adaptive Authentication On-Premise (AAOP) 7.1.0.3 and above. TeleSign's SMS plugin v1.0.2 supports AAOP 7.1.0.2. Earlier AAOP releases are no longer supported. Contact RSA Professional Services for details.

TeleSign Verify SMS is an Authentication Credentials Service Provider (ACSP) that can be integrated with online web applications to enable out-of-band authentication. In order to confirm a user's identity, a client application sends an initial request to the Verify service and includes the user's mobile phone number and language preference. In response, the Verify service generates a random code and delivers it to the user's mobile phone in an SMS message written in the user's preferred language. The client displays a login prompt that instructs the user to retrieve the code when it arrives and submit it for validation. When the user submits the code, the Verify service compares it to the original and returns the result to the client.

The TeleSign Verify SMS RSA Adaptive Authentication On-Premise (AAOP) Multi-Credential Framework (MCF) Plug-in enables an enterprise to integrate Verify SMS step-up authentication into its AAOP environment while minimizing development efforts. The plug-in extends generic AAOP SOAP API data structures to support TeleSign Verify's authentication and credential management use cases. This allows developers to focus on the new functionality without having to change existing AAOP Web Service workflows or make direct calls to TeleSign's Web Services API.

! > Important: Although the MCF relies on standard AAOP Web Service calls to establish bi-directional communication between a client and a plug-in, it requires custom data structures that encapsulate plug-in-specific request/response messages. In order to incorporate a plug-in into a customer's AAOP integration, a developer must modify the integration's existing code to use these structures as described in this guide.

After you have incorporated the plug-in into your AAOP environment, you will be able create AAOP policies to challenge users with the TeleSign Verify SMS authentication method.

The plug-in also allows you to maintain user profiles in the AAOP database, each containing a user's phone number and/or language preference. This is an optional feature. If you chose not to use it, you must retrieve the phone number and language preference from the customer's system and pass it to the plug-in when you challenge a user.

TeleSign Verify SMS RSA MCF Plug-in Overview	
Supports External User/Credential Provisioning	N/A
Contains a Synchronous Authentication Method	Yes
Contains an Asynchronous Authentication Method	No
Stores User Credentials Locally (required/optional/no)	Optional
Stores Custom User Data Locally (required/optional/no)	Optional
Caches Temporary Data Locally	Yes

Note: The TeleSign Verify SMS plugin v1.3 uses Transport Layer Security (TLS) 1.2 to communicate with the TeleSign server via TeleSign's REST API. Support for TLSv1.0 and SSLv3 has been deprecated.

Plug-In Configuration and Usage

The instructions in this document are divided into the following subsections:

- [TeleSign Verify SMS MCF Plug-In Configuration](#) contains instructions for deploying the TeleSign Verify SMS plug-in.
- [TeleSign Verify SMS MCF Plug-In SOAP Data Types](#) contains descriptions of the plug-in's SOAP data types and examples of how to use them to send and receive plug-in data through AAOP SOAP API. Use this section as a companion to the *RSA Adaptive Authentication (On-Premise) Workflows and Processes Guide* and the *AAOP Web Services API Reference Guide*.

Before You Begin

! > Important: The TeleSign Verify SMS plugin v1.3.0 is only compatible with RSA Adaptive Authentication On-Premise (AAOP) 7.1.0.3 and above. TeleSign's SMS plugin v1.0.2 supports AAOP 7.1.0.2. Earlier AAOP releases are no longer supported. Contact RSA Professional Services for details.

This guide is intended for developers who would like to integrate the TeleSign Verify SMS RSA AAOP Plug-in into an AAOP environment. To benefit from the guide's material, you should have working knowledge of XML schema, SOAP, AAOP workflows and the AAOP Web Services API. You should also have access to AAOP administrative and development documentation. Ensure that AAOP is running properly prior to configuring the integration.

The *tesign-rsa-aa-sms-acsp-impl-1.3.zip* integration kit contains the plug-in's JAR files, XML schemas and configuration files. Contact RSA Professional Services for a copy of the integration kit if you haven't already done so.

In order to use the plug-in, you must have a TeleSign customer account with access to the Verify SMS Web Services API. The plug-in needs a customer ID and an API key to authenticate to the TeleSign server. Contact RSA Professional Services for more information.

! > Important: You must obtain your TeleSign customer ID, TeleSign API key and a copy of the *tesign-rsa-aa-sms-acsp-impl-1.3.zip* integration kit from RSA Professional Services before proceeding.

Your AAOP application server must use Java 7 in order to run the plugin. If you are currently using an earlier version of Java, follow the appropriate instructions for your application server in the **Upgrade to Java 7** section of your *AAOP Installation and Upgrade Guide*.

Prerequisites for WebLogic Environments

If you are running AAOP on a WebLogic application server, you must set the following JVM argument in your WebLogic domain environment and restart your server:

```
-DUseSunHttpHandler=true
```

Failure to set the argument will prevent the plugin from communicating with the TeleSign server. See [Appendix B](#) for details.

! > Important: Failure to set the `-DUseSunHttpHandler=true` JVM argument in a WebLogic environment will prevent the plugin from communicating with the TeleSign server. See [Appendix B](#) for details.


TeleSign Verify SMS MCF Plug-In Configuration

Follow the instructions below to install and configure the plug-in. Once you have finished, you can begin to customize your AAOP integration to support TeleSign Verify SMS step-up authentication.

1. Extract the contents of the *telesign-rsa-aa-sms-acsp-impl-1.3.zip* integration kit in a temporary directory on the AAOP server. This will create the following directory tree:

TeleSign_SMS_ACSP (root directory)

- └─ **config/** – contains a custom initialization file for the plug-in:
 - └─ **c-config-acsp_telesignSms-1.3.0.xml** – contains plug-in initialization parameters
- └─ **lib/** – contains the plug-in's implementation JAR files:
 - └─ **telesignGen-rsa-aa-plugin-1.3.0.jar** – contains an abstract implementation of the MCF plug-in ACSP interface, and includes abstract subclasses of the ACSP SDK's request and response classes and custom enumerated types.
 - └─ **telesignSms-rsa-aa-plugin-1.3.0.jar** – contains the plug-in and includes concrete subclasses of the abstract classes above and an additional custom enumerated type.
- └─ **schema/** – contains XML schemas that define the plug-in's SOAP data structures/types:
 - └─ **telesign-gen-rsa-aa-plugin-1.3.0.xsd** – contains custom enumerated types and abstract data structures that extend the MCF plug-in base schema, *ACSP.xsd*.
 - └─ **telesign-sms-rsa-aa-plugin-1.3.0.xsd** – contains a custom enumerated type and the plug-in's concrete data structures, which extend the abstract data structures above.

 **Note:** The enumerated types and request/response classes in *telesignGen-rsa-aa-plugin-1.3.0.jar* and *telesignSms-rsa-aa-plugin-1.3.0.jar* are Java implementations of the enumerated types and data structures defined in the *telesign-gen-rsa-aa-plugin-1.3.0.xsd* and *telesign-sms-rsa-aa-plugin-1.3.0.xsd* schemas.

