RSA Adaptive Authentication On-Premise®
Implementation Guide

TeleSign Verify Call MCF Plugin 1.5

John Sammon, RSA Partner Engineering
Last Modified: 05/1/2018

RSA READY

Solution Summary

Important: The TeleSign Verify Call plugin v1.5.0 is only compatible with RSA Adaptive Authentication On-Premise (AAOP) 7.1.0.3 and above. TeleSign's Verify Call plugin v1.0.2 supports AAOP 7.1.0.2. Earlier AAOP releases are no longer supported. Consult RSA Professional Services for more information.

TeleSign Verify Call 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 phone number and language preference. In response, the Verify service generates a random code and delivers it to the user's phone as a voice message spoken in the user's preferred language. The client displays a login prompt that instructs the user to submit the code for validation after it arrives. When the user submits the code, the plug-in sends it to the Verify service, which compares it to the original and returns the result to the client.

The TeleSign Verify Call RSA Adaptive Authentication On-Premise Multi-Credential Framework (MCF) plug-in enables an enterprise to integrate Verify Call 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 Call 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 Call 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)	No
Stores Custom User Data Locally (required/optional/no)	Optional
Caches Temporary Data Locally	Yes

Note: The TeleSign Verify Call plugin v1.5.0 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 Call RSA AAOP MCF Plug-In Configuration** contains instructions for deploying the TeleSign Verify Call plug-in.
- **TeleSign Verify Call 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 Call plugin v1.5.0 is only compatible with RSA Adaptive Authentication On-Premise (AAOP) 7.1.0.3 and above. TeleSign's Verify Call plugin v1.0.2 supports AAOP 7.1.0.2. Earlier AAOP releases are no longer supported. Consult RSA Professional Services for more information.

This guide is intended for developers who would like to integrate the TeleSign Verify Call RSA AAOP MCF 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 *telesign-rsa-aa-call-acsp-impl-1.5.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 Call 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 *telesign-rsa-aa-call-acsp-impl-1.5.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 Call RSA AAOP 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 Call step-up authentication.

1. Extract the contents of the *telesign-rsa-aa-call-acsp-impl-1.5.zip* integration kit in a temporary directory on the AAOP server. This will create the following directory tree:

   TeleSign_Call_ACSP *(root directory)*

   - ***config/ –*** contains a custom initialization file for the plug-in:
     - ***c-config-acsp_telesignCall-1.5.0.xml*** – contains plug-in initialization parameters
   - ***lib/ –*** contains the plug-in's implementation JAR files:
     - ***telesignGen-rsa-aa-plugin-1.5.0.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.
     - ***telesignCall-rsa-aa-plugin-1.5.0.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-1.5.0.xsd*** – contains custom enumerated types and abstract data structures that extend the MCF plug-in base schema, *ACSP.xsd.*
     - ***telesign-call-rsa-aa-plugin-1.5.0.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-1.5.0.jar* and *telesignCall-rsa-aa-plugin-1.5.0.jar* are Java implementations of the enumerated types and data structures defined in the *telesign-gen-rsa-aa-plugin-1.5.0.xsd* and *telesign-call-rsa-aa-plugin-1.5.0.xsd* schemas.

2. Open the *c-config-acsp.xml* file in the AAOP configuration directory*, 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_CALL_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_CALL_METADATA_ENTRY"/>
        </list>
    </property>
</bean>
```

* 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_telesignCall-1.5.0.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 Call MCF Plug-in Initialization Parameter		
Parameter (entry key)	**Description**	**Default Value**
telesignCustomerId	A unique string that identifies your TeleSign account. Contact RSA Professional Services to obtain your ID.	*telesign_customer_id* (This is a placeholder†)
telesignApiKey	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.	*telesign_api_key* (This is a placeholder)
telesignRestApiUrl	The base URL for the TeleSign Rest API. At the time of this writing, the URL is *https://rest.telesign.com*.	*https://rest.telesign.com*
telesignApiVersion	The version of the TeleSign Rest API that the plug-in uses. The current plug-in uses *v1*.	*v1*
httpProxyFlag	A Boolean value that indicates whether the plug-in will use a proxy server. If the value is *true*, the plug-in will use the remaining parameters below to connect to the given server.	*false*
httpProxyIPAddress	The proxy server's IP address. You must set this parameter to a properly-formatted IP address even if you set *httpProxyFlag* to *false*.	*http_proxy_ip_address* (This is a placeholder)
httpProxyPort	The proxy server's port number. You must set this parameter to any valid port number even if you set *httpProxyFlag* to *false*.	*http_proxy_port* (This is a placeholder)
httpProxyAuthenticationFlag	A Boolean value that indicates whether the plug-in must authenticate to the proxy server. If the value is *true*, the plug-in will use the parameters below to authenticate. You must set this parameter to *true* or *false* even if you set *httpProxyFlag* to *false*.	*false*
httpProxyUsername	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 *httpProxyAuthenticationFlag* to *false*.	*proxy_user* (This is a placeholder)
httpProxyPassword	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 *httpProxyAuthenticationFlag* to *false*.	*proxy_password* (This is a placeholder)

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

Sample c-config-acsp_telesignCall-1.5.0.xml configuration:

