mChoice AVENTURA - REST Developer Guide

Of HTTP REST

Product Version 1.0

hSenid Mobile
Enabling the Mobile World

Sayuru Sewana Building

Nawam Mawatha

Colombo-2

Sri Lanka

Tel: +94-11-2446623/2446624 Fax: +94-11-2307579

Document Code HMS-HTTPREST-DGD-v1.0 Last edited: 20 September 2010

Table of contents

Change Control

Version	Date	Description	Author
1.0	20/09/10	Final Documentation	hSenid Mobile Technical Writing Team

About this document

The purpose of this hSenid document is to provide information on how Service Providers can use HTTP REST to connect to **mChoice AVENTURA.**

The intended audience for this document is Application developers.

The document is divided into the following chapters:

Chapter	Description
Overview	This chapter gives a brief description and main functionalities of **mChoice AVENTURA REST**.
SMS REST API	This chapter gives a brief description of **SMS REST API.**

Conventions used in this document

We use the following text formatting conventions:

italic

Used for document titles, emphasis, and for email addresses, web URLs and file and directory names.

Bold

Used for emphasis and for the command options you select.

`Letter gothic`

Used for literal code, such as configuration files, Java class names, method names and API calls.

`Letter gothic italic`

Used for arguments and parameters that will be replaced with an actual value.

Chapter 1

1 Overview

This document describes how the Service Providers can use the **HTTP REST** to send and receive content from mChoice AVENTURA.

This documentation focuses on Application Developers who will not be working on the APIs provided by mChoice and will be developing an application from the scratch. The guidance provided will help you to start developing your application in any language you want since the documentation provides basic steps.

In order to launch your application you will need a unique application ID and a password. This will be requested for access authentication when you want to send messages using Avantura system. Further more you are given a unique keyword so that subscribers can send messages to your application.

When your application is ready to launch, contact us to get your keyword and application id. You can select your own password and encode it using MD5 encryption. The encrypted password should be provided to us in order to launch your application.

Chapter 2

2 SMS REST API

The SMS REST Service provides operations for sending SMS to mChoice AVENTURA and to receive the SMS.

E.g., A user sending a text message to a mobile phone from an Application and receiving SMS from a mobile phone to an application.

NOTE: The examples in the following sections will describe request, response formats. All examples of responses use XML as datatype.

2.1 Send SMS

Send SMS supports only POST HTTP requests. This is used when sending SMS to a mobile phone from an application.

2.1.1 Basic Access Authentication

In an HTTP transaction, Basic Access Authentication method is used to allow the Application to provide user authentication details when making a request. HTTP basic authentication requires application ID and password before a client can access Aventura system. When the application ID and password are entered it will carry the credentials in base 64 format in the Authorization header of the HTTP request. You can see the Authorization header in the example request given below which contains the base 64 format of credentials.

While sending the request application id and the password have to be provided and it must be encoded in Basic HTTP Authentication format.

Before the password is set to Basic HTTP Authentication it has to be encoded using MD5 encryption.

E.g., Use the application ID as the username, encrypt the password in MD5 encryption and then use the encrypted password as the password when setting up HTTP Authentication.

Example Request:

```
Host: www.example.com

Accept: text/xml

Content-type: application/x-www-form-urlencoded

Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==


version=1.0&

address=tel:+447990123456&

address=tel:+447990121212&

message=Hello%20World&
```

NOTE:

- Address can either be tel:<phoneno> or list:<list_name>If 'tel' is used, the user can specify a list of phone numbers as shown in the above example whereas in list the user can only specify one list name where the name must be preconfigured, The list will contain a list of subscribers' phone numbers which to send the message.
- There is only one preconfigured list available for all applications, but if you want you can create new lists. For example, if your application is about sports, you can have separate lists for cricket, basketball and etc. The subscribers who prefer only a sub category such as Cricket can register under that particular list. These sub categories are represented using sub keyword when registering with an application, which will be discussed later on this document.
- The parameters are left un-encoded for readability purposes, for example, it would have to be encoded as;

  address=%2B447990123456

Following are the preconfigured list that is available for all applications.

List Name	**Description**
all_registered	Sends the message to all the users who have subscribed to your application.