2. Open the *c-config-acsp.xml* file in the AAOP configuration directory¹, add the following element to the *GenericMetadataList* bean's metadata list and save the file. You'll find another copy of *c-config-acsp.xml* in the *AdaptiveAuthenticationAdmin* application's *WEB-INF\classes\configs* directory. Make the same changes to this file as well.

```
<ref bean="TELESIGN_SMS_METADATA_ENTRY"/>
```

```
<bean class="com.rsa.csd.mcf.acsp.generic.GenericMetadataList"
  id="genericMetadataList">
  <!-- add ACSP metadata entries here -->
  <property name="metadataList">
    <list>
      <ref bean="OTP_METADATA_ENTRY"/>
      <ref bean="TELESIGN_SMS_METADATA_ENTRY"/>
    </list>
  </property>
</bean>
```

¹ by default, this directory is */rsa/configs* on UNIX and *C:\rsa\configs* on Windows

3. Copy both *JAR* files to the *AdaptiveAuthentication* application's *WEB-INF/lib* directory and the *AdaptiveAuthenticationAdmin* application's *WEB-INF/lib* directory.
4. Copy the *c-config-acsp_tesignSms-1.3.0.xml* file from the *config* directory to the AAOP configuration directory and open it for editing.
5. Set the *ClassFreeBean*'s parameters as described in the following table. Use the [image](#) on the next page as a reference.

TeleSign Verify SMS MCF Plug-in Initialization Parameter		
Parameter (entry key)	Description	Default Value
<i>tesignCustomerId</i>	A unique string that identifies your TeleSign account. Contact RSA Professional Services to obtain your ID.	<i>tesign_customer_id</i> (This is a placeholder ²)
<i>tesignApiKey</i>	A Base64-encoded string used to authenticate TeleSign API requests made on behalf of your customer ID. Contact RSA Professional Services to obtain your API Key.	<i>tesign_api_key</i> (This is a placeholder)
<i>tesignRestApiUrl</i>	The base URL for the TeleSign Rest API. At the time of this writing, the URL is <i>https://rest.tesign.com</i> .	<i>https://rest.tesign.com</i>
<i>tesignApiVersion</i>	The version of the TeleSign Rest API that the plug-in uses. The current plug-in uses <i>v1</i> .	<i>v1</i>
<i>maxMessageLength</i>	The maximum length of the text message that TeleSign will deliver to the user's mobile phone.	<i>160</i> ³
<i>httpProxyFlag</i>	A Boolean value that indicates whether the plug-in will use a proxy server. If the value is <i>true</i> , the plug-in will use the remaining parameters below to connect to the given server.	<i>false</i>
<i>httpProxyIPAddress</i>	The proxy server's IP address. You must set this parameter to a properly-formatted IP address even if you set <i>httpProxyFlag</i> to <i>false</i> .	<i>http_proxy_ip_address</i> (This is a placeholder)
<i>httpProxyPort</i>	The proxy server's port number. You must set this parameter to any valid port number even if you set <i>httpProxyFlag</i> to <i>false</i> .	<i>http_proxy_port</i> (This is a placeholder)
<i>httpProxyAuthenticationFlag</i>	A Boolean value that indicates whether the plug-in must authenticate to the proxy server. If the value is <i>true</i> , the plug-in will use the parameters below to authenticate. You must set this parameter to <i>true</i> or <i>false</i> even if you set <i>httpProxyFlag</i> to <i>false</i> .	<i>false</i>
<i>httpProxyUsername</i>	The username the plug-in will use to authenticate to the proxy server. You must set this parameter to a string even if you set <i>httpProxyAuthenticationFlag</i> to <i>false</i> .	<i>proxy_user</i> (This is a placeholder)
<i>httpProxyPassword</i>	The password the plug-in will use to authenticate to the proxy server. You must set this parameter to a string even if you set <i>httpProxyAuthenticationFlag</i> to <i>false</i> .	<i>proxy_password</i> (This is a placeholder)

² i.e. There is no default value for the parameter. The value in red is a placeholder for an actual parameter value.

³ There may be additional constraints on the maximum length of your SMS message. Consult RSA Professional Services for more information.

Sample `c-config-acsp_tesignSms-1.3.0.xml` configuration:

```
<bean class="com.passmarksecurity.config.bean.ClassFreeBean"
      id="tesignSmsConfiguration">
  <property name="parameters">
    <map>
      <entry key="tesignCustomerId">
        <value>ABCDABCD-ABCD-ABCD-ABCD-ABCDABCDABCD</value>
      </entry>
      <entry key="tesignApiKey">
        <value>z/7FRN9h4UOs9yREP4O3ywA2/P2+EkyWQg== ...</value>
      </entry>
      <entry key="tesignRestApiUrl">
        <value>https://rest.tesign.com</value>
      </entry>
      <entry key="tesignApiVersion">
        <value>v1</value>
      </entry>
      <entry key="maxMessageLength">
        <value>160</value>
      </entry>
      <entry key="httpProxyFlag">
        <value>false</value>
      </entry>
      <entry key="httpProxyIPAddress">
        <value>127.0.0.1</value>
      </entry>
      <entry key="httpProxyPort">
        <value>80</value>
      </entry>
      <entry key="httpProxyAuthenticationFlag">
        <value>false</value>
      </entry>
      <entry key="httpProxyUsername">
        <value>testuser</value>
      </entry>
      <entry key="httpProxyPassword">
        <value>testpwd</value>
      </entry>
    </map>
  </property>
</bean>
```

!> Important: You must assign a value to every `ClassFreeBean` parameter, even if you set `httpProxyFlag` and/or `httpProxyAuthenticationFlag` to `false`. You must set `httpProxyIPAddress` to a properly-formatted IP address and `httpProxyPort` to a valid port number.

- Keep the *GenericAcspType* bean's default property values and set the *AcspMetaData* bean's *encrypted* property value to *true* to enable encryption or *false* to disable encryption.

```
<bean class="com.rsa.csd.mcf.acsp.AcspMetaData" id="TELESIGN_SMS_METADATA">
  <property name="acspType"><ref bean="TELESIGN_SMS_TYPE"/></property>
  <property name="acspStatusString"><value>ACTIVE</value></property>
  <property name="billFlag"><value>true</value></property>
  <property name="encrypted"><value>true</value></property>
</bean>
```

! > **Important:** Consult the *RSA Adaptive Authentication On-Premise Operations Guide* for instructions to encrypt user credentials.

- Keep the *GenericMetadataListEntry* bean's default property values and save the file.
- Place a copy of the *c-config-acsp_tesignSms-1.3.0.xml* file in the *AdaptiveAuthenticationAdmin* application's *WEB-INF\classes\configs* directory.
- Log in to the AAOP backoffice application, select the **Administration** tab and select **Authentication Methods** from the **Components** list on the left.
- Type *TeleSign2FASms* in the **Authentication Methods** list, click the **Save** button and click the **Publish** button on the horizontal menu near top of the screen.

The screenshot shows the 'Authentication Methods' configuration interface. Under the 'General' tab, the following settings are visible:

- Maximum User Failure Count:** 3
- Count abandoned challenges as failures:**
- Out-of-Band Token Length:** 6
- Use Health Checks:**
- Proxy Host:** (empty field)
- Proxy Port:** 8080
- Proxy User Name:** (empty field)
- Proxy Password:** (empty field)
- Authentication Methods:** A list containing 'TeleSign2FASms' and 'OTP'. 'TeleSign2FASms' is highlighted.

At the bottom of the form, there are 'Save' and 'Undo' buttons.

TeleSign Verify SMS MCF Plug-In SOAP Data Types

This section describes the plug-in's SOAP data structures as defined in the *telesign-gen-rsa-aa-plugin-1.3.0.xsd* and *telesign-sms-rsa-aa-plugin-1.3.0.xsd* XML schema files. A client uses these data structures to pass and receive plug-in data in standard AAOP SOAP API request and response messages.

