Web Service API

Reference Manual

for
LinkInSMS
By
Halmax Pty Ltd

Date Commenced: 3 August 2006

Last Modified: 28 July 2021

Version: 2.40

Table of Contents

Overview

The Web Service API allows a registered user to submit a variety of messages and retrieve reports using SOAP over HTTP.

The location of the Web Service, service description and WSDL can be found at

http://message-service.org/webservice.asmx

This document is technical in nature and it is assumed that the reader is familiar with SOAP and the implementation of web services.

A note on code samples: These have been provided in .Net c# and Java using callable methods; however, any high level SOAP toolkit should provide the same functionality. The web service is cross platform and conforms to the web service standards.

Web Methods

Authentication

All web methods require an authentication header consisting of username and password (and optionally username) to be submitted within the SOAP header.

Authentication Header User Credentials:

Property	Type	Required	Description
Username	String	Yes	Assigned username
Password	String	Yes	Assigned password
Usercode	String	No	In the case that the user credentials belong to an administrator, the usercode of the job to be submitted under or retrieved, or of the user to be modified or deleted must be provided.

SubmitJob

This method is used to submit a SMS job with LinkInSMS functionality to the broadcast messaging system.

Expects:

Job entity conforming to the following type:- SmsJob

Returns:

Integer representing the job id or an exception on error.

Job

Job is an abstract class – submitted jobs must be SmsJob

Properties:

Property	Type	Required	Description
Name	String	Yes	A user defined name for the job
Scheduled	DateTime	No	A future date and time for the job to be activated
RepeatInterval	Integer	No	Causes a scheduled job to repeat every **"RepeatInterval"** seconds
ListIds[]	Integer array	No*	An array of list id's if using lists already uploaded to the broadcast website.
Sender	Sender	No	Complex type consisting of sender information.
Recipients[]	Recipient array	No*	Array of message recipients.
AdditionalReportEmails	String	No	List of additional emails to be CC'd on the report email

* One or more ListIds OR one or more Recipient required. Note that you may not use a combination of both ListIds AND Recipient as the object model differs between the two.

SmsJob

Extends **Job**
Also sends as MMS if a file is attached.

Properties:

Property	Type	Required	Description
Name	String	Yes	A user defined name for the job
Scheduled	DateTime	No	A future date and time for the job to be activated
RepeatInterval	Integer	No	Causes a scheduled job to repeat every **"RepeatInterval"** seconds
ListIds[]	Integer array	No*	An array of list id's if using lists already uploaded to the messaging website.
Sender	Sender	No	Complex type consisting of sender information.
Text	String	Yes	SMS text. Restriction of 3060 characters.
IsTwoWay	Boolean	Yes	Is this a two way SMS job?
SMSResponseEmailAddress	String	No	Email address to send the text of the SMS replies to. If blank, the users main email address will be used.
ExpiryHours	Byte	No	If this is a two way SMS job, how many hours before job should expire?

BatchReplies	Boolean	Yes	Should replies be batched at expiry? If not, replies will be sent individually to Sender.ReplyTo value as they arrive. For Two Way Jobs only.
Recipients[]	SmsRecipient array	No	Required if no ListIds submitted. SmsRecipient derives from Recipient
CallbackURL	String	No	If present, a HTML post request will be sent to the specified URL when the system receives a 2 way SMS reply. (Also see below field)
CallbackOnSMSStatusUpdate	bool	No	If present, and CallbackURL is set, then callbacks will be made to the specified URL on SMS status updates as well as 2 way SMS replies.
UseTwoWayCustomSenderID	bool	No	If present, and the job is a 2 Way SMS, then the ReplyTo field given in your sender object will be used as the SMS sender id, instead of the messaging services automated reply handling numbers. This means the messaging service will not get replies and optouts. *(Not for general use)*
CutOffHour	String	No	The hour that sms's will not be delivered past. NOTE, some carriers will deliver up to an hour after this time, and some carriers might ignore it completely. (format: 13:00)
OptoutCode	String	No	If a reply to a 2 Way SMS contains this string, then the recipient is added to the users optout list.
Dedupe	Boolean	No	Dedupe the recipient set
MMSFile	File	No	Send job as MMS if included. Supported file types are: `.gif, .jpg, .jpeg, .mp3, .mp4, .3gpp`
LinkInSMSPage	String	No	A Page template that will be merged with the recipient fields to create a custom Link In SMS page. See LinkInSMS details below.
LinkInSMSSecQuestion	String	No	When using LinkInSMS, this question must be answered by the recipient before viewing the linked content
LinkInSMSSecAnswer	String	No	When using LinkInSMS, this answer must be entered by the recipient before viewing the linked content

Sender

Properties:

Property	Type	Required	Description
Name	String	No	Contact name of user submitting job. If not provided, the company name from the stored user profile will be used.
Company	String	No	Company name. If not provided, the company name from the stored user profile will be used. Company name will be mentioned in TTS broadcasts as part of the introductory message
ReplyTo	String	No	Reply email (for email job) or number/name (for SMS job). If not provided, details in stored user profile will be used. The reply email will be displayed in email broadcasts; the reply mobile number will be displayed in one way SMS jobs. Note, for SMS jobs, an alphanumeric sender id must be 11 characters or less.