You will be using a list if you are sending a broadcast message, which will send SMS to all the mobile numbers that are subscribed to your application. In order to send a broadcast message, subscribers must get registered with your application. This has basically three sections;

1. **Register**

   In order to get registered with the application subscriber can send a message as follows;

   reg <keyword> <sub keyword>

   After sending this message the subscriber will be registered with the application and a record will be added to the database. These registered numbers will be available to you whenever you will be sending a broadcast message.

2. **Unregister**

   A registered subscriber can un-register with the application using the following format of message;

   unreg <keyword> <sub keyword>

   When an un-registration is done the record of the particular subscriber will be removed from database and same time it will be sent to the application as well.

3. **Report abuse**

   If the application is not working as it is supposed to work, the subscribers can report abuse using the following message format.

   abuse <keyword> <sub keyword>

   In case of frequent reports of abuse the application will be kept under monitoring.

NOTE:

- The keyword and sub keywords will be provided to you when your application is ready to launch.
- If you want to get registered with the main application, sub keyword should be omitted.

When sending messages you can send only 160 characters at a time. If the message length is more than 160 characters it will be rejected.

When a message is sent from your application it would return a response depending on the success or the failure of the message.

Response types can be of three types:

- Success Message
- Failed Message
- Error Message

If the request contains all the parameters correctly and AVENTURA can send the message to the receiver successfully, then the following response will be received.

Example coding for a Success message:

```
HTTP/1.1 200 OK
ACCEPT: text/xml
Content-Length: 212
CONTENT_TYPE: application/x-www-form-urlencoded

<mchoice_sdp_sms_response>
   <version>1.0</version>
   <correlator>10051016580002</correlator>
   <status_code>SBL-SMS-MT-2000</status_code>
   <status_message>SUCCESS</status_message>
</mchoice_sdp_sms_response>
```

Even though the request contains valid parameters it does not mean that it is going to be success. For example, a correctly formatted request can not contain a none-existing Application id. These kinds of scenarios are considered as Failed Message.

Example coding for a Failed message:

```
HTTP/1.1 200 OK
ACCEPT: text/xml
Content-Length: 212
CONTENT_TYPE: application/x-www-form-urlencoded

<mchoice_sdp_sms_response>
   <version>1.0</version>
   <correlator>10051016580003</correlator>
   <status_code>CORE-SMS-MT-4001</status_code>
   <status_message>Requested ApplicationID PD_FUN is not found within the System</status_message>
</mchoice_sdp_sms_response>
```

Example coding for an Error message:

If the request contains invalid parameters or unformatted parameters, it will be considered as an Error Message.

Following is an example for an error message received when the user does not send the `message` parameter in the body.

```
HTTP/1.1 400 Bad Request
ACCEPT: text/xml
Content-Length: 269
CONTENT_TYPE: application/x-www-form-urlencoded

<mchoice_sdp_sms_response>
   <version>1.0</version>
   <correlator>10051017020003</correlator>
   <status_code>400</status_code>
   <status_message>Bad Request[Couldn't find parameter (message) in the request or it is blank]</status_message>
</mchoice_sdp_sms_response>
```

Following are the parameters you will need to set to send a message to a mobile phone.

Parameter	**Description**	**Mandatory / Optional**
version	mChoice AVENTURA HTTP REST API version that is currently used	A Mandatory field
address	List of addresses to which the SMS will be sent	A Mandatory field
message	The message to be sent	A Mandatory field

2.2 Receive SMS

When a SMS is received to your application a POST HTTP request will be sent with the following format.

```
POST  /{your application url}/ HTTP/1.1

Host: www.example.com

Accept: text/xml

Content-type: application/x-www-form-urlencoded


version=1.0&

address=+447990123456&

message=Hello%20World&

correlator=123456&
```

Following are the parameters which would be sent to your application when a SMS is received.