ACSP Data Structures Overview


All MCF plug-in schemas include eight data structures that derive from generic structures defined in the MCF plug-in base schema, *ACSP.xsd* (listed below in request/response pairs). Four of the structures are used to perform synchronous authentication, two are used to manage user data, and the remaining two (in red below) are used to perform asynchronous authentication:

- *AcspAuthenticationRequest/AcspAuthenticationResponse*
- *AcspAuthStatusRequest/AcspAuthStatusResponse* (used for asynchronous authentication)
- *AcspChallengeRequest/AcspChallengeResponse*
- *AcspManagementRequest/AcspManagementResponse*

! > Important: The TeleSign Verify SMS Plug-in performs synchronous authentication. It doesn't utilize the asynchronous data structures.

TeleSign defined the Verify SMS Plug-in's data structures in two separate XML schemas: The *telesign-gen-rsa-aa-plugin-1.3.0.xsd* schema contains eight abstract structures that extend the MCF plug-in base schema structures listed above:

- *TelesignGenAcspAuthenticationRequest/TelesignGenAcspAuthenticationResponse*
- *TelesignGenAcspAuthStatusRequest/TelesignGenAcspAuthStatusResponse* (unused)
- *TelesignGenAcspChallengeRequest/TelesignGenAcspChallengeResponse*
- *TelesignGenAcspManagementRequest/TelesignGenAcspManagementResponse*

 **Note:** The *telesign-gen-rsa-aa-plugin-1.3.0.xsd* schema contains abstract data structure definitions that TeleSign reuses in its Verify Call MCF plug-in implementation. These structures extend the structures in the MCF plug-in base schema, *ACSP.xsd*.

The *telesign-sms-rsa-aa-plugin-1.3.0.xsd* XML schema contains the plug-in implementation's structures, which extend the abstract structures defined in *telesign-gen-rsa-aa-plugin-1.3.0.xsd*:

- *TelesignSmsAcspAuthenticationRequest/TelesignSmsAcspAuthenticationResponse*
- *TelesignSmsAcspAuthStatusRequest/TelesignSmsAcspAuthStatusResponse* (unused)
- *TelesignSmsAcspChallengeRequest/TelesignSmsAcspChallengeResponse*
- *TelesignSmsAcspManagementRequest/TelesignSmsAcspManagementResponse*

TeleSign Verify SMS Plug-in Enumerated Type Definitions

Some of the plug-in’s data structures use enumerated type variables to request a specific action, or to inform the client of the status or result of an operation. The following tables describe each of these enumerated types:

Type: <i>VerifyStateEnum</i>	
Used In: <i>TelesignSmsAcspAuthenticationResponse</i>	
Description: The <i>TelesignSmsAcspAuthenticationResponse</i> structure contains a <i>VerifyStateEnum</i> variable to indicate the result of an <i>authenticate</i> request. The <i>VerifyStateEnum</i> type is declared in the <i>telesign-gen-rsa-aa-plugin-1.3.0.xsd</i> schema.	
<i>VALID</i>	Indicates that the OTP contained in an <i>authenticate</i> request was valid.
<i>INVALID</i>	Indicates that the OTP contained in an <i>authenticate</i> request was invalid.
<i>UNKNOWN</i>	Indicates that the result of the <i>authenticate</i> request is unavailable.

Type: <i>ActionTypeEnum</i>	
Used In: <i>TelesignSmsAcspManagementRequest</i>	
Description: The <i>TelesignSmsAcspManagementRequest</i> contains an <i>ActionTypeEnum</i> variable to instruct the plug-in to perform a specific management function. The <i>ActionTypeEnum</i> type is declared in the <i>telesign-gen-rsa-aa-plugin-1.3.0.xsd</i> schema.	
<i>ADD_USER</i>	Instructs the plug-in to store a user’s mobile phone number and/or language preference in the AAOP database.
<i>DELETE_USER_DETAILS</i>	Instructs the plug-in to clear the user’s mobile phone number and/or language preference from the AAOP database.
<i>GET_USER_DETAILS</i>	Instructs the plug-in to retrieve and return the user’s mobile phone number and/or language preference from the AAOP database if one or both of them exist.
<i>UPDATE_PHONE_NUMBER</i>	Instructs the plug-in to replace the user’s mobile phone number in the AAOP database with a new phone number in the request.
<i>UPDATE_LANGUAGE</i>	Instructs the plug-in to replace the user’s language preference in the AAOP database with a new language preference in the request.
<i>UPDATE_PHONE_NUMBER_AND_LANGUAGE</i>	Instructs the plug-in to replace the user’s mobile phone number and language preference in the AAOP database with a new mobile phone number and language preference in the request.

Type: <i>StatusCodeEnum</i>	
Used In: <i>TelesignSmsAcspChallengeResponse</i>	
<p>Description: The <i>TelesignSmsAcspChallengeResponse</i> structure contains a <i>StatusCodeEnum</i> variable to inform the client whether the TeleSign Verify SMS service delivered, is in the process of delivering or failed to deliver the OTP to the user in an SMS message.</p> <p>The <i>StatusCodeEnum</i> type is declared in the <i>telesign-sms-rsa-aa-plugin-1.3.0.xsd</i> schema.</p>	
<i>DELIVERED_TO_HANDSET</i>	The service delivered the OTP to the user's mobile phone.
<i>DELIVERED_TO_GATEWAY</i>	The service delivered the OTP to the carrier's gateway.
<i>ERROR_DELIVERING_SMS_TO_HANDSET</i>	The service couldn't deliver the OTP due to an error.
<i>TEMPORARY_PHONE_ERROR</i>	The service couldn't deliver the OTP because the user's mobile phone is temporarily unavailable.
<i>PERMANENT_PHONE_ERROR</i>	The service couldn't deliver the OTP because the user's mobile phone is permanently unavailable.
<i>GATEWAY_OR_NETWORK_CANNOT_ROUTE_MESSAGE</i>	The carrier couldn't route the OTP message to the device.
<i>MESSAGE_EXPIRED_BEFORE_DELIVERY</i>	The service timed out before it could deliver the OTP.
<i>SMS_NOT_SUPPORTED</i>	The user's mobile phone doesn't support SMS.
<i>MESSAGE_BLOCKED_BY_TELESIGN</i>	TeleSign blocked the message from being delivered.
<i>INVALID_OR_UNSUPPORTED_MESSAGE_CONTENT</i>	The service couldn't deliver the OTP to the user because the SMS message wasn't formatted properly.
<i>FINAL_STATUS_UNKNOWN</i>	The delivery status is unknown.
<i>MESSAGE_IN_PROGRESS</i>	The service is in the process of sending the OTP message to the SMS gateway.
<i>QUEUED_BY_TELESIGN</i>	TeleSign has placed the OTP message in a queue.
<i>QUEUED_AT_GATEWAY</i>	The SMS gateway placed the OTP message in a queue
<i>STATUS_DELAYED</i>	The status of the transaction is temporarily unavailable.
<i>TRANSACTION_NOT_ATTEMPTED</i>	The plug-in didn't attempt to contact the TeleSign service because the request didn't contain the required data.
<i>NOT_AUTHORIZED</i>	The customer isn't authorized to use the TeleSign service or the plug-in didn't submit the proper credentials.
<i>STATUS_NOT_AVAILABLE</i>	The status of the transaction isn't available.

TeleSign Verify SMS Plug-in ACSP Data Structure Definitions

This section describes the plug-in's SOAP request and response data structures. Your client will use these structures to send and receive plug-in data within standard SOAP API messages.

TelesignSmsAcspManagementRequest

You must use the *TelesignSmsAcspManagementRequest* data structure to activate the plug-in for a user. You may also use it to create and manage persistent user profiles for the plug-in in the AAOP database.