```
<bean class="com.passmarksecurity.config.bean.ClassFreeBean"
        id="telesignCallConfiguration">
    <property name="parameters">
        <map>
            <entry key="telesignCustomerId">
                <value>ABCDABCD-ABCD-ABCD-ABCD-ABCDABCDABCD</value>
            </entry>
            <entry key="telesignApiKey">
                <value>z/7FRN9h4UOs9yREP4O3ywA2/P2+EkyWQg== ...</value>
            </entry>
            <entry key="telesignRestApiUrl">
                <value>https://rest.telesign.com</value>
            </entry>
            <entry key="telesignApiVersion">
                <value>v1</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>rsa</value>
            </entry>
            <entry key="httpProxyPassword">
                <value>rsa</value>
            </entry>
        </map>
    </property>
</bean>
```

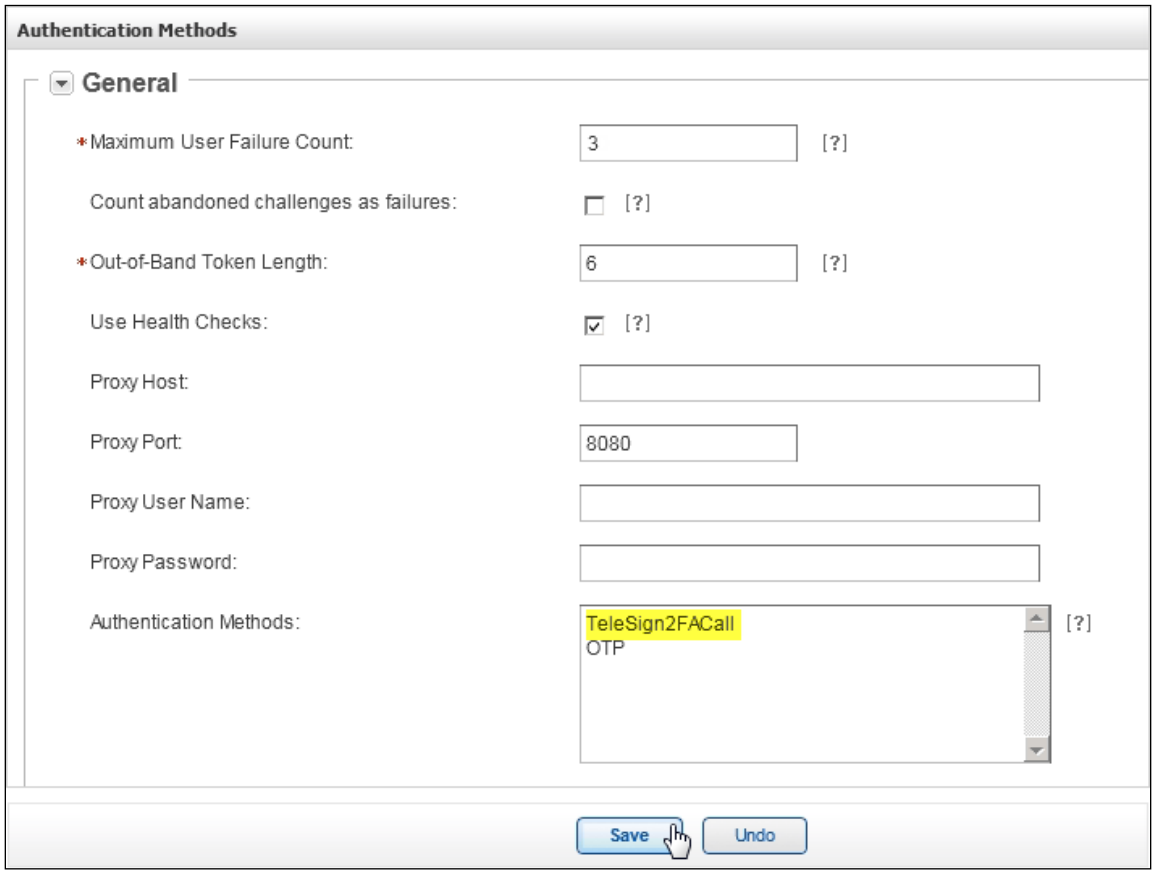
Important: You must assign a value to every *ClassFreeBean* parameter, even if you set *httpProxyFlag* and/or *httpProxyAuthenticationFlag* to *false*. You must set *httpProxyPort* to a valid port number and *httpProxyIPAddress* to a properly-formatted IP address.

6. 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_CALL_METADATA">
    <property name="acspType"><ref bean="TELESIGN_CALL_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.

7. Keep the *GenericMetadataListEntry* bean's default property values and save the file.
8. Place a copy of the *c-config-acsp_telesignCall-1.5.0.xml* file in the AdaptiveAuthenticationAdmin application's WEB-INF\classes\configs directory.
9. Log in to the AAOP backoffice application, select the **Administration** tab and select **Authentication Methods** from the **Components** list on the left.
10. Type *TeleSign2FACall* in the **Authentication Methods** list, click the **Save** button and click the **Publish** button on the horizontal menu near top of the screen.



TeleSign Verify Call 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-1.5.0.xsd* and *telesign-call-rsa-aa-plugin-1.5.0.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 Call Plug-in performs synchronous authentication, and doesn't utilize the asynchronous data structures.

TeleSign defined the Verify Call Plug-in's data structures in two separate XML schemas:

The *telesign-gen-rsa-aa-plugin-1.5.0.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-1.5.0.xsd* schema contains abstract data structure definitions that TeleSign reuses in its Verify SMS MCF plug-in implementation. These structures extend the structures in the MCF plug-in base schema, *ACSP.xsd*.

The *telesign-call-rsa-aa-plugin-1.5.0.xsd* XML schema contains the plug-in implementation's structures, which extend the abstract structures defined in *telesign-gen-rsa-aa-plugin-1.5.0.xsd*:

- *TelesignCallAcspAuthenticationRequest| TelesignCallAcspAuthenticationResponse*
- *TelesignCallAcspAuthStatusRequest/TelesignCallAcspAuthStatusResponse (unused)*
- *TelesignCallAcspChallengeRequest/TelesignCallAcspChallengeResponse*
- *TelesignCallAcspManagementRequest| TelesignCallAcspManagementResponse*

TeleSign Verify Call 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: *VerifyStateEnum*	
Used In: *TelesignCallAcspAuthenticationResponse*	
