SMS AppLink

Application-to-Person SMS Service

XML API

SMS AppLink XML API

Table of Contents

Version History

Version	Date	Changes
1.0	10 July 2008	First version.
1.1	15 July 2008	Schema changed to include WAP Push messages.
1.2	16 July 2008	Update testing procedure. Added descriptions for `reason` and `code` tags. Added WAP Push example XML.
1.3	20 July 2008	Corrected example XML.
1.4	1 September 2008	Removed serviceCode and messageSubmitDateTime Corrected MSISDN formats Added validityPeriod Added XML reserved characters

Abbreviations

API	Application Programming Interface
FQDN	Fully Qualified Domain Name
GMP	Grapevine Messaging Platform
GSM	Global System for Mobile
HTTP	Hyper Text Transfer Protocol
MSISDN	Mobile Station International Subscriber Directory Number
PIN	Personal Identification Number
SMS	Short Message Service
SMSC	Short Message Service Centre
SOAP	Simple Object Access Protocol
URL	Uniform Resource Locator
WAP	Wireless Application Protocol
WSDL	Web Services Description Language
XML	eXtensible Mark-up Language
XSD	XML Schema Definition

INTRODUCTION

SMS AppLink is a Grapevine Interactive product that enables the transmission of SMS (text and WAP Push) messages via the HTTP or the SOAP protocol.

This product is intended for use with applications that have been SMS-enabled. Applications can POST messages to SMS AppLink and receive receipts and replies via a call-back POST.

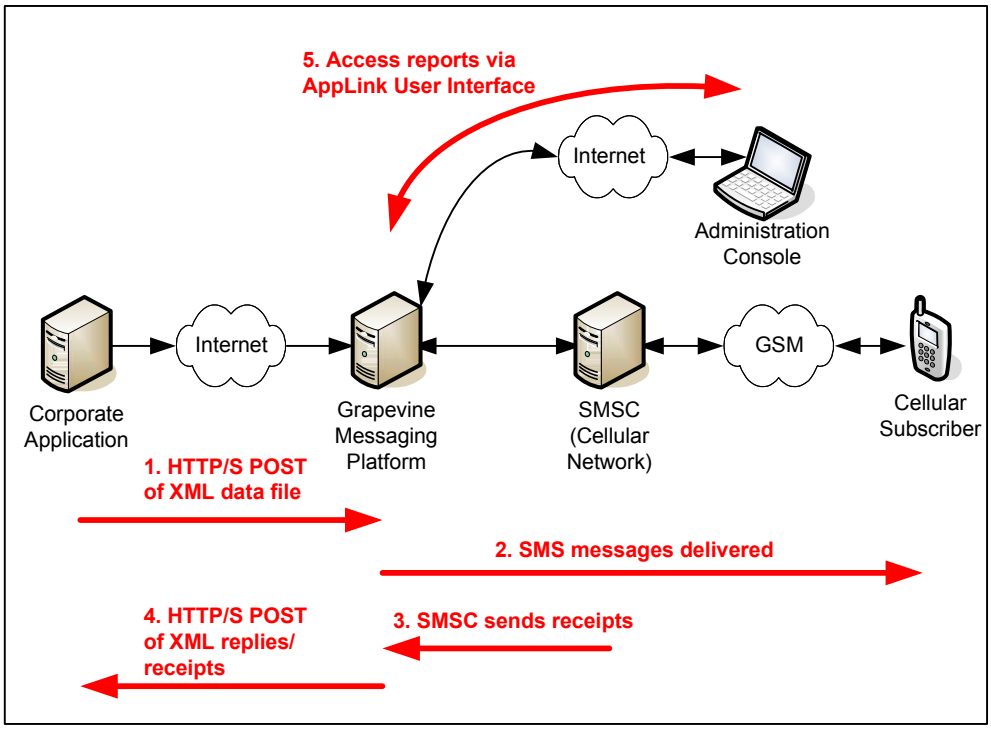

Figure 1: SMS Send using HTTP protocol

INFORMATION NEEDED TO POST SMS DATA

Once you have been provisioned by Grapevine Interactive, you will receive an e-mail message with the following information:

- URL to which to POST requests and the fully qualified domain name to invoke the web service
- Authentication code
- Affiliate code
- Fully Qualified Domain Name for the web service URL

You must include the Authentication Code and Affiliate Code in the XML upload data files.

TESTING THE HTTP CONNECTION

The AppLink Web User Interface allows you to test the HTTP connection by sending test XML data.