Schema Definition

Type: <i>TelesignSmsAcspManagementRequest</i>	
ACSP Base Type: <i>AcspManagementRequest</i> Super Type: <i>TelesignGenAcspAuthenticationResponse</i>	
Description: The <i>TelesignSmsAcspManagementRequest</i> data structure consists of two string variables that hold a user's profile data and an ActionTypeEnum variable that holds a profile management instruction for the plug-in. Use the structure to pass data and instructions to the plug-in during <i>createUser</i> , <i>updateUser</i> and <i>query</i> requests	
AAOP API Operations: <i>createUser</i> , <i>updateUser</i> and <i>query</i>	
Elements:	
name: <i>actionType</i> type: ActionTypeEnum	An instruction that indicates whether the plug-in should create, update, return or delete a user's profile data. This value is mandatory.
name: <i>phoneNo</i> type: <i>string</i>	The user's mobile phone number including the country code. This value must not contain any whitespaces or punctuation (parentheses, dashes, etc.) You must set this value if you set <i>actionType</i> to <i>UPDATE_PHONE_NUMBER</i> or <i>UPDATE_PHONE_NUMBER_AND_LANGUAGE</i> . You may omit this value if you set <i>actionType</i> to <i>DELETE_USER_DETAILS</i> , <i>GET_USER_DETAILS</i> , <i>UPDATE_LANGUAGE</i> or <i>ADD_USER</i> .
name: <i>language</i> type: <i>string</i>	A language code that represents the user's preferred language. Consult with RSA Professional Services for a list of language codes that TeleSign currently supports. You must set this value if you set <i>actionType</i> to <i>UPDATE_LANGUAGE</i> or <i>UPDATE_PHONE_NUMBER_AND_LANGUAGE</i> . You may omit this value if you set <i>actionType</i> to <i>DELETE_USER_DETAILS</i> , <i>GET_USER_DETAILS</i> , <i>UPDATE_PHONE_NUMBER</i> or <i>ADD_USER</i> .

Usage

Include a *TelesignSmsAcspManagementRequest* payload in *createUser*, *updateUser* and *query* request messages to enable the plug-in for a given user or to manage user plug-in profiles in the AAOP database.

- [Activate the Plug-in for a User](#)
- [Create/Update/Delete a User Profile](#)
- [Request a User's Profile Data](#)

Activate the Plug-in for a User: The AAOP API's *AcspManagementRequestData* structure contains an *AcspManagementRequest* payload structure and a *credentialProvisioningStatus* structure. Use them together as follows to activate the plug-in for a given user.

! > Important: If you want to make the Verify SMS step-up authentication method available to a given user, you must activate the plug-in for that use as described below.

Set the *AcspManagementRequestData* element's *payload type* attribute to *TelesignSmsAcspManagementRequest* and its *credentialProvisioningStatus* value to *ACTIVE*.

The management request example below enables the plug-in for a user and stores the user's profile data in the AAOP database. Note that the *payload actionType* value is *ADD_USER*, which instructs the plugin to create a user profile containing the given mobile phone number and language preference. However, you can choose a different [actionType](#) value based on your requirements. For example, if you want to activate a plugin for a user without creating a profile, set the *actionType* to *GET_USER_DETAILS*, which instructs the plugin to return the user's profile data if any exists in the database. If the user doesn't have a profile, the plugin will return an empty response.

Sample addUser SOAP request snippet that enables TeleSign Verify step-up authentication for a user:

```
<ws:credentialManagementRequestList>
  <ws:acspManagementRequestData>
    <ws:credentialProvisioningStatus>ACTIVE</ws:credentialProvisioningStatus>
    <ws:payload xsi:type="ns503:TelesignSmsAcspManagementRequest"
      xmlns:ns503="http://ws.sms.rsaaa.plugin.telesign.com"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <ws1:actionType xmlns:ws1="http://ws.gen.rsaaa.plugin.telesign.com">ADD_USER</ws1:actionType>
      <ws1:phoneNo xmlns:ws1="http://ws.gen.rsaaa.plugin.telesign.com">12155555556</ws1:phoneNo>
      <ws1:language xmlns:ws1="http://ws.gen.rsaaa.plugin.telesign.com">en-US</ws1:language>
    </ws:payload>
  </ws:acspManagementRequestData>
</ws:credentialManagementRequestList>
```

Note: To disable the plug-in for a user, set the *credentialProvisioningStatus* value to *DISABLED*.

Create/Update/Delete a User Profile: The plug-in needs a user's mobile phone number and language preference in order to issue a challenge. You can pass this information from the customer's system when you make a challenge request, or instruct the plug-in to maintain the information in the AAOP database. You may also let the customer's system manage one item and let the plug-in manage the other. For example, you might have the customer's system handle mobile phone numbers and the plug-in handle language preferences. In this case, you would only need to pass the user's mobile number to the plug-in when issuing a challenge.

! > Important: If you manage phone numbers with the plug-in, you are responsible for keeping them in sync with the customer's database. For example, if the customer's web application allows users to change their contact information (email addresses, phone numbers, addresses, etc.), you must modify the application to call the plug-in whenever a user changes his/her phone number. RSA recommends storing the user's number in the customer's data store if possible. This would require you to include the number with each challenge request, but it eliminates the need to replicate data and the potential to introduce profile data inconsistencies.

To create, update⁴ or delete a user's profile, set the *AcspManagementRequestData* payload type to *TelesignSmsAcspManagementRequest*, set the *actionType* to *ADD_USER*, *UPDATE_** or *DELETE_USER_DETAILS*, and set the [profile data that the actionType requires](#). The example below creates a user profile with a mobile phone number and language preference.

Snippet of an addUser SOAP request to create a user profile in the AAOP database:

```
<ws:credentialManagementRequestList>
  <ws:acspManagementRequestData>
    <ws:credentialProvisioningStatus>ACTIVE</ws:credentialProvisioningStatus>
    <ws:payload xsi:type="ns503:TelesignSmsAcspManagementRequest"
      xmlns:ns503="http://ws.sms.rsaaa.plugin.telesign.com"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <ws1:actionType xmlns:ws1="http://ws.gen.rsaaa.plugin.telesign.com">ADD_USER</ws1:actionType>
      <ws1:phoneNo xmlns:ws1="http://ws.gen.rsaaa.plugin.telesign.com">12155555556</ws1:phoneNo >
      <ws1:language xmlns:ws1="http://ws.gen.rsaaa.plugin.telesign.com">en-us</ws1:language>
    </ws:payload>
  </ws:acspManagementRequestData>
</ws:credentialManagementRequestList>
```

Snippet of an updateUser SOAP request to change a user's profile in the AAOP database:

```
<ws:payload xsi:type="ns503:TelesignSmsAcspManagementRequest"
  xmlns:ns503="http://ws.sms.rsaaa.plugin.telesign.com"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ws1:actionType xmlns:ws1="http://ws.gen.rsaaa.plugin.telesign.com">
    UPDATE_PHONE_NUMBER</ws1:actionType>
  <ws1:phoneNo xmlns:ws1="http://ws.gen.rsaaa.plugin.telesign.com">
    12155555775</ws1:phoneNo>
</ws:payload>
```

⁴ UPDATE_* refers to UPDATE_PHONE_NUMBER, UPDATE_LANGUAGE and UPDATE_PHONE_NUMBER_AND_LANGUAGE collectively.

