

CHANNEL MOBILE

SMS GATEWAY XML SPECIFICATION

V2.1

Table Of Changes

Version	Date	Description
v1.0	16/07/03	Initial Revision
v1.1	01/08/03	<ul style="list-style-type: none"> Added send_status attribute Added PARSE_ERRORS element Added compression instructions
v1.2	07/08/03	<ul style="list-style-type: none"> Fixes
v1.3	27/08/03	<ul style="list-style-type: none"> Fixes Added HTTP/HTTPS transports
v1.4	12/09/03	<ul style="list-style-type: none"> Added notes on HTTP Posting
v1.5	19/09/03	<ul style="list-style-type: none"> Added link for url-encoded format
v1.6	13/10/03	<ul style="list-style-type: none"> Added notes on XML encoding types (Section 7) DEPRECATED the EMAIL_VERBOSE for reply and reply_cc attribute. This was an oversight left over from the original SSL:55000 specification. It will work, but the data is encoded according to the old specification. This will be shortly replaced by an EMAIL_XML type which will be used to return developer level data.
v1.7	20/10/03	<ul style="list-style-type: none"> <i>reply_type</i> tag changed to <i>type</i> in documentation. Should not affect anyone, as <i>type</i> is the attribute that the server has been returning.
v1.8	13/11/03	<ul style="list-style-type: none"> Added a new reply type the to reply attribute that allows for HTTP pushes instead of polling the server Added notes in XML tags section about HTTP pushes. Added additional notes on URL-Encoding
v1.9	18/11/03	<ul style="list-style-type: none"> Added new tags for advanced scheduling features <ul style="list-style-type: none"> <i>time_zone</i> <i>send_on_weekends</i> <i>send_between_start</i> <i>send_between_end</i>
v1.10	08/12/03	<ul style="list-style-type: none"> XML via EMAIL now supports .xml attachments. Details can be found in the section entitled "Connecting to SMS Gateway".
V1.11	02/03/04	<ul style="list-style-type: none"> Added stricter destination number policies. See <i>to</i> tag Decreased the maximum schedule time to 14 days. This means you can only schedule a message a maximum of 14 days ahead of the time. See <i>send_at</i> tag.
v1.12	19/08/04	<ul style="list-style-type: none"> Added '&' to Section 6 – Indy patch for urlencoding. Added validity period specification (added ZZ for timezone)
v.1.13	09/03/05	<ul style="list-style-type: none"> SendStatus attribute added to RECEIVE_RESPONSE
V2		<ul style="list-style-type: none"> Override send_at at the child node <SMS_Send> Override send_before at the child node <SMS_Send>
V2.1	10/03/2010	<ul style="list-style-type: none"> Added ability to fetch shortcode requests from gateway

1.	Introduction.....	5
2.	Tag Properties.....	6
2.1	allow_reply.....	6
2.2	application.....	7
2.3	concatenation_level.....	8
2.4	delivery_report.....	9
2.5	extension.....	10
2.6	indication.....	11
2.7	password.....	12
2.8	reply.....	13
2.9	reply_cc.....	14
2.10	type.....	15
2.11	send_at.....	16
	* You can override send_at at the child node <SMS_Send>send_before.....	16
	send_before.....	17
	* You can override send_before at the child node <SMS_Send>send_between_end..	17
	send_between_end.....	18
2.12	send_between_start.....	19
2.13	send_on_weekends.....	20
2.14	send_status.....	21
2.15	sms_entry_ID.....	22
2.16	status_report.....	23
2.17	time_zone.....	24
2.18	to.....	25
2.19	to_name.....	26
2.20	uid.....	27
2.21	user.....	28
2.22	validity_period.....	29
2.23	version.....	30
3.	XML Elements.....	31
3.1	Sending Elements.....	31
3.2	Send Response Elements.....	33
3.3	Receive Elements.....	34
3.3.1	Receive Command for Shortcodes.....	34
3.4	Receive Response Elements.....	34
3.4.1	Receive Response for Shortcodes.....	35
3.5	Parse Errors Element.....	35
3.6	Compressing sends and receives.....	37
3.7	Send and Receive.....	39
3.8	HTTP Pushes.....	40
4	Notes and considerations when sending.....	41
4.1	Sockets and timeouts.....	41
5.	Connecting to SMS Gateway.....	42
5.1	HTTP.....	42
5.2	Email.....	42