File

Properties:

Property	Type	Required	Description
Name	String	Yes	File name including extension
Priority	Integer	No	Priority order for multiple files
Content[]	Byte array	Yes	Byte array of file content

Recipient

An abstract class representing a message recipient.

Properties:

Property	Type	Required	Description
Reference	String	No	User defined reference for this recipient
Title	String	No	Recipient title (e.g. Mr, Ms etc)
FirstName	String	No	First name of recipient
LastName	String	No	Last name of recipient
Destination	String	Yes	Destination – e.g. fax number, email address, phone number or mobile number.

PhoneRecipient

An abstract class representing a phone recipient. Extends **Recipient**

Properties:

Property	Type	Required	Description
Reference	String	No	User defined reference for this recipient
Title	String	No	Recipient title (e.g. Mr, Ms etc)
FirstName	String	No	First name of recipient
LastName	String	No	Last name of recipient
Destination	String	Yes	Valid phone, fax or mobile number

Valid number construction for Destination:-

USA and Canada number with area code	e.g. 5551231234
International number with “+”	e.g. +44212345678
International number with “0011”	e.g. 001144212345678

SmsRecipient

Extends **PhoneRecipient**

Properties:

Property	Type	Required	Description
Reference	String	No	User defined reference for this recipient
Title	String	No	Recipient title (e.g. Mr, Ms etc)
FirstName	String	No	First name of recipient
LastName	String	No	Last name of recipient
Destination	String	Yes	Valid phone style number
MergeField[]	MergeField array	No	A list of merge key/value pairs. This can include the LinkInSMS page content. See Link In SMS section.

MergeField

Properties:

Property	Type	Required	Description
Key	String	Yes	Key relating to the merge field in message to be replaced.
Value	String	Yes	Value to replace key with for specific recipient.

RetrieveReport

Use this method to retrieve information about a specific broadcast job by Job ID.

Expects:
Integer **jobId** representing the job to be retrieved.

Returns:
StatusReport entity

RetrieveDetailedReport

Use this method to retrieve a detailed job report by Job ID.

Expects:
Integer **jobId** representing the job to be retrieved.

Returns:
DetailedReport entity

StatusReport

Entity representing the status report of a job.

Properties:

Property	Type	Description
JobId	Integer	The job id to be retrieved.
JobName	String	The user assigned Job name
Submitted	Dateime	The datetime that the job was submitted to the network. This is the system time + System Timezone offset.
JobType	String	FAX, Email, Voice, SMS, Text-To-Speech
JobStatus	String	Scheduled, Pending, Broadcast Submitted, Complete, Closed, Queued
TotalRecipients	Integer	Total number of recipients submitted
SentRecipients	Integer	Total number successfully sent to
FailedRecipients	Integer	Total number of failed recipients
TotalCost	Double	Total cost of job (if complete or closed)
RelatedJobs	String	For repeat scheduled tasks, returns a comma separated string of job IDs that were started by the supplied JobID.
Pages	Integer	The number of Fax pages per recipient sent for fax job, or the number of SMS messages per recipient.
ResentAs	String	If a job has been resent to the failed or errored recipients, this will have the ID of the resend job.
ResendOriginalID	String	If this job is the resend of a previous job, it will have the id of the original job

DetailedReport

Extends **StatusReport**

Properties:

Property	Type	Description
JobId	Integer	The job id to be retrieved.
Submitted	Dateime	The datetime that the job was submitted to the network. This is the system time + System Timezone offset.
JobType	String	FAX, Email, Voice, SMS, Text-To-Speech
JobStatus	String	Scheduled, Pending, Broadcast Submitted, Complete, Closed, Queued
TotalRecipients	Integer	Total number of recipients submitted
SentRecipients	Integer	Total number successfully sent to
FailedRecipients	Integer	Total number of failed recipients
TotalCost	Double	Total cost of job (if complete or closed)
RelatedJobs	String	For repeat scheduled tasks, returns a comma separated string of job IDs that were started by the supplied JobID.
Pages	Integer	The number of Fax pages per recipient sent for fax job, or the number of SMS messages per recipient.
ReportRecipient[]	ReportRecipient array	Array of completed recipients for the job

ReportRecipient

Properties:

Property	Type	Description
Recipient	String	Recipient name
Reference	String	Recipient reference
Destination	String	For user lists only
Status	String	SENT, ERR etc.
Duration	Integer	N/A
Cost	Double	Cost of message
Keypress	Integer	N/A
Reply	String	2 Way SMS jobs only – user reply
Voicemail	Boolean	N/A

Using Merge Fields

Merge fields may be used in the SMS jobs only:

To indicate a merge field in your message, the field should be enclosed by double percentage characters – e.g. `%%MyField%%`

There are reserved merge fields which you may use without specifically creating using the merge key/value pairs. These are:

1. `%%Title%%` - recipient's title
2. `%%FirstName%%` - recipient's first name
3. `%%LastName%%` - recipient's last name
4. `%%Recipient%%` - will be replaced by recipient's full name
5. `%%Reference%%` - will be replaced by the recipient's reference
6. `%%Mobile%%` - will be replaced by the recipient's destination for SMS type jobs

Sample message:

Attention: `%%Recipient%%`

Dear `%%FirstName%%,`

According to our records you are currently overdue on payment for `%%BillName%%.` The amount owing is `%%BillAmount%%.`

Please organize payment before the due date of `%%DueDate%%.`

Sample Construction

```
<SmsRecipient>
      <Title>Mr</Title>
      <FirstName>Andrew</FirstName>
      <LastName>Citizen</LastName>
      <Reference>19462</Reference>
      <Destination>0412 345 678</Destination>
      <MergeField>
            <Key>BillName</Key>
            <Value>June Electricity Bill</Value>
      </MergeField>
      <MergeField>
            <Key>BillAmount</Key>
            <Value>$78.32</Value>
      </MergeField>
      <MergeField>
            <Key>DueDate</Key>
            <Value>1 August 2007</Value>
      </MergeField>
</SmsRecipient>
```

Code Samples

C# Examples

Notes:

The Broadcast Webservice should be setup as a "**Web reference**" pointing to http://message-service.org/webservice.asmx in your project. In recent versions of Visual studio, this is found under the "Advanced…" button in the "Service Reference" page.

```
public void TestSMS()
{
	List<SmsRecipient> recipientList = new List<SmsRecipient>();

	SmsRecipient recipient = new SmsRecipient();
	recipient.Title = "Ms";
	recipient.FirstName = "Annette";
	recipient.LastName = "Citizen";
	recipient.Reference = "Of the world";
	recipient.Destination = "+61 4 9876 5432";

	List<MergeField> fieldList = new List<MergeField>();
	MergeField mergeField = new MergeField();
	mergeField.Key = "FavouriteAnimal";
	mergeField.Value = "Cats";
	fieldList.Add(mergeField);

	recipient.MergeFields = fieldList.ToArray();
	recipientList.Add(recipient);

	SmsJob smsJob = new SmsJob();
	smsJob.Name = "SMS Test";
	smsJob.Sender = new Sender();
	smsJob.Sender.Company = "My Company";
	smsJob.Sender.Name = "Test Person";
	smsJob.Sender.ReplyTo = "+61 4 1234 5678";
	smsJob.IsTwoWay = false;
	smsJob.Recipients = recipientList.ToArray();
	smsJob.Text = "Test SMS message - your favourite animal is
	              %%FavouriteAnimal%%.";

	AuthenticationHeader authHeader = new AuthenticationHeader();
	authHeader.Username = "test";
	authHeader.Password = "test";

	BroadcastWebService service = new BroadcastWebService();
	service.AuthenticationHeaderValue = authHeader;
	int jobid = service.SubmitJob(smsJob);

}
```

```
public void TestTwoWaySMS()
{
      List<SmsRecipient> recipientList = new List<SmsRecipient>();

      SmsRecipient recipient = new SmsRecipient();
      recipient.Title = "Ms";
      recipient.FirstName = "Annette";
      recipient.LastName = "Citizen";
      recipient.Reference = "Of the world";
      recipient.Destination = "+61 4 9876 5432";

      List<MergeField> fieldList = new List<MergeField>();
      MergeField mergeField = new MergeField();
      mergeField.Key = "FavouriteAnimal";
      mergeField.Value = "Cats";
      fieldList.Add(mergeField);

      recipient.MergeFields = fieldList.ToArray();
      recipientList.Add(recipient);

      SmsJob smsJob = new SmsJob();
      smsJob.Name = "2 Way SMS Test";
      smsJob.Sender = new Sender();
      smsJob.Sender.Company = "My Company";
      smsJob.Sender.Name = "Test Person";
      smsJob.Sender.ReplyTo = "smsreplies@mycompany.com";
      smsJob.IsTwoWay = true;
      smsJob.BatchReplies = false;
      smsJob.ExpiryHours = 24;
      smsJob.Recipients = recipientList.ToArray();
      smsJob.Text = "Test SMS message - your favourite animal is
                     %%FavouriteAnimal%%. Reply to this to get 25% off
                     your next visit to Pet Care.";

      AuthenticationHeader authHeader = new AuthenticationHeader();
      authHeader.Username = "test";
      authHeader.Password = "test";

      BroadcastWebService service = new BroadcastWebService();
      service.AuthenticationHeaderValue = authHeader;
      int jobid = service.SubmitJob(smsJob);

}
```

```
public void GetReport()
{
      AuthenticationHeader authHeader = new AuthenticationHeader();
      authHeader.Username = "test";
      authHeader.Password = "test";

      BroadcastWebService service = new BroadcastWebService();
      service.AuthenticationHeaderValue = authHeader;
      StatusReport statusReport = service.RetrieveReport(1234);
}


public void GetDetailedReport()
{
      AuthenticationHeader authHeader = new AuthenticationHeader();
      authHeader.Username = "test";
      authHeader.Password = "test";

      BroadcastWebService service = new BroadcastWebService();
      service.AuthenticationHeaderValue = authHeader;
      DetailedReport statusReport =
                  service.RetrieveDetailedReport(1234);
}

public void TestCancelJob ()
{
      AuthenticationHeader authHeader = new AuthenticationHeader();
      authHeader.Username = "test";
      authHeader.Password = "test";

      BroadcastWebService service = new BroadcastWebService();
      service.AuthenticationHeaderValue = authHeader;
      bool success = service.CancelJob(1234);
}
```