Snippet of an `updateUser` SOAP request to delete a user's profile from the AAOP database:

```
<ws:payload xsi:type="ns503:TelesignSmsAcspManagementRequest"
  xmlns:ns503="http://ws.sms.rsaaa.plugin.telesign.com"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <wsl:actionType xmlns:wsl="http://ws.gen.rsaaa.plugin.telesign.com">
    DELETE_USER_DETAILS</wsl:actionType>
</ws:payload>
```

Request a User's Profile Data: You can retrieve a user's plug-in profile data by making a management request. Set the `AcspManagementRequestData` payload type attribute to `TelesignSmsAcspManagementRequest` and set the `actionType` value to `GET_USER_DETAILS`. You can also use `GET_USER_DETAILS` if you want to activate the plugin without creating a profile.

Snippet of a query SOAP request to return a user's profile data:

```
<ws:credentialManagementRequestList>
  <ws:acspManagementRequestData>
    <ws:credentialProvisioningStatus>ACTIVE</ws:credentialProvisioningStatus>
    <ws:payload xsi:type="ns633:TelesignSmsAcspManagementRequest"
      xmlns:ns633="http://ws.sms.rsaaa.plugin.telesign.com"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <wsl:actionType xmlns:wsl="http://ws.gen.rsaaa.plugin.telesign.com">
        GET_USER_DETAILS</wsl:actionType>
    </ws:payload>
  </ws:acspManagementRequestData>
</ws:credentialManagementRequestList>
```

Note: The MCF routes all `createUser`, `updateUser` and `query` SOAP requests to the same plug-in method (named `manage`), which is responsible for managing user profiles. The method determines which operation to perform based on the `TelesignSmsAcspManagementRequest` `actionType`, regardless of the SOAP API request type. This means, for example, that you can query a user's profile in an updateUser request (by setting `actionType` to `GET_USER_DETAILS`). The response would include the user's profile data along with the results of the update to the user's AAOP account.

Although this is legal, TeleSign and RSA recommend that you match the `actionType` to the request type for code readability and debugging purposes. If you choose to do this, use `createUser` requests to send an `ADD_USER` `actionType`, use `updateUser` requests to send `UPDATE_*` and `DELETE_USER_DETAILS` `actionTypes`, and use `query` requests to send a `GET_USER_DETAILS` `actionType`.

TelesignSmsAcspManagementResponse

The *TelesignSmsAcspManagementResponse* data structure is used to return a user's plug-in profile data to the client. It will only contain data in response to *GET_USER_DETAILS* request.

Schema Definition

Type: <i>TelesignSmsAcspManagementResponse</i>	
ACSP Base Type: <i>AcspManagementResponse</i> Super Type: <i>TelesignGenAcspManagementResponse</i>	
Description: <i>TelesignSmsAcspManagementResponse</i> contains two string fields to hold a user's plug-in profile data (mobile phone number and language preference). The plug-in only populates these fields in response to <i>GET_USER_DETAILS</i> request.	
AAOP API Operations: <i>createUserResponse</i> , <i>updateUserResponse</i> and <i>queryResponse</i>	
Elements:	
name: <i>phoneNo</i> type: <i>string</i>	The user's mobile phone number including the country code. When you request a user's plug-in profile , the plug-in will retrieve it from the AAOP database and return the number in this variable (if it exists).
name: <i>language</i> type: <i>string</i>	A language code that represents the user's language preference. When you request a user's plug-in profile, the plug-in will retrieve it from the AAOP database and return the user's language preference in this variable (if it exists).

Usage

Determine a Management Request Status: The plug-in doesn't return the status of a management request in the *TelesignSmsAcspManagementResponse* payload. Instead, it returns this information in the *acspManagementResponseData* *callStatus* structure. Read this structure's *statusCode* to determine the overall status of the transaction. The *statusCode* value will be *CallStatusCode.ERROR* if the plug-in encountered a system error, and *CallStatusCode.FAIL* if the request *actionType* was *UPDATE_** and the required profile data was missing.

Sample updateUserResponse snippet for a failed attempt to update the user's plug-in profile:

```

<ns1:acspManagementResponseData>
  <ns1:acspAccountId>jsammon</ns1:acspAccountId>
  <ns1:callStatus>
    <ns1:statusCode>FAIL</ns1:statusCode>
    <ns1:statusDescription>Phone number is missing in the request</ns1:statusDescription>
  </ns1:callStatus>
  <ns1:payload xsi:type="ns3:TelesignSmsAcspManagementResponse"
    xmlns:ns3="http://ws.sms.rsaaa.plugin.telesign.com"/>
</ns1:acspManagementResponseData>
    
```

Other than in the cases listed above, a response's *statusCode* will be *CallStatusCode.SUCCESS*.

Sample *createUserResponse* snippet indicating that the plug-in added a profile for a new AAOP user:

```
<nsl:credentialManagementResponseList xsi:type="ns1:CredentialManagementResponseList">
  <nsl:acspManagementResponseData>
    <nsl:acspAccountId>jsammon</nsl:acspAccountId>
    <nsl:callStatus>
      <nsl:statusCode>SUCCESS</nsl:statusCode>
      <nsl:statusDescription>User added successfully</nsl:statusDescription>
    </nsl:callStatus>
    <nsl:payload xsi:type="ns3:TelesignSmsAcspManagementResponse"
      xmlns:ns3="http://ws.sms.rsaaa.plugin.telesign.com" />
  </nsl:acspManagementResponseData>
</nsl:credentialManagementResponseList>
```

Sample *updateUserResponse* snippet indicating that the plug-in updated a user's profile:

```
<nsl:credentialManagementResponseList xsi:type="ns1:CredentialManagementResponseList">
  <nsl:acspManagementResponseData>
    <nsl:acspAccountId>jsammon</nsl:acspAccountId>
    <nsl:callStatus>
      <nsl:statusCode>SUCCESS</nsl:statusCode>
      <nsl:statusDescription>Phone number updated successfully</nsl:statusDescription>
    </nsl:callStatus>
    <nsl:payload xsi:type="ns3:TelesignSmsAcspManagementResponse"
      xmlns:ns3="http://ws.sms.rsaaa.plugin.telesign.com" />
  </nsl:acspManagementResponseData>
</nsl:credentialManagementResponseList>
```

Sample *updateUserResponse* snippet indicating that the plug-in deleted a user's profile:

```
<nsl:credentialManagementResponseList xsi:type="ns1:CredentialManagementResponseList">
  <nsl:acspManagementResponseData>
    <nsl:acspAccountId>json</nsl:acspAccountId>
    <nsl:callStatus>
      <nsl:statusCode>SUCCESS</nsl:statusCode>
      <nsl:statusDescription>
        User's details removed successfully from AA database</nsl:statusDescription>
    </nsl:callStatus>
    <nsl:payload xsi:type="ns3:TelesignSmsAcspManagementResponse"
      xmlns:ns3="http://ws.sms.rsaaa.plugin.telesign.com" />
  </nsl:acspManagementResponseData>
</nsl:credentialManagementResponseList>
```


Read Profile Data: If you requested a user's profile data in your *TelesignSmsAcspManagementRequest* payload, the plug-in will return the user's phone number (if found) and language preference (if found) in the *TelesignSmsAcspManagementResponse* payload. If the payload doesn't contain a phone number or a language preference, it means that the user doesn't have a database profile.