6.	Notes on HTTP/S Post Requests.....	43
6.1	General Notes.....	43
6.2	Indy Components	43
7.	Notes on XML encoding.....	44

1. Introduction

The new format for sending an SMS to the SMS Gateway will consist of an XML document. The layout for this document will be;

1. Tag properties
2. Tags for sending
3. Tags for receiving
4. Tags for other operations

2. Tag Properties

2.1 ***allow_reply***

2.1.1 *Description*

Specifies a whether the server will forward replies to the message. The message will be stored but never forwarded. This does not apply to status reports or delivery reports.

2.1.2 *Use*

`allow_reply=<0/1>`

0 means do not allow replies
1 (default) means allow replies

2.1.3 *Notes*

2.1.4 *Example*

`allow_reply="0 "`

2.1.5 *See also*

2.2 application

2.2.1 Description

Specifies the ID of the application that is sending the SMS. This is used to block traffic from certain applications. The ID of the application will be provided for you by Channel Mobile.

2.2.2 Use

application=<application ID>

2.2.3 Notes

2.2.4 Example

application="50"

2.2.5 See also

version

2.3 *concatenation_level*

2.3.1 Description

An optional parameter, which specifies the level to which a long message will be concatenated.

2.3.2 Use

`concatenation_level=<numerical ID>`

where <numerical ID> is one of the following

ID	Meaning
0	No concatenation message will be cropped at 160 characters
1 (default if not specified)	Break-Apart: Messages are broken into multiple 160 messages
2	Full network concatenation. Multiple 153 character messages. NOTE: Some handsets do not support this level and may incorrectly display the message.

2.3.3 Notes

2.3.4 Example

`concatenation_level="1"`

2.3.5 See also

2.4 *delivery_report*

2.4.1 Description

Specified whether a delivery report should be generated

2.4.2 Use

`delivery_report=<report type ID>`

Type ID	
0	None
1	Generate a report

2.4.3 Notes

default it 0 (if unspecified).

Please note that specifying 1 does not produce a ME delivery report, but rather whether or not the message was delivered to the network. Additionally any failure to deliver to the network will result in a "failed" report being generated.

2.4.4 Example

`delivery_report="0"`

or

`delivery_report="1"`

2.4.5 See also

status_report

2.5 extension

2.5.1 Description

Specifies the extension that the SMS came from. This is used purely for itemised billing purpose

2.5.2 Use

extension:<extension >

2.5.3 Notes

2.5.4 Example

extension="123456"

or

extension="John Smith"

or

extension="john@smith.com"

2.5.5 See also

uid

2.6 *indication*

2.6.1 Description

An optional parameter that allows you to set how the SMS message is indicated on ME.

2.6.2 Use

indication=<indication ID>

where indication is one of the following

ID	Description
0 (default)	Message to be stored normally on ME
1	Flash message. This message is displayed directly on the ME screen. The message may be truncated depending on the model or display size. The message may or may not be stored on the ME depending on the implementation.

2.6.3 Notes

2.6.4 Example

indication="1 "

2.6.5 See also

2.7 password

2.7.1 Description

Specifies the password associated with the user account

2.7.2 Use

password=<password>

2.7.3 Notes

2.7.4 Example

password="secret01"

2.7.5 See also

user

2.8 *reply*

2.8.1 Description

Specifies the reply path that any replies to an SMS should take.

2.8.2 Use

`reply=<transport>:<data>`

where `<transport>:<data>` can be

`SSL:<user account>`
`EMAIL:<email address>`
`EMAIL_W_SUBJECT:<email address>:<subject ID>`
`HTTP:<URL>`

Where `<user account>` is any valid user account on the SMS gateway (usually the one sending the SMS).

Where `<email address>` is any valid SMTP destination email address

Where `<subject ID>` is an arbitrary subject line that will be sent with delivery report, status reports and replies. The `<subject ID>` will be concatenated with the relevant subject line using brackets. e.g. If `<subject ID>` was "test" then on a reply the subject line would be "SMS Reply (test)".