Parameter	Description	Always Present
version	mChoice AVENTURA HTTP REST API version that is currently used	Yes
Correlator	correlator ID for the requested message, this is a unique id which is used to uniquely identify every message	Yes
Address	Unique Identifier to identify the MSISDN which the message generated from. The length of this identifier is lager then 40 characters.	Yes
Message	Received message	Yes

2.3 Exceptions

A ServiceException may be thrown to indicate that a service-related error has occurred as a result of a client invocation on the service. The following error will be returned in the HTTP response:

IF an internal error, for this error user may retry

```
 HTTP/1.1 500 Internal Server Error

Content-Type: text/xml

<status_code>UNKNOWN</status_code>

<status_message>UNKNOWN</status_message>
```

If the operator’s service is unavailable the following error will be returned in the XML response:

```
HTTP/1.1 502 Bad Gateway

Content-Type: text/xml

<status_code>CORE-SMS-MT-4002</status_code>

<status_message> Requested Application's Service Provider is in the “inactive” state.</status_message>
```

A PolicyException may be thrown to indicate a fault relating to a policy associated with the service. For example, the following will be returned in the HTTP response if a minimum message length policy is enforced and the message length is less than the minimum.

```
HTTP/1.1 402 Forbidden

Content-Type: text/xml

<status_code>CORE-SMS-MT-4016</status_code>

<status_message>Seconds based rationale Traffic Limit enforcement failed the Request message</status_message>
```

If a parameter is specified incorrectly or is missing, for example, “mess” instead of “message” when sending an SMS, the following error is thrown:

```
HTTP/1.1 400 Bad Request

Content-Type: text/xml

<status_code> </status_code>

<status_message> </status_message>
```

If authorization credentials are incorrect or not present, the following error is thrown:

```
HTTP/1.1 401 Unauthorized

<status_code>UNAUTHORIZED-REQUEST</status_code>

<status_message>Request could not be authenticated, make sure proper
authentication parameters are sent</status_message>
```

Appendix A

- **Error Messages from Aventura**
    - Aventura Error Type

Parameter name	Type
status_code	Alphanumeric
status_message	Alphanumeric

NOTE: The Aventura Error would appear only in case of a System Failure. In the other cases Error codes will appear within the responses of the respective operations.

Following are the re-triable and non-retriable error codes.

Status Code	Explanation	Recommended handling
SBL-SMS-MT-2000	Success	Should not retry
SBL-SMS-MT-5000	This code indicates that the request has failed due to the unexpected error, while trying to submit the request to the SMSC. E.g., When request cannot be submitted to the SMSC, temporary internal errors before submitting it to SMSC	Should retry
SBL-SMS-MT-5001	Submit Response timeout	Should retry
SBL-SMS-MT-5002	Submit failed after maximum number of retries	Should not retry
SBL-SMS-MT-5004	SMSC system error, This error occurs when the server fails to complete an apparently valid request.	May retry

SBL-SMS-MT-5005	SMSC server busy, as a result of SMSC Message Queue being full	This error might occur when the queue for a particular MSISDN is full. So request should not be retried for that particular MSISDN. But request can be sent for the other MSISDNs.
SBL-SMS-MT-5006	Unknown Error at SMSC, Internal Error at SMSC	Should not retry
SBL-SMS-MT-5007	SMS message submission failed	Should not retry
SBL-SMS-MT-5008	Unexpected error : Detail: {Code}-{ErrorDetail}, {Code} is the SMPP Error Code and the {ErrorDetail} would have the SMPP Error Detail	Should not retry

The following are some common Error Codes that would be return when validating a request.

Status Code	**Status Message**	**Recommended handling**
CORE-SMS-MT-4000	Requested MT request was successfully processed.	
CORE-SMS-MT-4001	Requested ApplicationID is not found within the System.	Service Provider should verify the ApplicationID in the request. It should be the same as the registered ApplicationID which is registered with the AVENTURA system.
CORE-SMS-MT-4002	Requested Application's Service Provider is in the "inactive" state.	Service Provider has to contact the respective personnel and inquire about this issue.