Sample queryResponse snippet containing a user's plug-in profile data:

```
<ns1:acspManagementResponseData>
  <ns1:acspAccountId>jsammon</ns1:acspAccountId>
  <ns1:callStatus>
    <ns1:statusCode>SUCCESS</ns1:statusCode>
    <ns1:statusDescription>
      User details fetched successfully from AA database</ns1:statusDescription>
    </ns1:callStatus>
    <ns1:payload xsi:type="ns3:TelesignSmsAcspManagementResponse"
      xmlns:ns3="http://ws.sms.rsaaa.plugin.telesign.com">
      <s3:phoneNo xmlns:s3="http://ws.gen.rsaaa.plugin.telesign.com">
        12155555775</s3:phoneNo>
      <s4:language xmlns:s4="http://ws.gen.rsaaa.plugin.telesign.com">
        en-us</s4:language>
      </ns1:payload>
    </ns1:acspManagementResponseData>
```

TelesignSmsAcspChallengeRequest

Include a *TelesignSmsAcspChallengeRequest* payload in an AAOP *challenge* SOAP request to initiate the Verify SMS authentication process for a given user. You must include the user's mobile phone number and/or language preference in the payload if you don't maintain the data in a plug-in profile.

If the SOAP request is successful, the TeleSign Verify service will generate an OTP and attempt to deliver it in a text message to the user's phone.

Schema Definition

Type: <i>TelesignSmsAcspChallengeRequest</i>	
ACSP Base Type: <i>AcspChallengeRequest</i>	Super Type: <i>TelesignGenAcspChallengeRequest</i>
<p>Description: The <i>TelesignSmsAcspChallengeRequest</i> structure contains a field to hold a given user's mobile phone number, a field to hold the user's language preference and a field to hold a custom SMS message template. Use the structure to pass data to the plug-in during a <i>challenge</i> request.</p>	
AAOP API Operations: <i>challenge</i>	
Elements:	
name: <i>phoneNo</i> type: <i>string</i>	<p>The user's mobile phone number including the country code. This value must not contain any whitespaces or punctuation (parentheses, dashes, etc.).</p> <p>The value is mandatory unless you maintain the user's mobile phone number in a plug-in profile.</p>
name: <i>language</i> type: <i>string</i>	<p>A language code that represents the user's language preference.</p> <p>This value is mandatory unless you maintain the user's language preference in a plug-in profile. Contact RSA Professional Services for details.</p>
name: <i>template</i> type: <i>string</i>	<p>An optional string parameter containing a custom SMS message template for TeleSign to use to deliver OTPs to mobile devices.</p> <p>If this value is not NULL, it must contain a properly-formatted message template that doesn't exceed the value of the maxMessageLength initialization parameter. The template must also include the string <code>\$\$CODE\$\$</code> as a placeholder for the OTP. See the Usage section for more details.</p>

Usage

Include a *TelesignSmsAcspChallengeRequest* payload in your *challenge* request to initiate the TeleSign Verify SMS authentication workflow.

- [Determine Whether to Initiate a TeleSign Verify SMS Authentication Workflow](#)
- [Initiate a TeleSign Verify SMS Authentication Workflow](#)

Determine Whether to Initiate an Authentication Workflow: When you receive an AAOP *AnalyzeResponse* that triggers a rule to challenge a given user, iterate through the *requiredCredentialsList* sequence. If you find a *requiredCredential* element with a *genericCredentialType* value of *TeleSign2FASms*, you may initiate the TeleSign Verify SMS authentication workflow.

Sample *AnalyzeResponse* snippet to challenge a user with the TeleSign Verify service:

```
<ns1:requiredCredential>
  <ns1:credentialType>USER_DEFINED</ns1:credentialType>
  <ns1:genericCredentialType>TeleSign2FASms</ns1:genericCredentialType>
  <ns1:groupName>DEFAULT</ns1:groupName>
  <ns1:preference>0</ns1:preference>
  <ns1:required>true</ns1:required>
</ns1:requiredCredential>
```

Initiate the TeleSign Verify SMS Authentication Workflow: Set the *AcspChallengeRequestData* element's *payload type* attribute to *TelesignSmsAcspChallengeRequest*. You must include the user's mobile phone number unless you manage it in a plug-in profile. You must include the language preference, again unless you manage it with the plug-in. You may include a template with a customized message for TeleSign to use when it delivers the OTP.

When the plug-in receives the request, it will instruct the Verify SMS service to generate an OTP and deliver it to the user's mobile phone in an SMS message written in the user's preferred language.

Sample *TelesignSmsAcspChallengeRequest* for a user who has a plug-in profile in the AAOP database:

```
<ws:credentialChallengeRequestList>
  <ws:acspChallengeRequestData>
    <ws:payload xmlns:ns503="http://ws.sms.rsaaa.plugin.telesign.com"
      xsi:type="ns503:TelesignSmsAcspChallengeRequest"/>
  </ws:acspChallengeRequestData>
</ws:credentialChallengeRequestList>
```

 **Note:** If you include a [message template](#) in the request, it will override the user's language preference.

Sample TelesignSmsAcspChallengeRequest payload with mobile number and language preference:

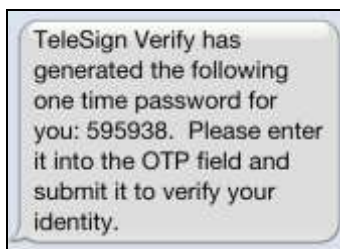
```
<ws:acspChallengeRequestData>
  <ws:payload xmlns:ns503="http://ws.sms.rsaaa.plugin.telesign.com"
    xsi:type="ns503:TelesignSmsAcspChallengeRequest">
    <ws1:phoneNo xmlns:ws1="http://ws.gen.rsaaa.plugin.telesign.com">
      12155555775</ws1:phoneNo>
    <ws1:language xmlns:ws1="http://ws.gen.rsaaa.plugin.telesign.com">
      en-us</ws1:language>
    </ws:payload>
  </ws:acspChallengeRequestData>
```

The payload in the snippet below contains the user’s mobile phone number and a template message. This example is only valid if the plug-in is maintaining the user’s language preference in an AAOP database profile.

Sample TelesignSmsAcspChallengeRequest payload with mobile number and message template:

```
<ws:payload xsi:type="ns503:TelesignSmsAcspChallengeRequest"
  xmlns:ns503="http://ws.sms.rsaaa.plugin.telesign.com" >
  <ws1:phoneNo xmlns:ws1="http://ws.gen.rsaaa.plugin.telesign.com">
    12155555775</ws1:phoneNo>
  <ns503:template>
    TeleSign Verify has generated the following one time password
    for you: $$CODE$$ Please enter it into the OTP field and
    submit it to verify your identity.</ns503:template>
  </ws:payload>
```

The template above is properly formatted because it contains the \$\$CODE\$\$ token. TeleSign will replace the token with the value of the OTP before it delivers the SMS message:



!> Important: If you include a template in a challenge request, it will override the user’s language preference. However, you still must either send the preference or store it a profile.

TelesignSmsAcspChallengeResponse

The *TelesignSmsAcspChallengeResponse* data structure is used in a *challengeResponse* to return the delivery status of a Verify SMS OTP message.

Schema Definition

Type: <i>TelesignSmsAcspChallengeResponse</i>	
ACSP Base Type: <i>AcspChallengeResponse</i>	Super Type: <i>TelesignGenAcspChallengeResponse</i>
Description: The <i>TelesignSmsAcspChallengeResponse</i> structure contains a StatusCodeEnum field, which the plug-in uses to notify the client of the OTP delivery status.	
AAOP API Operations: <i>challengeResponse</i>	
Elements:	
name: <i>telesign_status_code</i> type: StatusCodeEnum	The OTP delivery status. The <i>telesign_status_code</i> value notifies the client whether service delivered, is in the process of delivering, or failed to deliver the OTP message to the user's mobile phone.

Usage

When you receive an AAOP *ChallengeResponse*, determine the status of the Verify SMS service's attempt to deliver an OTP to a user and either prompt the user to submit an OTP or display an error message based on the status.