Where `<URL>` is a fully qualified http URL. Full compatibility with https certificates has not been established.

2.8.3 Notes

A message body contains only 1 *reply* header.

If a message does not have a reply tag then it will discard replies.

EMAIL and EMAIL_W_SUBJECT returns a friendly response while EMAIL_VERBOSE returns developer level responses.

2.8.4 Example

`reply="SSL:myaccount"`

OR

`reply="EMAIL:my_reply_email@myisp.com"`

OR

`reply="EMAIL_W_SUBJECT:my_reply_email@myisp.com:some subject line"`

OR

`reply="HTTP:http://www.mydomain.com/somepage.jsp"`

2.9 *reply_cc*

2.9.1 Description

An optional parameter that allows you to set an additional reply path for replies to be cc'ed to. This tag does NOT include status report and delivery reports.

2.9.2 Use

Parameters are the same as the *reply* tag

2.9.3 Notes

2.9.4 Example

```
reply_cc="EMAIL:someone@somewhere.com"
```

2.9.5 See also

reply

2.10 *type*

2.10.1 Description

Specifies the reply type

2.10.2 Use

`type =<reply type>`

Where <type> means the following

type	meaning
1	Received Message: A normal incoming reply
5	Send Failure (SOFT_REPORT) Generated by <i>delivery_report</i> tag
4	Send Succeeded (SOFT_REPORT) Generated by <i>delivery_report</i> tag
7	SMS_STATUS Report – Delivery Failed Generated by <i>status_report</i> tag
6	SMS_STATUS Report Delivery Report – Delivery Success Generated by <i>status_report</i> tag
8	SMS_STATUS Report Delivery Report – Temporary Error Generated by <i>status_report</i> tag

2.10.3 Notes

A reply type of 4 or 5 indicates a status of whether or not the message was delivered to the GSM network or not.

2.10.4 Example

`type =”0”`

2.10.5 See also

2.11 *send_at*

2.11.1 Description

Specifies a date at which the SMS must leave the gateway. This tag is used to schedule a message on the server. The message will wait on the server and be sent from the server at time *send_at*. A maximum schedule period may be imposed on message by the server and is tentatively set at 14 days. Messages, which try to be scheduled with a date greater than 14 days in advance, will receive a PARSE ERROR from the server describing the error.

2.11.2 Use

send_at=:<GMT date and time>

where <GMT date and time> is a date and time of the format YYYY-MM-DD HH:MM:SS

2.11.3 See also

send_before

* You can override *send_at* at the child node <SMS_Send>

send_before

2.11.4 Description

Specifies a date by which the SMS must have left the gateway. It is used to specify the latest time by which a scheduled message must be handed onto the global GSM network

2.11.5 Use

`send_before=<GMT date and time>`

where <GMT date and time> is a date and time of the format YYYY-MM-DD HH:MM:SS

2.11.6 Notes

This tag does not imply deliver before a certain time (see *validity_period* for that)

2.11.7 Example

`send_before="2002-12-25 17:55:00"`

2.11.8 See also

send_at

* You can override `send_before` at the child node `<SMS_Send>`

send_between_end

2.11.9 Description

This tag form the end time for when a message can be sent between. For instance, if you want a message to only go out between 15h00 and 19h00 then the *send_between_end* value would be 1140 (19 x 60).

2.11.10 Use

send_between_end=<minutes from 00h00>

2.11.11 Notes

requires

- *time_zone*
- *send_between_start*

2.11.12 Example

send_between_end='1140' //19h00

2.11.13 See also

time_zone
send_between_start

2.12 *send_between_start*

2.12.1 Description

This tag form the start time for when a message can be sent between. For instance, if you want a message to only go out between 15h00 and 19h00 then the *send_between_start* value would be 900 (15 x 60).

2.12.2 Use

send_between_start=<minutes from 00h00>

2.12.3 Notes

requires

- *time_zone*
- *send_between_end*

2.12.4 Example

send_between_start='900' //15h00

2.12.5 See also

time_zone
send_between_end

2.13 *send_on_weekends*

2.13.1 Description

This tag is used to specify whether this message can be send on a weekend. The weekend is considered to be from Friday 17h00 to Monday 08h00. The time it is send from the gateway is dependent on the value of the *time_zone* tag.