Java Axis Examples

Connecting to the Broadcast Web Service using Java and Axis 1.4

This document assumes that java and axis are setup appropriately, and the user has some basic familiarity with them.

1. Create the Broadcast Web Service interface classes:

   To create the WebService class files and the service Stub run:

   ```
   java org.apache.axis.wsdl.WSDL2Java (WSDL-file-URL)
   ```

   as per the axis documentation, then import the resultant classes into your java project.

2. Update Broadcast Web Service stubs to provide the required Authentication Header.

   In the two stub files:
   `Broadcast_x0020_Web_x0020_ServiceSoapStub`
   `Broadcast_x0020_Web_x0020_ServiceSoap12Stub`

   Insert the below code lines after the statement:
   `setRequestHeaders(_call);`
   in the **`submitJob`**, **`retrieveReport`** and **`retrieveDetailedReport`** methods

   ```
   org.apache.axis.message.SOAPHeaderElement auth = new
   SOAPHeaderElement("http://message-service.org/","AuthenticationHeader");
   auth.addChildElement("Username","").addTextNode("test");
   auth.addChildElement("Password","").addTextNode("test");
   _call.addHeader(auth);
   ```

3. Main Sample Code

```
/*
 * Main.java
 *
 * Created on 24 November 2006, 20:51
 *
 */

package broadcasttest;

import org.message-service.webservice.*;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Calendar;
import java.util.LinkedList;
import org.apache.axis.types.UnsignedByte;

public class Main {

    public Main() {
    }

    public static void main(String[] args) {
        try {
            TestSMS()
            //GetReport();
            //GetDetailsReport();
        } catch (Exception e){
            System.err.println(e.toString());
        }
    }
```

```
    static public void TestSMS(){
        // Add Recipients
        LinkedList recipientList = new LinkedList();
        SmsRecipient recipient = new SmsRecipient();
        recipient.setTitle("Ms");
        recipient.setFirstName("Annette");
        recipient.setLastName("Citizen");
        recipient.setReference("Of the world");
        recipient.setDestination("+61 4 9876 5432");

        //Create and add merge fields for recipient, then record recipient
        LinkedList fieldList = new LinkedList();
        MergeField mergeField = new MergeField();
        mergeField.setKey("FavouriteAnimal");
        mergeField.setValue("Cats");
        fieldList.add(mergeField);
        recipient.setMergeFields((MergeField[])fieldList.toArray(new
MergeField[1]));
        recipientList.add(recipient);

        //Set Sender Details
        Sender sender = new Sender();
        sender.setName("Test Person");
        sender.setCompany("My Company");
        sender.setReplyTo("+61 4 1234 5678");

        //Setup SmsJob and add sender, recipients and files
        SmsJob smsJob = new SmsJob();
        smsJob.setName("SMS Test");
        smsJob.setSender(sender);
        smsJob.setIsTwoWay(false);
        smsJob.setRecipients((SmsRecipient[])recipientList.toArray(new
SmsRecipient[1]));
        smsJob.setText("Test SMS message - your favourite animal is
%%FavouriteAnimal%%.");

        Calendar nextWeek = Calendar.getInstance();
        nextWeek.add(Calendar.DAY_OF_WEEK, 7);
        smsJob.setScheduled(nextWeek);

        try {
            int iret = SendJob(smsJob);
            System.out.println(iret);
        } catch (Exception e) {
            System.err.println(e.toString());
        }
    }
```

```
    static public void TestTwoWaySMS(){
        // Add Recipients
        LinkedList recipientList = new LinkedList();
        SmsRecipient recipient = new SmsRecipient();
        recipient.setTitle("Ms");
        recipient.setFirstName("Annette");
        recipient.setLastName("Citizen");
        recipient.setReference("Of the world");
        recipient.setDestination("5551231234");

        //Create and add merge fields for recipient, then record recipient
        LinkedList fieldList = new LinkedList();
        MergeField mergeField = new MergeField();
        mergeField.setKey("FavouriteAnimal");
        mergeField.setValue("Cats");
        fieldList.add(mergeField);
        recipient.setMergeFields((MergeField[])fieldList.toArray(new
MergeField[1]));
        recipientList.add(recipient);

        //Set Sender Details
        Sender sender = new Sender();
        sender.setName("Test Person");
        sender.setCompany("My Company");
        sender.setReplyTo("smsreplies@mycompany.com");

        //Setup SmsJob and add sender, recipients and files
        SmsJob smsJob = new SmsJob();
        smsJob.setName("SMS Test");
        smsJob.setSender(sender);
        smsJob.setIsTwoWay(true);
        smsJob.setExpiryHours(new UnsignedByte(24));
        smsJob.setRecipients((SmsRecipient[])recipientList.toArray(new
SmsRecipient[1]));
        smsJob.setText("Test SMS message - your favourite animal is
%%FavouriteAnimal%%." +
                "Reply to this to get 25% off your next visit to Pet Care.");

        Calendar nextWeek = Calendar.getInstance();
        nextWeek.add(Calendar.DAY_OF_WEEK, 7);
        smsJob.setScheduled(nextWeek);

        try {
            int iret = SendJob(smsJob);
            System.out.println(iret);
        } catch (Exception e) {
            System.err.println(e.toString());
        }
    }
```

```
    static public void GetReport() throws Exception{

        // Make a service
        Broadcast_x0020_Web_x0020_Service service = new
Broadcast_x0020_Web_x0020_ServiceLocator();
        // Now use the service to get a stub which implements the SDI.
        Broadcast_x0020_Web_x0020_ServiceSoapBroadcastWebService =
service.getBroadcast_x0020_Web_x0020_ServiceSoap();

        StatusReport report = BroadcastWebService.retrieveReport(1234);

        System.out.println(report.getJobStatus());
    }



    static public void GetDetailsReport() throws Exception{
        // Make a service
        Broadcast_x0020_Web_x0020_Service service = new
Broadcast_x0020_Web_x0020_ServiceLocator();
        // Now use the service to get a stub which implements the SDI.
        Broadcast_x0020_Web_x0020_ServiceSoapBroadcastWebService =
service.getBroadcast_x0020_Web_x0020_ServiceSoap();

        StatusReport report = BroadcastWebService.retrieveDetailedReport(1234);

        System.out.println(report.getJobStatus());

    }



    static public int SendJob(Job oJob) throws Exception{
        // Make a service
        Broadcast_x0020_Web_x0020_Service service = new
Broadcast_x0020_Web_x0020_ServiceLocator();
        // Now use the service to get a stub which implements the SDI.
        Broadcast_x0020_Web_x0020_ServiceSoapBroadcastWebService =
service.getBroadcast_x0020_Web_x0020_ServiceSoap();

        int iRes = BroadcastWebService.submitJob(oJob);

        return iRes;
    }
}


    static public void CancelJob () throws Exception{
        // Make a service
        Broadcast_x0020_Web_x0020_Service service = new
Broadcast_x0020_Web_x0020_ServiceLocator();
        // Now use the service to get a stub which implements the SDI.
        Broadcast_x0020_Web_x0020_ServiceSoapBroadcastWebService =
service.getBroadcast_x0020_Web_x0020_ServiceSoap();

        Boolean success = BroadcastWebService.canceljob(1234);

        System.out.println(success);

    }
```