CORE-SMS-MT-4003	Requested Application is not in the appropriate state to accept the requests, 1) Application is not in either of "Limited-Production","Active-Production" state to accept the requests for the production. 2) Application is not in "Active-Testbed" state to accept the TestBed requests.	Service Provider have to contact the respective personnel, and inquire about this issue
CORE-SMS-MT-4004	IP_Address, which the request originates from, is not listed within the allowed_host_addresses list.	Service Provider have to contact the respective personnel, and inquire about this issue
CORE-SMS-MT-4005	Given Password in the request is not matching with the password configured in the SLA.	Service Provider has to contact the respective personnel from, and inquire about this issue.
CORE-SMS-MT-4006	MSISDN, which is in the request, is blacklisted.	Same request cannot be retried.
CORE-SMS-MT-4007	Request has more concatenated message parts than the allowed message parts in the system.	Same request cannot be retried.
CORE-SMS-MT-4009	Request message has more recipients than the allowed no. of recipients within the system.	Application may resend the request after converting the request in to multiple message requests.
CORE-SMS-MT-4010	MSISDN which is in the request is in invalid state.(MSISDN may be in a Blocked State)	Same request cannot be retried.
CORE-SMS-MT-4011	MSISDN which is in the request is invalid. (MSISDN may have invalid length of digits)	Same request cannot be retried.
CORE-SMS-MT-4012	MT flow is not allowed for this Application	Same request cannot be retried.

CORE-SMS-MT-4015	MSISDN which is in the request is not listed within the White-User List **NOTE:** it will happen when the Application is in the Limited-Production state.	Application may resend the request using the MSISDN, which is in the assigned White-User List.
CORE-SMS-MT-4016	Seconds based rationale Traffic Limit enforcement failed the Request message. **NOTE:** This enforcement may have been enforced at SP level OR App level.	Application may resend the request after a small period of time.
CORE-SMS-MT-4017	Days based rationale Traffic Limit enforcement failed the Request message. **NOTE:** This enforcement may have been enforced at SP level OR App level.	Application may not resend the request during that particular day, so it can be only retried on the following day.
CORE-SMS-MT-4018	Requested category of user is not allowed to use the application	Same request cannot be retried.
CORE-SMS-MT-4020	Sender Address is not provided. **NOTE:** Sender Address can come in the request or it can be set as the default_sender_address, if both the values are not provided this error will be thrown.	
CORE-SMS-MT-4021	Sender Address is not validated by the sender Address aliasing field.	
CORE-SMS-MT-4024	Subscriber has insufficient balance amount	

CORE-SMS-MT-4030	Internal Error occurred	Internal Error in AVENTURA, same request can be retried.
CORE-SMS-MT-4037	Message size exceeded limit	
CORE-SMS-MT-4046	Tariff amount must be available in the request	Same request cannot be retried
CORE-SMS-MT-4049	Network Capability Service unavailable	Application may resend the request to the AVENTURA system.
CORE-SMS-MT-4061	Tariff not expected in the request	
CORE-SMS-MT-4063	MSISDN is not allowed for the operation	
CORE-SMS-MT-4120	All the recipients are not allowed. Check the individual delivery statuses	Same request cannot be retried
CORE-SMS-MT-4200	Number Checking Error : Detail: {ErrorDetail}	Number Checking Error, {ErrorDetail} will give the reason for the failure. Same request should not be retried.

Following Error codes will be returned for any unformatted message. None of the Error Codes should be retried by the User.

Error Code	**Description**
400	Bad Request. Request missing required parameters or parameter format is wrong.
401	Unauthorized. Authentication details missing or unformatted authentication details exists.
500	Internal Server Error

Appendix B

Base 64 encoding

The base 64 encoding is designed to arbitrary sequence of octets in a form that requires case sensitivity but need not be human readable.

This encoding method is based on 64 characters. (10 digits, 26 lower case characters, 26 upper case characters as well as '+' and '/'

In base 64 encoding, it takes the ASCII values of the characters that you enter and store as bytes. And convert to base 2 each consisting of 8 bits.
From the bit stream created, six by six bits are taken and converted again to decimal. Then you can get the Base 64 index of the decimal value from base 64 index table.

MD5 Encryption

MD5 is a formula to take a message of an arbitrary length, and create a 128-bit "fingerprint" or "message digest" of the message. It is a way to verify data integrity and often used to store user passwords and other sensitive data.
When you encrypt your password using MD5, it will create a fingerprint of the message but it will not include the original message. Therefore your password will not be decrypted by anyone else.