2.13.2 Use

`send_on_weekends=<0/1 >`

0 = No

1 = Yes

2.13.3 Notes

2.13.4 Example

`send_on_weekends='1' //this message can be sent on the weekend`

`send_on_weekends='0' //this message should not be sent on the weekend`

2.13.5 See also

time_zone

2.14 *send_status*

2.14.1 Description

Specifies the status of a message as it is sent.

2.14.2 Use

0	OK
1	Authentication Failed
2	Account Closed
3	Generic Failure
4	Destination Address blocked
5	No Credits
6	Reserved
7	Reserved
8	Decryption Failed
9	Account Not Found
10	Client UID Duplicated (i.e. same uid, destination address and account as another SMS).
11	Application ID incorrect

2.14.3 Notes

2.14.4 Example

2.14.5 See also

2.15 sms_entry_ID

2.15.1 Description

Specifies an ID of the message as generated on the server. Used for checking for duplicates when doing a receive.

2.15.2 Use

sms_entry_ID=<numerical ID>

2.15.3 Notes

2.15.4 Example

sms_entry_ID="1234567"

2.15.5 See also

2.16 *status_report*

2.16.1 Description

An optional parameter, which specifies that a ME SMS_STATUS_REPORT request should be added to this message. This report is generated by the network. This tag refers to the status of the message while on the network and is different to the *delivery_report* tag.

2.16.2 Use

`status_report=<0/1>`

Default is 0. 1 meaning TRUE

2.16.3 Notes

There may be additional costs associated with using this tag

2.16.4 Example

`status_report="1"` (*generated a status report from the network*)

2.16.5 See also

delivery_report

2.17 *time_zone*

2.17.1 Description

This tag is used with the following tags

- *send_on_weekends*
- *send_between_start*
- *send_between_end*

It is used to calculate the correct times to send an SMS.

2.17.2 Use

`time_zone=<time zone in minutes from GMT>`

2.17.3 Notes

2.17.4 Example

```
time_zone='120' //GMT + 2  
time_zone='-60' //GMT - 1
```

2.17.5 See also

send_on_weekends
send_between_start
send_between_end

2.18 to

2.18.1 Description

Specifies a number that should be sent to.

2.18.2 Use

to=<cellular number>

2.18.3 Notes

The cellular number should contain a full country code and a +.

The cellular number should be in the following format +<country code><network code><number>

As of 02/03/2004 we will be enforcing the following rules.

1. By default numbers not formatted correctly WILL NOT BE SENT and you WILL BE CHARGED. The correct format for a message is +<country code><network code><number>. e.g. +27821234567. There are no exceptions.
2. Where possible we will try and anticipate what could go wrong with the formatting and apply the following rules (these are just guideline and should NOT be relied upon)
 - a. numbers not starting with a + will have a + attached. If the number starts with a 0 it will not be sent. e.g. 2782123 will become +2782123, however numbers like 082123 or 09441234567 will NOT BE SENT.
 - b. numbers with punctuation (excluding +), spaces and letter will be stripped. e.g. "+27 82 123" will become +2782123; "+27 select number" will become +27.

2.18.4 Example

to="+27821231234"

2.19 to_name

2.19.1 Description

Specifies the “friendly” name of the recipient

2.19.2 Use

to_name=<friendly name>

2.19.3 Notes

Used in replies to specify who the message came from

i.e. Instead of “You have a message from +55 555 1234” the message will read “You have a message from +55 555 1234 (John Smith)”

2.19.4 Example

to_name=”John Smith”

2.19.5 See also

2.20 uid

2.20.1 Description

Specifies a unique identifier from the clients system that would uniquely identify the SMS. The tag is used in a reply message to specify to which SMS the reply pertains.

If a message from a client is received with a duplicate uid to the same destination number, then the message is marked as duplicate and not sent.

2.20.2 Use

uid=<uuid>

Where <uuid> is an alpha numeric value.

2.20.3 Notes

The uid needs only be unique on the client system.

