Genesys SMS Aggregation Service HTTP API User Guide

Genesys SMS Aggregation Service 2016CD

1/5/2022

Table of Contents

Genesys SMS Aggregation Service HTTP API User's Guide

Genesys supports sending and receiving messages via a RESTful HTTP API. To connect to this API, you must connect using either a VPN or an HTTPS connection.

The client should provide Genesys a URL to which Genesys can send MOs and DRs using an HTTP POST.

This API supports three types of messages:

- Mobile Terminated messages (MT)—An MT is destined for a handset.
  1. The client sends the MT request to the Genesys SMS Aggregation Service.
  2. Genesys acknowledges this request in the HTTP response.
  3. The message is then queued for delivery to the handset.

     The client can use either the POST or the GET method to send MTs to a Genesys URL. All the parameters and values in the query strings should be URL-encoded using the UTF-8 character set.
- Delivery Receipt (DR) messages—A DR provides a status update for an MT. Genesys sends the DR to a client URL.
  - DRs have to be requested during the MT API request.
  - DRs are not supported on all networks and/or handsets.
- Mobile Originated (MO) messages—An MO is a message Genesys receives from a handset.
  - Genesys sends the MO to a client URL

> **Important**
>
> In order to make full use of the high availability and disaster recovery features of the SMS Aggregation Service, clients should ensure they obey the DNS TTL provided by the authoritative server.

Sending a Mobile Terminated (MT) Message

The URL where you should send MT requests is:

- `https://smsc-api.genesyscloud.com/httpapi/receiver`

The table below lists the expected query parameters:

Name	Type	Mandatory?	Description
login	String	Yes	Supplied by Genesys
password	String	Yes	Supplied by Genesys
phone_number	String	Yes	The destination address for this message. *Must* be numeric and in international format. For example, a US phone number should be formatted as `12025550100`, and a UK phone number should be formatted as `447700900000`.
sender	String	Yes	Source address for this message. Must be enabled on your account.
carrier	String	No	If a Carrier ID is supplied, Genesys will trust it. See the Reach list for Carrier ID definitions. If a Carrier ID is not supplied Genesys will make its routing determination based upon the Country Code included in the phone_number parameter (see example above).
message	String	Yes	The message body.
message_id	String	Yes	A unique identifier for this message. Can be made optional if you cannot provide a unique message_id per request. Please contact your account manager.
receipt	String	No	The valid option values

Name	Type	Mandatory?	Description
			are `TRUE`, `FALSE`. FALSE (default)—No receipt is needed. TRUE—The client is asking for any DRs to be sent to their configured URL.
app_id	String	No	An optional parameter that you can provide, which will be returned in a field of the same name with Delivery Receipts
conversation_id	String	No	An optional parameter that you can provide, which will be returned in a field of the same name with Delivery Receipts
priority	String	No	The priority of the message. Valid values are: `BULK_LOW`, `BULK`, `INTERACTIVE_LOW`, and `INTERACTIVE`. If this parameter is not specified, the default for the account (BULK) is used. Use of an invalid priority results in an error.
mms_attachment[n].content_type	String	No	If this is an MMS message, the MIME type of the *n*th attachment. MMS attachments must be numbered consecutively starting with `0` (see the Important note below).
mms_attachment[n].content	String	No	If this is an MMS message, the Base64-encoded binary content of the *n*th attachment (see the Important note below).

Important

If your `mms_attachment[0].content_type` is given as any type of `multipart/*` content-type then the system does not automatically smil wrap the given content. If it

> is anything else, the given content is automatically smil wrapped.

HTTP API Response

After Genesys validates the MT request, an HTTP 200 response is sent to the client. The body of this response is in the form of a parameter `result_code`. This code indicates the acceptance state of MT request and has the following values:

Code	Description	Explanation
0	Queued	Message has been accepted for processing.
1	Request Invalid	Required parameters are missing.
2	Authentication Failed	The user credentials are not correct or do not exist.
3	Carrier ID Invalid	The carrier ID is not an integer or there is no mapping for this carrier ID.
4	Source address not allowed	The shortcode is not provisioned for that account
5	Source address denied	The provided shortcode is not registered for this client.
6	Throttled	The request rate is higher than the amount provisioned for the account. Lower the request rate.
7	Duplicate Message	This message has the same message_id as a previously accepted request
8	Message body too long	The length of the message to be sent is greater than allowed for this client.
98	Internal server error	There is an internal problem.
99	Any other error	Any errors not specified by one of the other result codes.

MT Retries

Depending on the result code, the client can retry the request. However, note the following recommendations:

- Do not retry the message immediately, but wait a short period of time, such as 30 seconds.

- Do not retry the message indefinitely. Stop retrying after a reasonable number of attempts, such as 10.
- If, when submitting an MT, the connection drops before the response is read (and therefore it is unknown whether the message succeeded or not), retry the message without changing the ID. If the Genesys server successfully processed the message on the previous attempt, it will respond with result code 7 (duplicate message ID). Otherwise, it will process the message normally.