Description: The *TelesignCallAcspAuthenticationResponse* structure contains a *VerifyStateEnum* variable to indicate the result of an *authenticate* request**.** The *VerifyStateEnum* type is declared in the *telesign-gen-rsa-aa-plugin-1.5.0.xsd* schema.	
VALID	Indicates that the OTP contained in an *authenticate* request was valid.
INVALID	Indicates that the OTP contained in an *authenticate* request was invalid.
UNKNOWN	Indicates that the result of the *authenticate* request is unavailable.

Type: *ActionTypeEnum*	
Used In: *TelesignCallAcspManagementRequest*	
Description: The *TelesignCallAcspManagementRequest* contains an *ActionTypeEnum* variable to instruct the plug-in to perform a specific management function. The *ActionTypeEnum* type is declared in the *telesign-gen-rsa-aa-plugin-1.5.0.xsd* schema.	
ADD_USER	Instructs the plug-in to store a user's phone number and/or language preference in the AAOP database.
DELETE_USER_DETAILS	Instructs the plug-in to clear the user's phone number and/or language preference from the AAOP database.
GET_USER_DETAILS	Instructs the plug-in to retrieve and return the user's phone number and/or language preference from the AAOP database if one or both of them exist.
UPDATE_PHONE_NUMBER	Instructs the plug-in to replace the user's phone number in the AAOP database with a new phone number in the request.
UPDATE_LANGUAGE	Instructs the plug-in to replace the user's language preference in the AAOP database with a new language preference in the request.
UPDATE_PHONE_NUMBER_AND_LANGUAGE	Instructs the plug-in to replace the user's phone number and language preference in the AAOP database with a new phone number and language preference in the request.

Used In: *TelesignCallAcspChallengeResponse*	
Description: The *TelesignCallAcspChallengeResponse* structure contains a *StatusCodeEnum* variable to inform the client whether the TeleSign Verify Call service delivered, is in the process of delivering or failed to deliver the OTP voice message to the user. The *StatusCodeEnum* type is declared in the *telesign-call-rsa-aa-plugin-1.5.0.xsd* schema.	
CALL_ANSWERED	The TeleSign service delivered the OTP to the user's phone.
NOT_ANSWERED	The TeleSign service attempted to deliver the OTP to the user's phone, but the user didn't answer the call.
DISCONNECT_OCCURRED_BEFORE_MESSAGE_COMPLETED	The TeleSign service began delivering the OTP, but the call disconnected prematurely.
CALL_IN_PROGRESS	The TeleSign service is in the process of delivering the OTP to the user's phone.
WRONG_OR_INVALID_PHONE_NUMBER	The plug-in provided an invalid phone number to the TeleSign service.
CALL_NOT_HANDLED_YET	The TeleSign service hasn't attempted to call the user's phone yet. Delivery is pending.
CALL_FAILED	The TeleSign service attempted to deliver the OTP to the user's phone, but the phone call failed.
LINE_BUSY	The TeleSign service attempted to deliver the OTP to the user's phone, but the line was busy.
TRANSACTION_NOT_ATTEMPTED	The plug-in didn't attempt to contact the TeleSign service because the challenge request didn't contain the required data.
NOT_AUTHORIZED	The customer isn't authorized to use the TeleSign service or the plug-in didn't submit the proper credentials.
STATUS_NOT_AVAILABLE	The status of the transaction isn't available.

Important: The *TelesignCallAcspAuthenticationResponse* structure also contains a *StatusCodeEnum* variable, but the variable doesn't contain any significant information.