2.20.4 Example

uid="12345"

or

uid="xyz123"

2.21 user

2.21.1 Description

Specifies the user account to send from. This is for authentication.

2.21.2 Use

user=<user account>

Where <user account> is any valid user account on the SMS gateway

2.21.3 Notes

A message body contain only 1 *user* header

2.21.4 Example

user="myaccount01"

2.21.5 See also

password

2.22 *validity_period*

2.22.1 Description

Specifies an absolute time at which an SMS is no longer valid

2.22.2 Use

`validity_period=<absolute time>`

Where <absolute time> is in the format YYMMDDHHMMSSZZ which represents a GMT time

2.22.3 Notes

default is maximum validity period for network.

ZZ should be the validity period timezone offset in 15minute intervals eg. for GMT+2, ZZ=8.

2.22.4 Example

```
//2002 Dec 25 23h00:00 GMT  
validity_period="021225230000"
```

2.22.5 See also

2.23 version

2.23.1 Description

Specifies the version of the application that is sending the SMS

2.23.2 Use

version=<version number>

where <version number> is of the form <major>.<minor>

2.23.3 Notes

Must be used with an application tag

2.23.4 Example

version="1.1"

or

version="10.3"

2.23.5 See also

application

3. XML Elements

If should be noted that the root for the XML document is an XML element which is different from the <?xml?> version tag. There can only be one root element per document.

A (n) after a tag denotes that there can or should be more than 1 of these children nodes.

3.1 Sending Elements

Tag	Properties	Optional
SENDER		N
SENDTO		N
SENDTO	allow_reply	Y
SENDTO	application	N ¹
SENDTO	concatenation_level	Y
SENDTO	delivery_report	Y
SENDTO	extension	Y
SENDTO	indication	Y
SENDTO	password	N ¹
SENDTO	reply	N ¹
SENDTO	reply_cc	Y
SENDTO	send_at	Y
SENDTO	send_before	Y
SENDTO	status_report	Y
SENDTO	user	N ¹
SENDTO	validity_period	Y
SENDTO	version	Y
SMSLIST		N
SMS_SEND(n)	allow_reply	Y
SMS_SEND(n)	application	N ¹
SMS_SEND(n)	concatenation_level	Y
SMS_SEND(n)	delivery_report	Y
SMS_SEND(n)	extension	Y
SMS_SEND(n)	indication	Y
SMS_SEND(n)	password	N ¹
SMS_SEND(n)	reply	N ¹
SMS_SEND(n)	reply_cc	Y
SMS_SEND(n)	send_at	Y
SMS_SEND(n)	send_before	Y
SMS_SEND(n)	status_report	Y
SMS_SEND(n)	to	N
SMS_SEND(n)	to_name	Y
SMS_SEND(n)	uid	N
SMS_SEND(n)	user	N ¹
SMS_SEND(n)	validity_period	Y
SMS_SEND(n)	version	Y

N¹: The property must either be set in the SEND_BATCH defaults or on every SMS_SEND element.

To send a message with the XML you would format the XML like this.

<XML>

```

    <SENDERBATC>
      <SMSLIST>
        <SMS_SEND user="user" password="password" to="+27123000001"
uid="1" application="50" reply="EMAIL:john@smith.com">This is a test message</SMS_SEND>
      </SMSLIST>
    </SENDERBATC>
  </XML>

```

To send to more than one number you could send like this

```

<XML>
  <SENDERBATC>
    <SMSLIST>
      <SMS_SEND user="user" password="password" to="+27123000001"
uid="1" application="50" reply="EMAIL:john@smith.com">This is a test message</SMS_SEND>
      <SMS_SEND user="user" password="password" to="+27123000002"
uid="2" application="50" reply="EMAIL:john@smith.com">This is a test message</SMS_SEND>
      <SMS_SEND user="user" password="password" to="+27123000003"
uid="3" application="50" reply="EMAIL:john@smith.com">This is a test message</SMS_SEND>
    </SMSLIST>
  </SENDERBATC>
</XML>