Receiving a Delivery Receipt (DR)

Genesys sends a Delivery Receipt (DR) to a client-provided URL via a HTTP POST.

Genesys may send the request from several IP address ranges. Moreover, Genesys may send concurrent requests.

Delivery Receipt Parameters

This request is sent using the following parameters:

Parameter	Value
type	receipt
message_id	The message_id sent to Genesys in the original MT request.
message_state	The codes are defined in the Status Table, below.
result_code	The result code provides more detailed information on the status code. This is discussed in the Result Code section below.
app_id	If a value for app_id had been provided in the original MT request.
conversation_id	If a value for conversation_id had been provided in the original MT request.

Message State

Code	Description
1	The message is en-route to the handset.
2	The message has been delivered to the handset.
3	The message expired before reaching the handset.
4	The message was deleted by an Aggregator.
5	The message is undeliverable to the handset.
6	The message has been accepted by an Aggregator.
7	The state of the message is unknown.
8	The message has been rejected.

Result Code

In the case of failures the result code can be used to provide more detailed information. By default, the following detailed information is provided:

Code	Detailed Reason
0	Success.
1	The queued MT has been rejected by the upstream provider. Check the phone number and carrier and try again.
2	There has been an authentication failure.
3	The carrier is invalid.
4	The shortcode sent is an alpha numeric, which is not permitted on this carrier.
5	The shortcode is not allowed for this carrier.
6	The message has been rejected due to throttle errors. Please resubmit.
7	The message is a duplicate.
8	The message body is too long.
98	There has been an internal service error. Try again and, if the problem persists, contact your account manager.
99	An unmapped error has occurred. Contact your account manager if you need more information.

Important

If you need additional result code information, contact your account manager.

Receiving a Mobile Originated (MO) Message

Mobile Originated messages (MOs) are delivered via HTTP or HTTPS.

Genesys may send the request from several IP address ranges, and also may send concurrent requests.

The query parameters are listed in the HTTP Parameter table below. If Genesys does not receive a 2XX response to an MO request, it will be retried. The same `message_id` is used on subsequent retries.

If a network error occurs before the server can read the HTTP response, the message will be redelivered, even though the client might have been able to process it successfully. As a result, it is important to check the `message_id` for uniqueness before processing the MO any further.

HTTP Parameter	Value
type	MO
message_id	A unique ID for this message.
shortcode	The shortcode for the message.
phone_number	The phone number from which the message originated.
carrier	The carrier's ID, from the Reach list.
message	The message body.
mms_attachment[n].content_type	If it is an MMS message, then the Base64 encoded content of the attachment.
mms_attachment[n].content	If it is a multimedia message, then the Base64 encoded bytes of the attachment.

MO and DR Retry Strategy

Genesys uses the HTTP response code sent by the client to the Genesys HTTP request to determine whether a message has been accepted. Anything in the 2XX range indicates that the client has accepted the message. All other response codes cause Genesys to retry the request. Genesys retries using the following setup:

- A failed request is retried every ten minutes until one of two things happen:
  - The retry is successful.
  - The request has been retried 25 times with no success. At this point Genesys stops trying to send this request