In order to access the AppLink user interface, you need the following information:

- AppLink Web User Interface URL
- Your username and password

Contact your system administrator or Grapevine support if you don't know what these are.

You will also need the following information to create the test XML:

- Affiliate Code
- Authentication Code
- MSISDN to which you want to send the test message

You can test the HTTP connection in two ways: edit the XML document in the web user interface or upload your own XML document to the web interface.

If the HTTP connection is active, you should see the following reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<gviRequestResult>
  <resultCode>0</resultCode>
  <resultText>{url encoded name value pairs} Messages accepted for
   delivery</resultText>
</gviRequestResult>
```

When you've completed the test, you can send the actual XML files to Grapevine as explained in the rest of this document.

Testing with the Supplied XML Document

1. When you have logged in to the AppLink web user interface, click '**TRIGGER TESTER**'.
2. The system displays an editable XML document and you replace the content of the affiliateCode, authenticationCode and msisdn node with your details.
3. Click **POST**.

If the HTTP connection is active, you should see the following reply:

```
<?xml version="1.0" encoding="UTF-8" ?>
<gviRequestResult>
  <resultCode>0</resultCode>
  <resultText>{url encoded name value pairs} Messages accepted for
   delivery</resultText>
</gviRequestResult>
```

Testing with Your Own XML Document

1. When you have logged in to the AppLink web user interface, click '**TRIGGER TESTER**'.
2. Create your test file (see Appendix B and C for examples) and click **Browse**.
3. Navigate to your test file and select it.
4. Click **POST**.

SENDING MESSAGES

To send messages, you can use either of two methods:
1. POST an XML data file to the Grapevine platform to the specified URL.
2. Invoke a Grapevine web service to send the XML data to the platform.

The platform extracts the message data from the XML and generates and dispatches SMS messages.

POST Response

The response to the POST or the web service invocation indicates whether AppLink has received the post or not. The table below lists the possible response codes and their meaning.

Code	Meaning
0	Message received OK
1	Authentication failure (invalid authentication code)
2	Input data format invalid (client error)
3	General system failure (Grapevine server failure)

GRAPEVINE SMS MESSAGE XML

The table below gives an overview of all the elements that you can use in an XML document that you submit to AppLink. Below the table is more detailed information about each element. Please also see the appendices for the full XML schema and XML examples.

Name	Element Name	Short Description	Default Value	Content and Occurrence Rules
Root element	gviSmsMessage	Root element of the document. Contains all the other elements.	n.a.	n.a.
Affiliate Code	affiliateCode	A code that, together with the authentication code, identifies your organisation on the Grapevine Messaging Platform.	None	String
Authentication Code	authenticationCode	A code that, together with the affiliate code, identifies your organisation on the Grapevine Messaging Platform.	None	String
Message Type	messageType	Use either “Text” or “Wappush”.	None	“Text” or “Wappush”
Recipient List	recipientList	Envelops a set of tags that contains a message, recipient(s) and custom data.	n.a.	At least one element per document.
Message Text	message	The message text to be sent to the recipients or the URL for WAP Push messages.	None	Standard messages may contain at most 160 characters. If you requested concatenation, AppLink can break up longer messages in up to 5 message parts. The GMP truncates messages that are too long.
Message Description	description	The display text for WAP Push messages.	None	String
Message Recipient	recipient	A container element for the MSISDN, a message per recipient and custom data per recipient.	None	n.a.
MSISDN	msisdn	The number of the mobile device that is the destination for this message.	None	Local format (0831234567 for example) or international format (27831234567 for example)