```

This is rather inefficient, so by using defaults you could in the SEND_BATCH tag you can decrease the total data needed to send like this:

```

<XML>
  <SENDERBATC user="user" password="password" application="50"
reply="EMAIL:john@smith.com">This is a test
    <SMSLIST>
      <SMS_SEND to="+27123000001" uid="1"/>
      <SMS_SEND to="+27123000002" uid="2"/>
      <SMS_SEND to="+27123000003" uid="3"/>
    </SMSLIST>
  </SENDERBATC>
</XML>

```

3.2 Send Response Elements

Tags	Properties	Optional
SENDERBATCHRESPONSE		N
SMS_SEND_RESPONSE(n)		
	uid	N
	to	N
	send_status	N

After a SENDERBATCH the response would be like this

```
<XML>
  <SENDERBATCHRESPONSE>
    <SMS_SEND_RESPONSE uid="1" to="+275550000001"
send_status="1">authentication failed</SMS_SEND_RESPONSE>
    <SMS_SEND_RESPONSE uid="2" to="+275550000002"
send_status="2">Duplicate UID</SMS_SEND_RESPONSE>
    <SMS_SEND_RESPONSE uid="3" to="+275550000003"
send_status="3">OK</SMS_SEND_RESPONSE>
  </SENDERBATCHRESPONSE>
</XML>
```

3.3 Receive Elements

Tags	Properties	Optional
RECEIVE		N
	user	N
	password	N

To receive messages off the server the XML should be formatted as such

```
<XML>  
  <RECEIVE user="12345678" password="password"></RECEIVE>  
</XML>
```

or

```
<XML>  
  <RECEIVE user="12345678" password="password"/>  
</XML>
```

if you want to receive for multiple accounts the XML can be formatted as such

```
<XML>  
  <RECEIVE user="12345678" password="password"></RECEIVE>  
  <RECEIVE user="87654321" password="drowssap"></RECEIVE>  
</XML>
```

3.3.1 Receive Command for Shortcodes

To receive shortcode messages that are stored on gateway send the following command:

```
<XML><RECEIVE user="12345678" password="password"/></XML>
```

The receive command allows you to fetch any inbound message that were sent to your shortcode and are sitting on the gateway.

3.4 Receive Response Elements

Tags	Properties	Optional
RECEIVE_RESPONSE		N
	User	N
	Status	N
SMS_RECEIVE(n)		Y
	Uid	N
	To	N
	Type	N

The response after a receive can be as follows

```
<XML>
  <RECEIVE_RESPONSE user=12345678>
    <SMS_RECEIVE uid="1" to="+275550000001" type="6"/>
    <SMS_RECEIVE uid="2" to="+275550000002" type="1">thanks for the
message</SMS_RECEIVE>
  </RECEIVE_RESPONSE>
</XML>
```

3.4.1 Receive Response for Shortcodes

The receive response shows all SMS's sent to your shortcode number:

```
<?xml version="1.0" encoding="utf-16"?><XML><RECEIVE_RESPONSE
user="12345678"><SMS_RECEIVE uid="73dd640f-cb9c-4517-80f8-1e31b4194368" to="+40171"
from="+27820001111" type="1">Test</SMS_RECEIVE></RECEIVE_RESPONSE></XML>
```

3.5 Parse Errors Element

When an XML response is sent back it may contain a PARSE_ERRORS element, which has an indication of why the XML sent failed to parse.

The hierarchy is as follows

Tags	Properties	Optional
PARSE_ERRORS		N
PARSE_ERROR(n)		N

e.g.

```
<XML>
  <PARSE_ERRORS>
    <PARSE_ERROR>MALFORMED XML</PARSE_ERROR>
  </PARSE_ERRORS>
</XML>
```

or

```
<XML>
  <PARSE_ERRORS>
    <PARSE_ERROR>NO SMSs IN SEND LIST</PARSE_ERROR>
  </PARSE_ERRORS>
</XML>
```

3.6 Compressing sends and receives

Conserving of bandwidth is important, therefore it is possible to compress send and receive data. The compression algorithm is standard ZIP format which is further encoded into XML using Base64 encoding.