- [Determine the Status of the OTP Delivery](#)
- [Prompt a User to Submit an OTP](#)
- [Display an Error Message](#)


Determine OTP Delivery Status: Read the *acspChallengeResponseData* element's *statusCode* to determine the overall status of the phone SMS, and read the *TelesignSmsAcspChallengeResponse* payload's *telesign_status_code* for lower level details. If the *statusCode* is *CallStatusCode.SUCCESS*, prompt the user to submit the TeleSign Verify OTP when it arrives. Otherwise, display an error message.

The plug-in will set *statusCode* to *CallStatusCode.SUCCESS* if *telesign_status_code* is one of the following values:

- *DELIVERED_TO_HANDSET*
- *MESSAGE_IN_PROGRESS*
- *DELIVERED_TO_GATEWAY*
- *QUEUED_BY_TELESIGN*
- *QUEUED_AT_GATEWAY*
- *STATUS_DELAYED*

The plug-in will set the *statusCode* to *CallStatusCode.ERROR* if it encountered a system error and to *CallStatusCode.FAIL* if *telesign_status_code* is any of the remaining [StatusCodeEnum](#) values.

Prompt a User to Submit an OTP: If the *statusCode* value is *CallStatusCode.SUCCESS*, display a login form and instruct the user to retrieve the OTP from the SMS message and submit it for verification.

 **Note:** You may want to tailor the instructions based on the value of the *telesign_status_code*. For example “Your one-time-password has been delivered to your mobile device ...” vs. “TeleSign is attempting to deliver a one-time-password to your mobile device ...”.

Sample *TelesignSmsAcspChallengeResponse* payload indicating that the service delivered an OTP:

```
<ns1:payload xsi:type="ns3:TelesignSmsAcspChallengeResponse"
  xmlns:ns3="http://ws.sms.rsaaa.plugin.telesign.com"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ns3:telesign_status_code>DELIVERED_TO_HANDSET</ns3:telesign_status_code>
</ns1:payload>
```


Set a time-limit for displaying the prompt. If it expires, instruct the user to try logging in again later. Keep in mind that a *challengeResponse* message's *sessionId* and *transactionID* need to be passed to a subsequent *authenticate* request before they expire. Ensure your time-limit doesn't exceed the lifetime of these values to avoid a SOAP fault. See the *AAOP Web Services API Reference Guide* for information about setting authentication attempt timeout limits.

!> Important: If the time-limit expires, the OTP will no longer be valid. Instruct the user to delete the OTP (if it arrives) and to use the new one that will be delivered when he/she attempts to log in again.

Sample *challengeResponse* SOAP message indicating that the service is attempting to deliver an OTP:

```
<ns1:credentialChallengeList xsi:type="ns1:CredentialChallengeList">
  <ns1:acspChallengeResponseData>
    <ns1:acspAccountId>jsammon</ns1:acspAccountId>
    <ns1:callStatus>
      <ns1:statusCode>SUCCESS</ns1:statusCode>
      <ns1:statusDescription>Message in progress</ns1:statusDescription>
    </ns1:callStatus>
    <ns1:payload xsi:type="ns3:TelesignSmsAcspChallengeResponse"
      xmlns:ns3="http://ws.sms.rsaaa.plugin.telesign.com">
      <ns3:telesign_status_code>MESSAGE_IN_PROGRESS</ns3:telesign_status_code>
    </ns1:payload>
  </ns1:acspChallengeResponseData>
</ns1:credentialChallengeList>
```

Display an Error Message: If the *statusCode* value is *CallStatusCode.FAIL* or *CallStatusCode.ERROR*, display an error message that includes the appropriate instructions for the user.

 **Note:** If the *statusCode* value is *CallStatusCode.ERROR*, the *statusDescription* will contain a description of the error. When you receive a *CallStatusCode.ERROR*, you may want to read the *statusDescription* value and include it in your error message.

Sample challengeResponse SOAP message indicating that the plug-in didn't attempt to deliver an OTP:

```
<nsl:credentialChallengeList xsi:type="ns1:CredentialChallengeList">
  <nsl:acspChallengeResponseData>
    <nsl:acspAccountId>jsammon</nsl:acspAccountId>
    <nsl:callStatus>
      <nsl:statusCode>FAIL</nsl:statusCode>
      <nsl:statusDescription>
        Template format is incorrect, it doesn't contain $$CODE$$ in it</nsl:statusDescription>
      </nsl:callStatus>
      <nsl:payload xsi:type="ns3:TelesignSmsAcspChallengeResponse"
        xmlns:ns3="http://ws.sms.rsaaa.plugin.telesign.com">
        <ns3:telesign_status_code>TRANSACTION_NOT_ATTEMPTED</ns3:telesign_status_code>
      </nsl:payload>
    </nsl:acspChallengeResponseData>
  </nsl:credentialChallengeList>
```

TelesignSmsAcspAuthenticationRequest

Include a *TelesignSmsAcspAuthenticationRequest* payload in an *authenticate* SOAP request message to submit a user's OTP to the TeleSign Verify service for verification.

Schema Definition

Type: <i>TelesignSmsAcspAuthenticationRequest</i>	
ACSP Base Type: <i>AcspAuthenticationRequest</i> Super Type: <i>TelesignGenAcspAuthenticationRequest</i>	
Description: The <i>TelesignSmsAcspAuthenticationRequest</i> data structure contains a string field to hold an OTP. Use it in an <i>authenticate</i> request to pass a user's OTP to the plug-in for verification.	
AAOP API Operations: <i>authenticateRequest</i>	
Elements:	
name: <i>verify_code</i> type: <i>string</i>	An OTP submitted by a user. This value is mandatory

Usage

In order to verify a user's identity, the user must submit an OTP to the TeleSign Verify service.

Submit an OTP: Set the *AcspAuthenticateRequestData* element's *payload type* attribute to *TelesignSmsAcspAuthenticationRequest* and set the payload's *verify_code* value to the OTP the user submitted from the login form. The client shouldn't allow users to submit an empty string.

! > Important: The plugin will return an error if it receives a null or empty OTP. Your client's login form should require users to enter a non-empty OTP before submitting it to the plugin.

Sample TelesignSmsAcspAuthenticationRequest payload:

```

<ws:credentialDataList>
  <ws:acspAuthenticationRequestData>
    <ws:payload xsi:type="ns283:TelesignSmsAcspAuthenticationRequest"
      xmlns:ns3="http://ws.sms.rsaaa.plugin.telesign.com">
      <ws:verify_code xmlns:ws="http://ws.gen.rsaaa.plugin.telesign.com">
        123456</ws:verify_code>
      </ws:payload>
    </ws:acspAuthenticationRequestData>
  </ws:credentialDataList>
  
```


TelesignSmsAcspAuthenticationResponse

An *authenticateResponse* message's *TelesignSmsAcspChallengeResponse* payload will contain a TeleSign Verify service OTP authentication result.

Schema Definition

Type: <i>TelesignSmsAcspAuthenticationResponse</i>	
ACSP Base Type: <i>AcspAuthenticationRequest</i> Super Type: <i>TelesignGenAcspAuthenticationResponse</i>	
Description: The <i>TelesignSmsAcspAuthenticationResponse</i> data structure contains a <i>VerifyStateEnum</i> field to hold the result of a TeleSign Verify SMS authentication. It also contains a <i>StatusCodeEnum</i> field, but you should ignore its value.	
AAOP API Operations: <i>authenticateResponse</i>	
Elements:	
name: <i>telesign_verify_state</i> type: VerifyStateEnum	The TeleSign Verify OTP authentication result.
name: <i>telesign_status_code</i> type: <i>StatusCodeEnum</i>	Ignore this value. Use <i>telesign_verify_state</i> to determine the authentication result

 **Note:** When you process a *TelesignCallAcspAuthenticationResponse* payload, you should ignore the *telesign_status_code* variable's value.