TeleSign Verify Call 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.

TelesignCallAcspManagementRequest

You must use the *TelesignCallAcspManagementRequest* 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: *TelesignCallAcspManagementRequest*	
ACSP Base Type: *AcspManagementRequest* **Super Type***: TelesignGenAcspAuthenticationResponse*	
Description: The *TelesignCallAcspManagementRequest* 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 *createUser*, *updateUser* and *query* requests.	
AAOP API Operations: *createUser, updateUser and query*	
Elements:	
name: *actionType* **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: *phoneNo* **type:** *string*	The user's 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 *actionType* to *UPDATE_PHONE_NUMBER* or *UPDATE_PHONE_NUMBER_AND_LANGUAGE*. You may omit this value if you set *actionType* to *DELETE_USER_DETAILS*, *GET_USER_DETAILS*, *UPDATE_LANGUAGE* or *ADD_USER*.
name: *language* **type:** *string*	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 *actionType* to *UPDATE_LANGUAGE* or *UPDATE_PHONE_NUMBER_AND_LANGUAGE*. You may omit this value if you set *actionType* to *DELETE_USER_DETAILS*, *GET_USER_DETAILS*, *UPDATE_PHONE_NUMBER* or *ADD_USER*.

Usage

Include a *TelesignCallAcspManagementRequest* 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 to activate the plug-in for a given user.

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

Set the *AcspManagementRequestData* element's *payload type* attribute to *TelesignCallAcspManagementRequest* and its *credentialProvisioningStatus* value to *ACTIVE*.
The management request example below enables the plug-in for a user without creating a profile. Note that the *payload* ***actionType*** value is *GET_USER_DETAILS*, which instructs the plugin to return the user's profile data if any exists. If you want to create a profile when you activate the plugin for a user, choose the appropriate action type. For example, if you want to store the user's phone number language preference, use the ***ADD_USER*** *actionType*.

Sample 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:TelesignCallAcspManagementRequest"
                xmlns:ns503="http://ws.call.rsaaa.plugin.telesign.com"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <ws1:actionType xmlns:ws1="http://ws.gen.rsaaa.plugin.telesign.com">GET_USER_DETAILS</ws1:actionType>
    </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 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 you can instruct the plug-in to maintain the information in the AAOP database. You may also let the customer's system manage one item and the plug-in manage the other. For example, you might have the customer's system handle 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 and language preference in the customer's data store if possible. This would require you to include both of them with each challenge request, but it eliminates the need to replicate data and the potential to introduce profile data inconsistencies.

To create, update[‡] or delete a user's profile, set the *AcspManagementRequestData payload type* to *TelesignCallAcspManagementRequest*, set the *actionType* to *ADD_USER*, *UPDATE_** or *DELETE_USER_DETAILS*, and set the **profile data that the *actionType* requires**.

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