The transformation is as follows

XML Data -> ZIP -> Base64 -> XML Package

To send compressed XML

1. Form up your XML as usual

```
<XML>
  <SENDERBATC>
    <SMSLIST>
      <SMS_SEND user="user" password="password" to="+27123000001" uid="1"
application="50" reply="EMAIL:john@smith.com">This is a test message</SMS_SEND>
      <SMS_SEND user="user" password="password" to="+27123000001" uid="2"
application="50" reply="EMAIL:john@smith.com">This is a test message</SMS_SEND>
      <SMS_SEND user="user" password="password" to="+27123000001" uid="3"
application="50" reply="EMAIL:john@smith.com">This is a test message</SMS_SEND>
      <SMS_SEND user="user" password="password" to="+27123000001" uid="4"
application="50" reply="EMAIL:john@smith.com">This is a test message</SMS_SEND>
    </SMSLIST>
  </SENDERBATC>
</XML>
```

2. Compress to ZIP stream
3. Base64 Encode the binary data

```
UESDBBQACAAIAGVrAS8AAAAAAAAAAAAAAAAAAAAEAAAAZGF0YbOJ8PWx4+K0CXb1c3FyDHH
2AHKAPN9g
H8/gEBAzlkHSSuUFqcW2SqBSCWFgsTi4vL8ohRbJRhLSaEk31ZJ28jc0MjYAAwMIRRK4Eqg
HRi
QUFOZnJiSWZ+nq2SpaWSQIFqQU6lrZKrr6Onj1VWfkaeQ3FuZkmGXnJ+rpJdSEZmsQIQJSqUp
BaX
KOSmFhcnpqfa6ANdQiUnGQ0+JxkPPieZUNIJYAY0YQHziCRnow9KhgBQSwcI3Xjd3rIAAACMA
gAA
UESBAhQAFAAIAAgAZWsBL9143d6yAAAAjAIAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAGRhdGFQ
SwUGAAAA
AAEAAQAAyAAAA5AAAAAA
```

It should be noted that different encoders might insert *new lines* and/or *carriage returns*. These should, if received, be stripped before decoding.

4. Insert into XML

```
<XML>
  <ZIP_XML>
UESDBBQACAAIAGVrAS8AAAAAAAAAAAAAAAAAAAAEAAAAZGF0YbOJ8PWx4+K0CXb1c3FyDHH
2AHKAPN9g
H8/gEBAzlkHSSuUFqcW2SqBSCWFgsTi4vL8ohRbJRhLSaEk31ZJ28jc0MjYAAwMIRRK4Eqg
HRi
```

```
QUFOZnJiSWZ+nq2SpaWSQIFqQU6lrZKrr6Onj1VWfkaeQ3FuZkmGXnJ+rpJdSEZmsQIQJSqUp
BaX
KOSmFhcnpqfa6ANdQiUnGQ0+JxkPPieZUNIYAY0YQHziCRnow9KhgBQSwcl3Xjd3rIAAACMA
gAA
UESBAhQAFAAIAAgAZWsBL9143d6yAAAAjAIAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAGRhdGFQ
SwUGAAAA
AAEAAQAYAAAA5AAAAAA
  </ZIP_XML>
</XML>
```

To decode data that is in a ZIP_XML element reverse the process

1. Get text from ZIP_XML element
2. Remove *new lines* and *carriage returns*
3. Decode from Base64
4. Decode from ZIP

To request that the server compress responses add a *compress_response="1"* attribute to your XML element.

It should be noted that the ZIP stream should only have 1 entry. In the case of received XML data the name of this entry is *data*.

The benefits of compressing the data are only really seen on large sends. A send that has 5000 SMSs requires about 400K of response data. If this data is compressed the data is about 50K. Therefore there is little point in compressing sends smaller than 500.

3.7 *Send and Receive*

It is possible to have multiple types of requests in one XML request

As an example you could do this

```
<XML>
  <SEENDBATCH>
    <SMSLIST>
      <SMS_SEND user="user" password="password" to="+271230000001"
uid="1" application="50" reply="EMAIL:john@smith.com">This is a test message</SMS_SEND>
    </SMSLIST>
  </SEENDBATCH>
  <RECEIVE user="12345678" password="password"></RECEIVE>
</XML>
```

and the response might look like this

```
<XML>
  <SEENDBATCHRESPONSE>
    <SMS_SEND_RESPONSE uid="1" to="+271230000001"
send_status="1">authentication failed</SMS_SEND_RESPONSE>
  </SEENDBATCHRESPONSE>
  <RECEIVE_RESPONSE user="12345678">
    <SMS_RECEIVE uid="1" to="+275550000001" type="6"/>
    <SMS_RECEIVE uid="1" to="+275550000001" type="1">thanks for the
message</SMS_RECEIVE>
  </RECEIVE_RESPONSE>
</XML>
```