Sample Java code to interact with Genesys API

```
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;

import java.util.UUID;

public class Demo {

        public static void main(String[] args) throws IOException {

                String username         = "USERNAME";                                   // The
username provided to the client by Genesys
                String password                 = "PASSWORD";
// The password provided to the client by Genesys
                String message                  = "Hey";                                  // The
message to be sent.
                String messageId         = String.valueOf(UUID.randomUUID());        // A
unique message id.
                // In this example we use UUID to generate an ID, but any value can be used
as long as it's unique
                String phoneNumber         = "12345";                                   // The
phone number to which the message is to be sent.
                String shortCode         = "99222";                                  // The
shortcode on this message.
                String carrier                  = "99901";                                    //
The carrier ID can be found in the reach list.

                String genesysURL         = "https://smsc-api.genesyscloud.com/httpapi/
receiver";

                // QueryString to be sent to Genesys API
                String messageBody= URLEncoder.encode( "login", "UTF-8" ) + "=" +
URLEncoder.encode( username, "UTF-8" )
                + "&" + URLEncoder.encode( "password", "UTF-8" ) + "=" + URLEncoder.encode(
password, "UTF-8" )
                + "&" + URLEncoder.encode( "phone_number", "UTF-8" ) + "=" +
URLEncoder.encode( phoneNumber, "UTF-8" )
                + "&" + URLEncoder.encode( "shortcode", "UTF-8" ) + "=" + URLEncoder.encode(
shortCode, "UTF-8" )
                + "&" + URLEncoder.encode( "carrier", "UTF-8" ) + "=" + URLEncoder.encode(
carrier, "UTF-8" )
                + "&" + URLEncoder.encode( "message", "UTF-8" ) + "=" + URLEncoder.encode(
message, "UTF-8" )
                + "&" + URLEncoder.encode( "message_id", "UTF-8" ) + "=" + URLEncoder.encode(
messageId, "UTF-8" );

                URL url = new URL(genesysURL);
                HttpURLConnection conn = (HttpURLConnection)url.openConnection();
                conn.setDoOutput(true);
                conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
```

```
                conn.setRequestProperty("Accept", "text/plain");


                DataOutputStream out = new DataOutputStream(conn.getOutputStream());
                out.writeBytes(messageBody);
                out.flush();
                out.close();

                // Response Code. If not in 2XX region then the request failed
                if(conn.getResponseCode() < 200 || conn.getResponseCode() > 299){
                        System.out.print("The request failed. Please try again in some
time.");
                        System.exit(0);
                } else {
                        // Printing the Response
                        BufferedReader reader= new BufferedReader(new
InputStreamReader(conn.getInputStream()));
                        String response= reader.readLine();

                        // Getting the Result Code from the Response
                        Integer resultCode= Integer.parseInt(response.split("=")[1]);

                        switch(resultCode){
                        case 0:
                                System.out.println("Success: The message has been accepted
for processing.");
                                break;
                        case 1:
                                System.out.println("Failure: Either one(or more) require
parameters are missing. ");
                                break;
                        case 2:
                                System.out.println("Failure: The credentials provided are
either wrong or do not exist.");
                                break;
                        case 3:
                                System.out.println("Failure: The carrier ID is not an Integer
value or it does not exist in the reach list.");
                                break;
                        case 4:
                                System.out.println("Failure: The shortcode is not an
Integer.");
                                break;
                        case 5:
                                System.out.println("Failure: The shortcode is not registered
for the client.");
                                break;
                        case 6:
                                System.out.println("Failure: The client is trying to send
more messages in the alloted time than they are allowed.");
                                break;
                        case 7:
                                System.out.println("Failure: This message is a duplicate.");
                                break;
                        case 8:
                                System.out.println("Failure: The message is larger than
allowed for the client.");
                                break;
                        case 98:
                                System.out.println("Failure: Internal Server Error. Please
try again");
                                break;
                        case 99:
```

```
                                System.out.println("Failure: There was some error in
processing request. Please try again");
                            }
                    }
            }
}
```

Sample Python code to interact with Genesys API

```
import uuid
import urllib.parse
import http.client
def main():
    "Method to send HTTP requests to Genesys API"

    username        = "USERNAME"                        # The username provided to the client
by Genesys
    password        = "PASSWORD"                        # The password provided to the client
by Genesys
    message         = "Hey"                             # The message to be sent.
    messageId       = uuid.uuid1()                      # A unique message id.
                                                        # In this example we use UUID to
generate an ID, but any value can be used as long as it's unique
    phoneNumber     = "12345"                           # The phone number to which the
message is to be sent.
    shortCode       = "99222"                           # The shortcode on this message.
    carrier         = "99999"                           # The carrier ID can be found in the
reach list.

    genesysURL      = "https://smsc-api.genesyscloud.com"
    headers = {
            "Content-Type" : "application/x-www-form-urlencoded",
            "Accept" : "text/plain" }

    # QueryString to be sent to Genesys API
    messageBody = {
            "login" : username,
            "password" : password,
            "phone_number" : phoneNumber,
            "shortcode" : shortCode,
            "carrier" : carrier,
            "message_id" : messageId,
            "message" : message}
    messageBody= urllib.parse.urlencode(messageBody)

    # Sending the request to Genesys API
    conn= http.client.HTTPConnection(genesysURL)
    conn.request("POST","/httpapi/receiver", messageBody, headers)

    response= conn.getresponse()
    # Response Code. If not in 2XX region then the request failed
    if response.status < 200 or response.status > 299:
        print("Request not successful. Please try again.")
    else:
        resultCode = response.read().decode().split('=')[1]

    if resultCode == '0':
        print("Success: The message has been accepted for processing.")
    elif resultCode == '1':
        print("Failure: Either one(or more) require parameters are missing. ")
    elif resultCode == '2':
        print("Failure: The credentials provided are either wrong or do not exist.")
    elif resultCode == '3':
```

```
        print("Failure: The carrier ID is not an Integer value or it does not exist in the
reach list.")
    elif resultCode == '4':
        print("Failure: The shortcode is not an Integer.")
    elif resultCode == '5':
        print("Failure: The shortcode is not registered for the client.")
    elif resultCode == '6':
        print("Failure: The client is trying to send more messages in the allotted time than
they are allowed.")
    elif resultCode == '7':
        print("Failure: This message is a duplicate.")
    elif resultCode == '8':
        print("Failure: The message is larger than allowed for the client.")
    elif resultCode == '98':
        print("Failure: Internal Server Error. Please try again")
    else:
        print("Failure: There was some error in processing request. Please try again")


if __name__ == '__main__':
    main()
```