```
<ws:acspManagementRequestData>
   <ws:credentialProvisioningStatus>ACTIVE</ws:credentialProvisioningStatus>
   <ws:payload xsi:type="ns503:TelesignCallAcspManagementRequest"
               xmlns:ns503="http://ws.call.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">12155555555</ws1:phoneNo>
      <ws1:language xmlns:ws1="http://ws.gen.rsaaa.plugin.telesign.com">en-us</ws1:language>
   </ws:payload>
</ws:acspManagementRequestData>
```

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

```
<ws:payload xsi:type="ns598:TelesignCallAcspManagementRequest"
            xmlns:ns598="http://ws.call.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>
```

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

```
<ws:payload xsi:type="ns598:TelesignCallAcspManagementRequest"
            xmlns:ns598="http://ws.call.rsaaa.plugin.telesign.com"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <ws1:actionType xmlns:ws1="http://ws.gen.rsaaa.plugin.telesign.com">
      DELETE_USER_DETAILS</ws1:actionType>
</ws:payload>
```

[‡] UPDATE_* refers to UPDATE_PHONE_NUMBER, UPDATE_LANGUAGE and UPDATE_PHONE_NUMBER_AND_LANGUAGE collectively.

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 *TelesignCallAcspManagementRequest* and set the *actionType* value to *GET_USER_DETAILS*.

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

```
<ws:credentialManagementRequestList>
   <ws:acspManagementRequestData>
      <ws:payload xsi:type="ns149:TelesignCallAcspManagementRequest"
                  xmlns:ns149="http://ws.call.rsaaa.plugin.telesign.com"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
         <ws1:actionType xmlns:ws1="http://ws.gen.rsaaa.plugin.telesign.com">
            GET_USER_DETAILS</ws1: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 *TelesignCallAcspManagementRequest 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*.

TelesignCallAcspManagementResponse

The *TelesignCallAcspManagementResponse* 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: *TelesignCallAcspManagementResponse*	
ACSP Base Type: *AcspManagementResponse* **Super Type:** *TelesignGenAcspManagementResponse*	
Description: *TelesignCallAcspManagementResponse* contains two string fields to hold a user's plug-in profile data (phone number and language preference). The plug-in only populates these fields in response to *GET_USER_DETAILS* request.	
AAOP API Operations: *createUserResponse, updateUserResponse and queryResponse*	
Elements:	
name: *phoneNo* **type:** *string*	The user's 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: *language* **type**: *string*	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 *TelesignCallAcspManagementResponse* 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.

1. *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:TelesignCallAcspManagementResponse"
                xmlns:ns3="http://ws.call.rsaaa.plugin.telesign.com"/>
</ns1:acspManagementResponseData>
```

Other than 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:

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

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

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

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

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

Read Profile Data: If you requested a user's profile data in your *TelesignCallAcspManagementRequest* payload, the plug-in will return the user's phone number (if found) and language preference (if found) in the *TelesignCallAcspManagementResponse* payload.

2. *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:TelesignCallAcspManagementResponse"
                xmlns:ns3="http://ws.call.rsaaa.plugin.telesign.com">
      <s3:phoneNo  xmlns:s3="http://ws.gen.rsaaa.plugin.telesign.com">12155555555</s3:phoneNo>
      <s4:language xmlns:s4="http://ws.gen.rsaaa.plugin.telesign.com">en-us</s4:language>
   </ns1:payload>
</ns1:acspManagementResponseData>
```

TelesignCallAcspChallengeRequest

Include a *TelesignCallAcspChallengeRequest* payload in an AAOP *challenge* SOAP request to initiate the Verify Call authentication process for a given user. You must include the user's 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 voice message to the user's phone.