3.8 HTTP Pushes

By sending an SMS with the following tag

```
reply='HTTP:http://myserver.com/somepage.jsp'
```

you are requesting the server to “push” the responses from the server to your web site. The data the SMS Gateway will be associated with the form data *xml/data*. It will take the form as per a *RECEIVE* request i.e.

```
<XML>
  <RECEIVE_RESPONSE user=12345678>
    <SMS_RECEIVE uid="1" to="+275550000001" type="6"/>
    <SMS_RECEIVE uid="2" to="+275550000002" type="1">thanks for the
message</SMS_RECEIVE>
  </RECEIVE_RESPONSE>
</XML>
```

To minimise time the server will expect a reply back in the form of XML as such

```
<XML>
  <POST_RESPONSE post_status=<status>/>
</XML>
```

where *<status>* is 0 for failure and !0 for true.

eg

```
//success
<XML>
  <POST_RESPONSE post_status="1"/>
</XML>
```

Any bad XML or HTTP errors will be considered a failure as will any failure to connect to the server. The server will retry 5 times before deeming a message as undeliverable.

4 Notes and considerations when sending

4.1 Sockets and timeouts

When sending on a socket the behaviour of the server is as follows.

1. Receive data (buffer)
2. Validate XML
3. Parse XML
4. Persist SMSs into the database.
5. Generate response

The time you may have to wait for after sending, before receiving a response back from the server can be quite long. The server processes between 2500 – 3200 SMSs per minute. Therefore a delay of between 1.5 and 2 minutes can be expected on a send of 5000. As a result we ask that batches be broken up into sizes of less than 5000 at a time. This will increase the availability of the server.

5. Connecting to SMS Gateway

5.1 HTTP

server: <http://gateway.channelmobile.co.za/service/restservice.ashx>

Request method: POST

Form-request data name: xmldata

5.2 Email

email address: xmlsms@gateway.saratoga.co.za

XML can either be sent in the body or as an attachment of called "sms.xml" and the attachment content-type should be "text/xml"

6. Notes on HTTP/S Post Requests

6.1 General Notes

It has been noted that some components do not encode data correctly when performing a POST. The primary reason for this is that the *urlencoded* encoding is not properly implemented; specifically the encoding of the + character. Any 3rd party component that uses “application/x-www-form-urlencoded” as the content-type should check that the encoding is complete.

With URL encoding certain characters become escaped by using the % character.

These characters include

```
'*',  
'#',  
'%',  
'<',  
'>',  
':',  
'[',  
']',  
'+',  
'&'
```

For example + is escaped as %2b.

To check if you suffer from this problem send an XML encoded message via a POST with the *to* set to your number excluding the +country code. e.g. *to*="0821234567" and as the text use "test++++test" if it is encoded correctly you will receive the message correctly otherwise you will receive the message as "test test".

You can look at the URL-encoded values at the following link

<http://www.i-technica.com/whitestuff/urlencodechart.html>

6.2 Indy Components

This problem can be fixed by adding the '+' character to the *UnsafeChars* array in the method *TIdURI.ParamsEncode*.

7. Notes on XML encoding

When sending XML the first line is usually of the form

```
<?xml version="1.0" encoding=encoding_type?>
```

If this line is omitted then it is assumed that the encoding type is UTF-8 which means that extended characters are encoded as such `&#xHHHH` where HHHH is a hex value

eg. `é` (dec:130) becomes `‚`

If you intend on including extended characters and do not want to encode correctly the best option is to use the ISO-8859-1 character set, which will allow you for the most part to include these extended characters

To specify ISO-8859-1 as the encoding like this

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

Please note that the `<?xml?>` tag is DIFFERENT from the `<XML>` tag and that the `<?xml?>` tag should appear before the `<XML>` tag.