PHP Code Example

Connecting to the Broadcast Web Service using PHP and NuSoap

This document assumes that PHP is installed and configured, and the newsoap library is downloaded and installed.

Change '`./nusoap/lib/nusoap.php`' to match the location of your nusoap library.

Basic SMSJob example:

```
<html><body>
<?php
	require_once('./nusoap/lib/nusoap.php');

	$namespace = "http://message-service.org/";

	//Setup Header
	$hdrArr = array(
		'Username' => 'test',
		'Usercode' => 'test',
		'Password' => 'test');

	//Create properly typed header object
	$hdr = array('AuthenticationHeader' => new
soapval('AuthenticationHeader', 'AuthenticationHeader', $hdrArr, false,
$namespace));

	//Setup Job parameters
	$jobarr = array(
		'Name' => 'Test Job name',
		'Sender' => array(
			'ReplyTo' => '5551231234'
		),
		'Recipients' => array(
			'Recip_1' => array(
				'Destination' => '5551231234',
				'Reference' => 'testing'
			)
		),
		'Text' => 'Test message goes here.',
		'IsTwoWay' => false,
		'BatchReplies' => false,
		'ExpiryHours' => new soapval('ExpiryHours', 'unsignedByte', 0,
false, 'http://www.w3.org/2001/XMLSchema')  //Need this soapval, otherwise it
goes out as an int, not an unsignedbyte
	);

	//Create properly typed job object
	$smsJob = array(new soapval('job', 'SmsJob', $jobarr, false,
$namespace));

	//create client and get wsdl
	$client = new nusoap_client('http:// message-
service.org/webservice.asmx?WSDL', 'wsdl');

	$err = $client->getError();
	if ($err) echo '<h2>Constructor error</h2><pre>' . $err . '</pre>';

	//Do call
	$result = $client->call('SubmitJob', $smsJob, $namespace, false, $hdr);
```

```
        // Display the result
        print_r($result);
        // Display the request and response
        echo '<h2>Request</h2>';
        echo '<pre>' . htmlspecialchars($client->request, ENT_QUOTES) . '</pre>';
        echo '<h2>Response</h2>';
        echo '<pre>' . htmlspecialchars($client->response, ENT_QUOTES) .
'</pre>';
?>
</body></html>
```

Appendix

Valid number formats

Valid phone, fax or mobile number construction:-

USA or Canada number with area code	e.g. 5551231234
International number with “+”	e.g. +44212345678
International number with “0011”	e.g. 001144 212345678

Other number constructions are invalid.