Usage

Read the *TelesignSmsAcspAuthenticationResponse* payload *telesign_verify_state* variable to determine if the user passed or failed authentication. Allow or deny a user access to a requested resource based on the value of this variable. Ignore the payload's *telesign_status_code*.

- [Allow the User Access to the Requested Resource](#)
- [Deny the User Access to the Requested Resource](#)

Allow Access: If the *telesign_verify_state* variable's value is *VALID*, allow the user access to the requested resource.

Sample *TelesignSmsAcspAuthenticationResponse* indicating that the user passed authentication:

```
<ns1:credentialAuthResultList xsi:type="ns1:CredentialAuthResultList">
  <ns1:acspAuthenticationResponseData>
    <ns1:acspAccountId>jsammon</ns1:acspAccountId>
    <ns1:callStatus>
      <ns1:statusCode>SUCCESS</ns1:statusCode>
      <ns1:statusDescription>Delivered to gateway</ns1:statusDescription>
    </ns1:callStatus>
    <ns1:payload xsi:type="ns3:TelesignSmsAcspAuthenticationResponse"
      xmlns:ns3="http://ws.sms.rsaaa.plugin.telesign.com">
      <ns2:telesign_verify_state xmlns:ns2="http://ws.gen.rsaaa.plugin.telesign.com">
        VALID</ns2:telesign_verify_state>
      <ns3:telesign_status_code>DELIVERED_TO_GATEWAY</ns3:telesign_status_code>
    </ns1:payload>
  </ns1:acspAuthenticationResponseData>
</ns1:credentialAuthResultList>
```

Deny Access: If the *telesign_verify_state* variable's value is *INVALID* or *UNKNOWN*, deny the user access and display the appropriate instructions to the user.

If the value is *INVALID*, and the user hasn't exceeded the failed authentication attempts limit, set a timer to delay the subsequent process. When the timer expires, send another *challenge SOAP* request to begin the authentication workflow again. The time interval of the delay should be configurable if possible.

!> Important: When a user fails authentication, the client should wait a brief amount of time before issuing another challenge request. The duration of the delay should be configurable if possible.

If the *telesign_verify_state* variable's value is *UNKNOWN*, instruct the user to contact the system administrator/customer service.

Sample TelesignSmsAcspAuthenticationResponse indicating that the user failed authentication:

```
<nsl:credentialAuthResultList xsi:type="ns1:CredentialAuthResultList">
  <nsl:acspAuthenticationResponseData>
    <nsl:acspAccountId>jsammon</nsl:acspAccountId>
    <nsl:callStatus>
      <nsl:statusCode>SUCCESS</nsl:statusCode>
      <nsl:statusDescription>Delivered to handset</nsl:statusDescription>
    </nsl:callStatus>
    <nsl:payload xsi:type="ns3:TelesignSmsAcspAuthenticationResponse"
      xmlns:ns3="http://ws.sms.rsaaa.plugin.telesign.com">
      <ns2:telesign_verify_state xmlns:ns2="http://ws.gen.rsaaa.plugin.telesign.com">
        INVALID</ns2:telesign_verify_state>
      <ns3:telesign_status_code>DELIVERED_TO_HANDSET</ns3:telesign_status_code>
    </nsl:payload>
  </nsl:acspAuthenticationResponseData>
</nsl:credentialManagementResponseList>
```

Certification Checklist for RSA AAOP MCF Plug-Ins

Dates Tested: January 7th, 2016 - February 25th, 2016

Certification Environment		
Product Name	Version	Platform
RSA AAOP	7.1.0.5 & 7.2.0	<ul style="list-style-type: none"> CentOS 6/Tomcat/Oracle RHEL/WebLogic/Oracle Windows/Tomcat/SQL Server
TeleSign Verify SMS RSA AAOP MCF Plug-in	1.3.0	<ul style="list-style-type: none"> CentOS 6/Tomcat/Oracle RHEL/WebLogic/Oracle Windows/Tomcat/SQL Server
TeleSign Verify	N/A	N/A

Mandatory Functionality	
Manage	
Provision external users/credentials	N/A
Create Plug-in Data	✓
Modify Plug-in Data	✓
Query Plug-in Data	✓
Delete Plug-in Data	✓
Challenge	
Initialize Challenge	✓
Return Challenge Status	✓
Deliver Authentication Credentials	✓
Authenticate (Synchronous)	
Submit Credentials for Validation	✓
Validate Credentials and Return Results	✓
Authenticate (Asynchronous)	
Query Authentication Status	N/A

JGS

✓ = Pass ✗ = Fail N/A = Not Applicable to Integration

Appendix A – Logging Configuration

To enable logging for the plug-in:

1. Navigate to the `/AdaptiveAuthentication/WEB-INF/classes/` directory on your web application server, open the `log4j.properties` file for editing and add the following lines to the end of the file:

```
# TeleSign Verify SMS Plug-in Logging Properties  
log4j.logger.com.telesign = DEBUG,TELESIGN_APPENDER  
log4j.appender.TELESIGN_APPENDER = org.apache.log4j.DailyRollingFileAppender  
log4j.appender.TELESIGN_APPENDER.layout = org.apache.log4j.PatternLayout  
log4j.appender.TELESIGN_APPENDER.layout.ConversionPattern= %d{dd MMM yyyy  
HH:mm:ss,SSS} %-5p [%t] [%X{sesstag}] [%X{txidtag}] %c - %m%n  
log4j.appender.TELESIGN_APPENDER.File = VrsaVlogsVtelesign.log  
log4j.appender.LOGFILE.DatePattern = '.yyyy-MM-dd
```

2. Save the file and restart your web application server.

Appendix B – WebLogic Domain Configuration

The TeleSign Verify SMS plugin v1.3 uses Transport Layer Security (TLS) 1.2 to communicate with the TeleSign server via TeleSign's REST API. By default, each time the plugin attempts to create an HTTPS connection over TLS during an AAOP challenge request, WebLogic will attempt to cast the connection object `t` to a proprietary, deprecated class (`weblogic.net.http.SOAPHttpsURLConnection`), which will result in the following class cast exception:

```
java.lang.ClassCastException: weblogic.net.http.SOAPHttpsURLConnection
cannot be cast to javax.net.ssl.HttpsURLConnection>com.rsa.csd.mcf.McfException:
Failure in activating ACSP challenge functionality
```

To avoid this issue, you must add the following JVM argument to your WebLogic domain environment and restart your server. The argument will force WebLogic to use Sun's HTTP handlers:

```
-DUseSunHttpHandler=true
```

The method you use to add the argument will depend on various factors such as your WebLogic server version, the method you use to start and stop your server, etc. Consult your WebLogic administrator for details.

The script below can be used as a reference for setting the argument in a WebLogic 12.1.2 environment. The script must be named `setUserOverrides.sh` and placed in the `DOMAIN_HOME/bin` directory. For example, `/opt/wls12120/user_projects/domains/mydomain/bin/setUserOverrides.sh`

! > Important: The following example is only included as a reference. Consult your WebLogic documentation and your WebLogic administrator before making any changes to your environment.

```
#!/bin/sh
#
# setUserOverrides.sh

# Cause Weblogic to use Sun's HTTP handlers.
USE_SUN_HTTP_HANDLER="-DUseSunHttpHandler=true"
export USE_SUN_HTTP_HANDLER

JAVA_OPTIONS="${JAVA_OPTIONS} ${USE_SUN_HTTP_HANDLER}"
export JAVA_OPTIONS
```