Custom Data	customData	Use the customData element to add any data that you want to use in your system. For example, you can use it to capture reference numbers for messages.	None	None

Name	Element Name	Short Description	Default Value	Content and Occurrence Rules
Transmission Rules	transmissionRules	Encapsulates the elements that you use to optionally specify rules for the transmission of your messages.	None	None
Transmission Start Date	transmitDateTime	Specifies a date and time in the future when the GMP must start sending out your submitted messages.	None	YYYY-MM-DDThh24:mm:dd
Transmission Time of Day	transmitPeriod	Encapsulate the startHour and endHour tags that define a period during the day when AppLink must transmit the messages in this document.	None	None
Transmission Period Start Hour	startHour	The hour when the system must start sending your messages.	None	Whole numbers from 0 to 23.
Transmission Period End Hour	endHour	The hour when the system must stop sending your messages.	None	Whole numbers from 0 to 23.
Validity Period	validityPeriod	The period that the network must try to deliver a message before marking it as expired.	None	Whole number in hours.

Root Element <gviSmsMessage>

`gviSmsMessage` is the root element and it envelops all the other tags in the XML document.

Authentication <affiliateCode> <authenticationCode>

To make sure that you may use Grapevine's Messaging Platform, you need to submit the following tags in your XML:

- `affiliateCode` – you will have received this with your activation e-mail
- `authenticationCode` – you will have received this with your activation e-mail

Message Type <messageType>

Use either "Text" or "Wappush". Text messages are ordinary SMS messages. WAP Push messages arrive at a mobile device as an SMS and when the user opens it, the phone automatically launches its browser and opens the page specified in the WAP Push message's URL.

Message Description <description>

The display text for WAP Push messages. This is the text that will appear in the body of the message. When the receiver selects to view the message, the mobile device launches its web browser and opens the URL specified in the `<message>` tag.

Recipient List <recipientList>

You must have at least one `recipientList` element per submitted XML document. This element envelops a set of tags that contains a message, recipient(s) and custom data.
If you want a separate message per recipient, you need a recipient list per recipient. Otherwise one message goes to all recipients.

Message Text <message>

This element contains the message to be sent to the recipients. If this element is a child of `recipientList`, the same message goes to all recipients. If you want a separate message for each recipient, place this element as a child of each of the `recipient` elements.

Standard messages may contain at most 160 characters. If you requested concatenation when you were provisioned with the AppLink service, the Grapevine Messaging Platform (GMP) can break up longer messages in up to a maximum of 5 message parts of 140 characters each. The GMP truncates messages that are too long.

Below are the characters that the GMP accepts.

space	!	"	#	$	%	&	'	(	)
*	+	,	-	.	/	0	1	2	3
4	5	6	7	8	9	:	;	<	=
>	?	@	A	B	C	D	E	F	G
H	I	J	K	L	M	N	O	P	Q
R	S	T	U	V	W	X	Y	Z	[
\	]	^	_	`	a	b	c	d	e
f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y
z	{	|	}	~					

XML Reserved Characters

The XML data format requires that you escape XML's reserved characters except when used as markup delimiters, or within a comment, a processing instruction, or a CDATA section.[1] The table below indicates the XML reserved characters and their escaped versions.

Reserved Character	Escaped Version	Example
&	&amp;	<message>Itchy & scratchy show</message>
<	&lt;	<message>for < R100</message>
>	&gt;	<message>save > R100</message>
"	&quot;	<message>"happines" drug </message>
'	&apos;	<message>dog''s tail</message>

[1] http://www.w3.org/TR/REC-xml/#syntax

Delete Message Attribute

If your message content is confidential (such as PINs or passwords) you can set this attribute and so instruct AppLink to delete the message content before logging the message information for reporting purposes. AppLink inserts a comment in the message text stating that it deleted the message content. Below is an example that shows how to use this attribute.