Broadcast Status Codes

List Created	User has created their job, but has not yet submitted it.
Pending	For SMS only:- job is in the pending queue.
Retry	For SMS only:- aggregator cannot be contacted, job has been re-queued.
Queued By WS	For Web Service submissions only:- job has been queued and awaiting processing.
Test Sent	N/A
Scheduled	Job is scheduled for later date and time.
Broadcast Submitted	Job has been submitted and is currently in progress.
Complete	Job is complete.
Closed	Job has been closed (either completed or cancelled).
Cancelled	Scheduled job has been cancelled by user.

SMS Message Lengths

1 message	Up to 160 characters.
2+ messages	153 characters per SMS message. (so 400 characters would take 3 messages)

Current maximum is 3060 characters, for a 20 part SMS message

Send Codes

SENT	Message was successfully sent to the recipient.

SMS Specific Error Codes

DERR	Aggregator error.
UERR	User validation error.
TRAN	Transaction limit reached.
PERR	Invalid password.
SVRE	SMS server error - please retry later.
SNDE	Error in sender number.
RECE	Error in receiver number.
MSGE	No SMS message.
LENE	SMS message too long.
NUME	Sending number invalid.
OUTE	SMS outgoing update error.

Notes:

For SMS: Status SENT indicates that the message has been successfully submitted to the network. In some cases the message may fail after if the number does not exist or for some other network related issue. In these cases, the status will be updated accordingly up to four hours after submission.

Timezone Numeric codes

Used in the User object to set the users timezone.

Albania +1 GMT	1
Algeria +1 GMT	2
American Samoa -11 GMT	195
Andorra +1 GMT	3
Angola +1 GMT	4
Anguilla -4 GMT	5
Antarctica -2 GMT	6
Antigua And Barbuda -4 GMT	7
Argentina -3 GMT	8
Argentina - Western Province -4 GMT	9
Armenia +4 GMT	10
Aruba -4 GMT	11
Ascension Island0 GMT	12
Australia - AEST +10 GMT	14
Australia - Lord Howe Island +10.5 GMT	13
Australia - Northern Territory +9.5 GMT	15
Australia - Queensland +10 GMT	16
Australia - South Australia +9.5 GMT	17
Australia - Western Australia +8 GMT	18
Austria +1 GMT	19
Azerbaijan +3 GMT	263
Bahamas -5 GMT	21
Bahrain +3 GMT	22
Bangladesh +6 GMT	23
Barbados -4 GMT	24
Belarus +2 GMT	25
Belgium +1 GMT	26
Belize -6 GMT	27
Benin +1 GMT	28
Bermuda -4 GMT	29
Bhutan +6 GMT	30
Bolivia -4 GMT	31
Botswana +2 GMT	32
Brazil - Acre -5 GMT	33
Brazil - Atlantic Islands -2 GMT	34
Brazil - East -3 GMT	35
Brazil - West -4 GMT	36
Brunei +8 GMT	38
Bulgaria +2 GMT	39
Burkina Faso0 GMT	40
Burundi +2 GMT	41
Cambodia +7 GMT	42
Cameroon +1 GMT	43
Canada - Atlantic -4 GMT	44
Canada - Central -6 GMT	45
Canada - Eastern -5 GMT	46
Canada - Mountain -7 GMT	47
Canada - Newfoundland -3.5 GMT	48
Canada - Yukon & Pacific -8 GMT	49
Cayman Islands -5 GMT	52
Chad +1 GMT	53
Chile -4 GMT	56
China +8 GMT	57
Colombia -5 GMT	58
Cook Islands -10 GMT	60
Costa Rica -6 GMT	61
Croatia +1 GMT	62
Cuba -5 GMT	63
Cyprus +2 GMT	64
Czech Republic +1 GMT	65
Denmark +1 GMT	66
Djibouti +3 GMT	67
Dominica -4 GMT	68
Ecuador -5 GMT	70
Egypt +2 GMT	71
El Salvador -6 GMT	72
Equatorial Guinea +1 GMT	73
Eritrea +3 GMT	74
Estonia +2 GMT	75
Ethiopia +3 GMT	76
Falkland Islands -4 GMT	77
Fiji +12 GMT	78