Schema Definition

Type: *TelesignCallAcspChallengeRequest*	
ACSP Base Type: *AcspChallengeRequest*	**Super Type:** *TelesignGenAcspChallengeRequest*
Description: The *TelesignCallAcspChallengeRequest* structure contains a field to hold a given user's phone number and another to hold the user's language preference. Use the structure to pass data to the plug-in during a *challenge* request.	
AAOP API Operations: *challenge*	
Elements:	
name: *phoneNo* **type:** *string*	The user's phone number including the country code. This value must not contain any whitespaces or punctuation (parentheses, dashes, etc.). The value is mandatory unless you maintain the user's phone number in a plug-in profile.
name: *language* **type:** *string*	A language code that represents the user's language preference. This value is mandatory unless you maintain the user's language preference in a plug-in profile. Contact RSA Professional Services for details.

Usage

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

- **Determine Whether to Initiate an Authentication Workflow**
- **Initiate an 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 *TeleSign2FACall*, you may initiate the TeleSign Verify Call authentication workflow.

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

```
<ns1:requiredCredential>
    <ns1:credentialType>USER_DEFINED</ns1:credentialType>
    <ns1:genericCredentialType>TeleSign2FACall</ns1:genericCredentialType>
    <ns1:groupName>DEFAULT</ns1:groupName>
    <ns1:preference>0</ns1:preference>
    <ns1:required>true</ns1:required>
</ns1:requiredCredential>
```

Initiate an Authentication Workflow: Set the *AcspChallengeRequestData* element's *payload type* attribute to *TelesignCallAcspChallengeRequest*. You must include the user's phone number and language preference unless you manage them with the plug-in. When the plug-in receives the request, it will instruct the Verify Call service to generate an OTP and deliver it to the user's phone.

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

```
<ws:credentialChallengeRequestList>
    <ws:acspChallengeRequestData>
        <ws:payload xsi:type="ns422:TelesignCallAcspChallengeRequest"
                    xmlns:ns422="http://ws.call.rsaaa.plugin.telesign.com"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    </ws:acspChallengeRequestData>
</ws:credentialChallengeRequestList>
```

Sample TelesignCallAcspChallengeRequest payload containing phone number and language preference:

```
<ws:payload xsi:type="ns422:TelesignCallAcspChallengeRequest"
            xmlns:ns422="http://ws.call.rsaaa.plugin.telesign.com"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <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>
```

TelesignCallAcspChallengeResponse

The *TelesignCallAcspChallengeResponse* data structure is used in a *challengeResponse* to return the delivery status of a Verify Call OTP voice message.

Schema Definition

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

Usage

When you receive an AAOP *ChallengeResponse*, determine the status of the Verify Call 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 call, and read the *TelesignCallAcspChallengeResponse* 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:

- *CALL_ANSWERED*
- *CALL_IN_PROGRESS*
- *CALL_NOT_HANDLED_YET*
- *STATUS_NOT_AVAILABLE*

The plug-in will set *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 submit the OTP for verification when it arrives.