```
<message deleteMessage="true”>Your PIN is 5678.</message>
```

Message Recipient <recipient>

The `recipient` tag contains the `msisdn` element, an optional `message` element for unique messages per recipient and an optional `customData` element.

MSISDN <msisdn>

The MSISDN can be in local format (0831234567 for example) or international format (27831234567 for example). It may not contain any separators, such as spaces, between the digits.
Each recipient has only one MSISDN.

Custom Data <customData>

You can use the customData element to add any data that you want to use in your system. For example, you can use it to capture reference numbers for messages.

The XML may contain `customData` elements on three levels:

- global (child of `gviSmsMessage`)
- as a child of the `recipientList` element
- as a child of the `recipient` element

The three `customData` elements MAY NOT have child elements that have the same names. If elements with the same names exist on all these levels, the lower-level elements will OVERWRITE the higher-level element and you will lose the custom data.

Transmission Rules <transmissionRules>

This element encapsulates the elements that you use to optionally specify rules for the transmission of your messages. If you omit this element, the GMP starts sending out your messages as soon as it receives them.

> Note that the GMP cannot interrupt the sending of messages once they have been processed.

Transmission Start Date <transmitDateTime>

Use this tag to specify a date and time in the future when the GMP must start sending out your submitted messages. Use the format YYYY-MM-DDThh24:mm:dd.

Transmission Time of Day <transmitPeriod>

Use this tag to encapsulate the `startHour` and `endHour` tags that define a period during the day when the GMP must transmit the messages in this document.

If you use this without specifying a `transmitDateTime`, transmission starts when the next `startHour` arrives (today or on the next day). At `endHour`, no new incoming messages are sent out until the next `startHour` arrives on the next day.

If you use this tag together with `transmitDateTime`, the following happens: on the day specified in `transmitDateTime`, the system waits for the start hour and then starts transmitting messages. At `endHour` on the day specified in `transmitDateTime`, the system stops sending out incoming messages until the next `startHour` arrives on the next day.

When the next `startHour` arrives (on the next day), transmission starts again.

`transmitPeriod` takes precedence over `transmitDateTime`, so it overrides `transmitDateTime` in case of any conflict.

Transmission Period Start Hour <startHour>

Use a whole number from 0 to 23 to set the hour when the GMP must start sending your messages. Use with `endHour`.

Transmission Period End Hour <endHour>

Use a number between 0 and 23 to set the hour when the GMP must stop sending your messages. Use with `startHour`.

Validity Period <validityPeriod>

Specify the period that the networks must try to deliver messages before marking them as expired. Use a relative time in hours. The GMP measures the validity period from the time that it submits the messages to the networks' gateways.

If you leave out this value, the networks user their own default values, which is 24 hours.

See Appendices B, C and D for examples of text and WAP Push XML.

WEB SERVICE

If you want to use SOAP rather than HTTP to submit messages to AppLink, you can use the web service.

You will have received the Fully Qualified Domain Name that forms the first part of the SOAP URL in your AppLink activation e-mail. You can find the rest of the URL in the http:address node of the WSDL (see Appendix E on page 23).

The web service has two operations: an uploadRequest and an uploadResponse. The up-LoadResponse have the same range of possible values as the POST response:

Code	Meaning
0	Message received OK
1	Authentication failure (invalid authentication code)
2	Input data format invalid (client error)
3	General system failure (Grapevine server failure)

See Appendix E for the full web service WSDL.

RESPONSES

Responses are receipts, replies and message errors from networks.

The GMP can send responses (receipts, errors and replies) via HTTP POST to a user-specified URL. Receipts and replies are also stored on the platform and may be viewed and reported on via the web-based user interface.

If you want your application to receive receipts and replies directly, you must configure an HTTP listener (such as an ASP page on a Microsoft IIS server or a Java servlet on a web server such as Apache/Tomcat.)

The table below gives an overview of all the elements that you can receive in an AppLink response XML document. Below the table is more detailed information about each element.

See also Appendices E to H for the response XSD and XML examples of receipts, replies and errors.

Name	Element Name	Short Description	Default Value	Content and Occurrence Rules
Root element	gviSmsResponse	Root element of the document. Contains all the other elements.	n.a.	n.a.
Response Date & Time	responseDateTime	The time when the Grapevine Messaging Platform received the receipt or reply from a network.	n.a.	Format YYYY-MM-DDTHH24:MM:SS
Message Submit Date & Time	submitDateTime	In receipts and error messages only (not in replies). It is the time when the Grapevine server submitted the original message to the network.	n.a.	Format YYYY-MM-DDTHH24:MM:SS
Recipient	recipient	Encapsulates the msisdn element.	n.a.	n.a.
Recipient MSISDN	msisdn	The number of the mobile device that received the original message.	None	Local format (0831234567 for example) or international format (27831234567 for example)
Response Type	responseType	Identifies what type of response this XML document contains.	None	Any of “receipt” or “reply” or “error”.
Response Status	status	Reports the success or failure of the message transmission in the two child nodes: code and reason.	None	n.a.
Response Code	code	A code from the network that describes the success or failure of the message.	None	“0” or “-1”.