Finland +2 GMT	79
France +1 GMT	80
French Guiana -3 GMT	81
French Polynesia -10 GMT	82
Gabon +1 GMT	83
Gambia0 GMT	84
Georgia +4 GMT	85
Germany +1 GMT	86
Ghana0 GMT	87
Greece +2 GMT	88
Greenland -3 GMT	89
Greenland - Scoresbysun -1 GMT	90
Greenland - Thule -4 GMT	91
Grenada -4 GMT	92
Guadeloupe -4 GMT	93
Guam +10 GMT	94
Guatemala -6 GMT	95
Guyana -3 GMT	97
Haiti -5 GMT	98
Honduras -6 GMT	99
Hong Kong +8 GMT	100
Hungary +1 GMT	101
Iceland0 GMT	102
India +5.5 GMT	103
Indonesia - Central +8 GMT	104
Indonesia - East +9 GMT	105
Indonesia - West +7 GMT	106
Iran +3.5 GMT	107
Iraq +3 GMT	108
Ireland0 GMT	109
Israel +2 GMT	110
Italy +1 GMT	111
Ivory Coast0 GMT	112
Jamaica -5 GMT	113
Japan +9 GMT	114
Jordan +2 GMT	115
Kazakhstan +6 GMT	116
Kenya +3 GMT	117
Kiribati +12 GMT	118
Kuwait +3 GMT	121
Kyrgyzstan +5 GMT	122
Laos +7 GMT	123
Latvia +2 GMT	124
Lebanon +2 GMT	125
Lesotho +2 GMT	126
Liberia0 GMT	127
Libya +2 GMT	128
Lithuania +2 GMT	129
Luxembourg +1 GMT	130
Macedonia +1 GMT	131
Madagascar +3 GMT	132
Malawi +2 GMT	134
Malaysia +8 GMT	135
Maldives +5 GMT	136
Mali0 GMT	137
Malta +1 GMT	138
Mariana Island +10 GMT	165
Marshall Islands +12 GMT	140
Martinique -4 GMT	141
Mauritania0 GMT	142
Mauritius +4 GMT	143
Mayotte +3 GMT	144
Mexico -6 GMT	145
Mexico - Baja Calif Norte -8 GMT	146
Mexico - Nayarit Sinaloa Sonora -7 GMT	147
Moldova +2 GMT	148
Monaco +1 GMT	149
Mongolia +8 GMT	150
Morocco0 GMT	151
Mozambique +2 GMT	152
Namibia +1 GMT	153
Nepal +5.75 GMT	155
Netherlands +1 GMT	156
New Caledonia +11 GMT	158
New Zealand +12 GMT	159
Nicaragua -6 GMT	160
Niger +1 GMT	161
Niue -11 GMT	163
Norfolk Island +11.5 GMT	164
North Korea +9 GMT	119
Norway +1 GMT	167
Oman +4 GMT	168

Pakistan +5 GMT	169
Palau +9 GMT	170
Panama -5 GMT	171
Papua New Guinea +10 GMT	172
Paraguay -4 GMT	173
Peru -5 GMT	174
Philippines +8 GMT	175
Poland +1 GMT	176
Portugal +1 GMT	177
Puerto Rico -4 GMT	178
Qatar +3 GMT	179
Reunion +4 GMT	181
Romania +2 GMT	182
Russia - zone eight +9 GMT	183
Russia - zone eleven +12 GMT	184
Russia - zone five +6 GMT	185
Russia - zone four +5 GMT	186
Russia - zone nine +10 GMT	187
Russia - zone one +2 GMT	188
Russia - zone seven +8 GMT	189
Russia - zone six +7 GMT	190
Russia - zone ten +11 GMT	191
Russia - zone three +4 GMT	192
Russia - zone two +4 GMT	193
Rwanda +2 GMT	194
Saint Helena0 GMT	212
Saint Kitts and Nevis -4 GMT	213
Saint Lucia -4 GMT	214
Saint Vincent Grenadines -4 GMT	216
San Marino +1 GMT	197
Saudi Arabia +3 GMT	199
Senegal0 GMT	200
Seychelles +4 GMT	203
Sierra Leone0 GMT	204
Singapore +8 GMT	205
Slovenia +1 GMT	206
Solomon Islands +11 GMT	207
Somalia +3 GMT	208
South Africa +2 GMT	209
South Korea +9 GMT	120
Spain +1 GMT	210
Sri Lanka +5.5 GMT	211
Sudan +2 GMT	217
Swaziland +2 GMT	218
Sweden +1 GMT	219
Switzerland +1 GMT	220
Syria +2 GMT	221
Taiwan +8 GMT	222
Tanzania +3 GMT	224
Thailand +7 GMT	225
Togo0 GMT	226
Tonga +13 GMT	227
Trinidad and Tobago -4 GMT	228
Tunisia +1 GMT	229
Turkey +2 GMT	230
Turkmenistan +5 GMT	231
Tuvalu +12 GMT	233
Uganda +3 GMT	234
United Arab Emirates +4 GMT	235
United Kingdom0 GMT	236
United Kingdom - Channel Islands0 GMT	54
United Kingdom - New Hebrides +11 GMT	237
United Kingdom - Northern Ireland0 GMT	238
Uruguay -3 GMT	239
USA - Alaska -9 GMT	240
USA - Aleutian -10 GMT	241
USA - Arizona -7 GMT	242
USA - Central -6 GMT	243
USA - Eastern -5 GMT	244
USA - Hawaii -10 GMT	245
USA - Indiana East -5 GMT	246
USA - Mountain -7 GMT	247
USA - Pacific -8 GMT	248
Uzbekistan +5 GMT	249
Vanuatu +11 GMT	250
Venezuela -4 GMT	252
Vietnam +7 GMT	253
Wallis And Futuna Islands +12 GMT	256
Yemen +3 GMT	257
Zambia +2 GMT	261
Zimbabwe +2 GMT	262