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>
      < s1:callStatus>
         <ns1:statusCode>SUCCESS</ns1:statusCode>
         <ns1:statusDescription>Call in progress</ns1:statusDescription>
      < ns1:callStatus>
      < s1:payload xsi:type="ns3:TelesignCallAcspChallengeResponse"
            xmlns:ns3="http://ws.call.rsaaa.plugin.telesign.com">
         <ns3:telesign_status_code>CALL_IN_PROGRESS</ns3:telesign_status_code>
      < ns1:payload>
   </ns1 acspChallengeResponseData>
</ns1:credentialChallengeList>
```

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 7.3 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. Whenever the time-limit expires, instruct the user to ignore the OTP (if it arrives) and to use the new one that will be delivered when he/she attempts to log in again.

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:

```
<ns1:acspChallengeResponseData>
    <ns1:acspAccountId>jsammon</ns1:acspAccountId>
    <ns1:callStatus>
        <ns1:statusCode>ERROR</ns1:statusCode>
        <ns1:statusDescription>
          Invalid Request: CountryCode Parameter(-10001)
          Invalid Request: PhoneNumber Parameter(-10001)
        </ns1:statusDescription>
    </ns1:callStatus>
    <ns1:payload xmlns:ns3="http://ws.call.rsaaa.plugin.telesign.com"
                 xsi:type="ns3:TelesignCallAcspChallengeResponse">
        <ns3:telesign_status_code>TRANSACTION_NOT_ATTEMPTED</ns3:telesign_status_code>
    </ns1:payload>
</ns1 acspChallengeResponseData>
```

TelesignCallAcspAuthenticationRequest

Include a *TelesignCallAcspAuthenticationRequest* payload in an *authenticate* SOAP request message to submit a user's OTP to theTeleSign Verify Call service for verification.

Schema Definition

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

```
<ws:credentialDataList>
   <ws:acspAuthenticationRequestData>
      <ws:payload xsi:type="ns503:TelesignCallAcspAuthenticationRequest"
         xmlns:ns503="http://ws.call.rsaaa.plugin.telesign.com"
         mlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <ws:verify_code xmlns:ws="http://ws.gen.rsaaa.plugin.telesign.com">
            60277</ws:verify_code>
      </ws:payload>
   </ws:acspAuthenticationRequestData>
</ws:credentialDataList>
```

TelesignCallAcspAuthenticationResponse

An *authenticateResponse* message's *TelesignCallAcspChallengeResponse* payload will contain the result of the authentication.

Schema Definition

Type: *TelesignCallAcspAuthenticationResponse*	
ACSP Base Type: *AcspAuthenticationResponse* **Super Type:** *TelesignGenAcspAuthenticationResponse*	
Description: The *TelesignCallAcspAuthenticationResponse* data structure contains a *VerifyStateEnum* field to hold the result of a TeleSign Verify Call authentication. It also contains a *StatusCodeEnum* field, but you should ignore its value.	
AAOP API Operations: *authenticateResponse*	
Elements:	
name: *telesign_verify_state* **type:** ***VerifyStateEnum***	The TeleSign Verify OTP authentication result.
name: *telesign_status_code* **type:** *StatusCodeEnum*	Ignore this value. Use telesign_*verify_state* 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 *TelesignCallAcspAuthenticationResponse payload telesign_verify_state* variable to determine if the user passed or failed authentication. Allow or deny user access to the requested resource and notify AAOP of the authentication result. 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 TelesignCallAcspAuthenticationResponse indicating that the user passed authentication:

```
<ns1:acspAuthenticationResponseData>
    <ns1:acspAccountId>jsammon</ns1:acspAccountId>
    <ns1:callStatus>
        <ns1:statusCode>SUCCESS</ns1:statusCode>
        <ns1:statusDescription>Call answered</ns1:statusDescription>
    </ns1:callStatus>
    <ns1:payload xsi:type="ns3:TelesignCallAcspAuthenticationResponse"
                 xmlns:ns3="http://ws.call.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>CALL_ANSWERED</ns3:telesign_status_code>
   </ns1:payload>
</ns1:acspAuthenticationResponseData>
```

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 TelesignCallAcspAuthenticationResponse indicating that the user failed authentication:

```
<ns1:payload xsi:type="ns3:TelesignCallAcspAuthenticationResponse"
             xmlns:ns3="http://ws.call.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>CALL_ANSWERED</ns3:telesign_status_code>
</ns1:payload>
```

Certification Checklist for RSA AAOP MCF Plug-Ins

Dates Tested: April 12th, 2018

Certification Environment	
Product	**Version**
RSA Adaptive Authentication (On-Premise)	7.3
TeleSign Verify Call RSA AAOP MCF Plug-in	1.5.0

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 – 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 Call 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 = \/rsa\/logs\/telesign.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 Call plugin v1.5 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
```