Callback System

Version 1.1

Introduction

The API callback system returns a simple Post request to a user's chosen URL containing details about the success or failure of a chosen message. For SMS, it will send a callback containing the reply details for a given 2 Way SMS reply, and optionally it can also do a callback when the SMS delivery/failure is confirmed. (See notes below)

There is one call back for each broadcast recipient, so a broadcast to 10 people should generate 10 status callback messages.

Configuration

The activation and setting of the callback URL is defined in two ways.

1. On a per user basis, and is set either via the user administration page by a system admin (field: "callbackURL"), or via the AddUser or ModifyUser method in the API (field "callbackURL").
2. Via the API, the callback can be set on a per broadcast basis using the "**CallbackURL"** parameter

If the callbackURL parameter is blank, no callback is attempted for that user/broadcast, otherwise a callback is sent to the given URL.
The Callback URL should be of the standard format. Eg: http://testdomain.com/testscript.php

For SMS, if a callback parameter is defined, then by default callbacks are only done for 2 way SMS replies. If you need callback on SMS status updates as well, then the additional Boolean parameter: **"CallbackOnSMSStatusUpdate"** can be set to true, either in the API SMSJob request, or in the user administration page.

Technical details

On receipt of a final message status report from the internal Telephony systems, an asynchronous Callback request is sent to the specified URL containing the following POST fields:

SMS Status Update:

BroadcastID: **1234567**
BroadcastName: **Test Broadcast**
Timestamp: **2010-09-20T23:04:56 10:00**
Reference: **TestCompany**
Recipient: **TestUser**
Destination: **001133123456789**
Status: **SENT**
HandsetDeliveryTimestamp: **2012-03-09 13:48:46**

SMS Reply Details:

BroadcastID: **1234567**
BroadcastName: **Test Broadcast**
Timestamp: **2010-09-20T23:04:56 10:00**
Reference: **TestCompany**
Recipient: **TestUser**
Destination: **001133123456789**
Response: **The msg returned by the recipient**

Link In SMS

Overview

Link In SMS is a feature where custom, user specific pages of html or text data can be presented through a simple SMS message. How it works is that for each SMS recipient a custom web page will be created and linked to in their SMS, allowing viewing of the page of html with a simple click. This allows a single SMS to communicate arbitrary large amounts of data. For example, a SMS broadcast could send out employee's individual upcoming monthly timetables, or their invested stock prices, or lists of overdue library books, etc. each accessed by a simple click in their SMS.

Implementation

There are two methods to use the Link In SMS System:

1. The Page data can be passed in full for each user.
2. A Link In SMS page template can be passed once for the job, and the pages generated by merging with the recipient fields.

Option 1:

With the first method of using Link In SMS through the API, the desired page data is added to the API request by adding the page data in a mergefield called "linkinsms" for each SMS recipient, and including the phrase %%linkinsms%% in your SMS message.

An example Link in SMS request would have recipient fields like:

```
Destination = '+098765432',
Reference = 'mytest',
FirstNane = 'John'
MergeField= [
      LinkInSMS = '<pageof html data for John's schedule…>'
]

Destination = '+098765431',
Reference = 'myothertest',
FirstNane = 'Jane'
MergeField= [
      LinkInSMS = '<pageof html data for Jane's schedule…>'
]
```

And the SMS message could be:

"Hi %%firstname%%. Please click on %%linkinsms%% to view your schedule for next week."

The resulting SMS received by the employee would look like:

"Hi Jane. Please click on http://lis.ms/3jKu738aU to view your schedule for next week."

And clicking this would take them to their schedule in the provided page data.

Option 2:

The second method of including a Link In SMS page is to populate the SMSJob field "LinkInSMSPage" with a template containing merge fields. This is appropriate for simple pages where there are minor differences between the recipients' pages.

For example:

```
SMSJob.LinkInSMSPage = "<p>Welcome %%firstname%%.</p><p>Please read the below
Terms and Conditions</p>
<p>… Long terms and conditions  …</p>"
```

Other details:

In both cases, if the "page content" is a just a URL, then when the recipient clicks on the link, they are redirected to the URL supplied

For example:

Page content = "http://HealthNews.com/latest_flu_info.html

Or the link could let them access custom pages just for them:

Page content = "http://myschedule.com/schedule.php?userid=123&auth=h6Fh89Uz

To reduce the amount of page data needed for each recipient, you can set Link In SMS header and footer values in your user profile. So you can have a Logo and other links automatically wrapped around any uploaded html content. This way you can just send the core content and all header, logo and styling are already done.