Response Reason	reason	Describes the meaning of the reason code.	None	String

Root Element <gviSmsResponse>

The `gviSmsResponse` element encapsulates the post back document.

Response Date & Time < responseDateTime>

The `responseDateTime` is the time when the Grapevine Messaging Platform received the receipt or reply from a network. It is in the format YYYY-MM-DDTHH24:MM:SS.

Message Submit Date & Time < submitDateTime>

The `submitDateTime` are in receipts and error messages only (not in replies). It is the time when the Grapevine server submitted the original message to the network. It is in the format YYYY-MM-DDTHH24:MM:SS.

Recipient <recipient>

The `recipient` element has one child: the MSISDN of the message recipient.

Recipient MSISDN

The number of the mobile device that received the original message.
The MSISDN can be in local format (0831234567 for example) or international format (27831234567).

Response Type <responseType>

The `responseType` element identifies what type of response this XML document contains. It can be any of “receipt” or “reply” or “error”.

Response Status (Receipts Only) <status>

Wrapper element for the `code` and `reason` elements, which describe the success or failure of message delivery.

Response Code <code>

The code that describes the success or failure of the message transmission.
The table below lists the status codes:

Code	Meaning
0	Message delivered successfully
-1	Message failed (`reason` tag contains the error information returned by the SMSC)

Response Reason <reason>

The reason for the message failure; supplied by the SMSC.

Custom Data <customData>

If the submitted XML contained `customData` elements, the Messaging Platform adds them to the response XML, starting with the child of `gviSmsMessage`, followed by the child of `recipientList`, followed by the child of `recipient`.

CONTACT US

For more information:

Contact:

Cape Town
T . + 27(0)21 702 3333 | F. +27(0)21 702 3334

Address

Grapevine Interactive SA

Grapevine House

Steenberg Office Park

Silverwood Close

Tokai

7945

For support

Contact:

The Call Centre
Phone (local): (021) 702-3333
Phone (International): +27 21 702-3333
Email: support@vine.co.za

APPENDIX A – GRAPEVINE SMS XML SCHEMA

Note on validation
For historical reasons, the two `message` tags (child of `recipientList` and child of `recipient`) are both required in the schema. However, you shouldn't use both these tags together; use only type of message tag per XML document. If your XML validation fails only because the schema expects the other type of message tag to also be present in the document, you can be sure that AppLink will interpret your XML correctly.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc"
xmlns:jaxb="http://java.sun.com/xml/ns/jaxb" elementFormDefault="qualified"
jaxb:extensionBindingPrefixes="xjc">
   <xsd:element name="gviSmsMessage">
      <xsd:complexType>
         <xsd:complexContent>
            <xsd:extension base="gviSmsMessageType"/>
         </xsd:complexContent>
      </xsd:complexType>
   </xsd:element>
   <xsd:complexType name="customDataType">
      <xsd:sequence>
         <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded">
            <xsd:annotation>
               <xsd:appinfo>
                  <xjc:dom/>
               </xsd:appinfo>
            </xsd:annotation>
         </xsd:any>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="gviSmsMessageType">
      <xsd:sequence>
         <xsd:element name="affiliateCode" type="xsd:string"/>
         <xsd:element name="authenticationCode" type="xsd:string"/>
         <xsd:element name="messageType" type="xsd:string">
            <xsd:annotation>
               <xsd:documentation>Text or Wappush</xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element name="recipientList" type="recipientListType"
          maxOccurs="unbounded"/>
         <xsd:element name="customData" type="customDataType" minOccurs="0"/>
         <xsd:element name="transmissionRules" minOccurs="0">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="transmitDateTime" type="xsd:string"
                   minOccurs="0">
                     <xsd:annotation>
                        <xsd:documentation>Date format: yyyy-MM-ddT hh24:mm:dd
                        </xsd:documentation>
                     </xsd:annotation>
                  </xsd:element>
                  <xsd:element name="transmitPeriod" minOccurs="0">
                     <xsd:complexType>
                        <xsd:sequence>
                           <xsd:element name="startHour">
                              <xsd:annotation>
                                 <xsd:documentation>0 - 23</xsd:documentation>
```

```
                        </xsd:annotation>
                     </xsd:element>
                     <xsd:element name="endHour">
                        <xsd:annotation>
                           <xsd:documentation>0 - 23</xsd:documentation>
                        </xsd:annotation>
                     </xsd:element>
                  </xsd:sequence>
               </xsd:complexType>
            </xsd:element>
            <xsd:element name="validityPeriod" type="xsd:string" minOccurs="0">
               <xsd:annotation>
                  <xsd:documentation>Relative Time in Hours</xsd:documentation>
               </xsd:annotation>
            </xsd:element>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="recipientListType">
   <xsd:sequence>
      <xsd:element name="message">
         <xsd:annotation>
            <xsd:documentation>Used for sms content or wap push URL
            </xsd:documentation>
         </xsd:annotation>
         <xsd:complexType>
            <xsd:simpleContent>
               <xsd:extension base="xsd:string">
                  <xsd:attribute name="deleteMessage" type="xsd:boolean"
                   use="optional"/>
               </xsd:extension>
            </xsd:simpleContent>
         </xsd:complexType>
      </xsd:element>
      <xsd:element name="description" minOccurs="0">
         <xsd:annotation>
            <xsd:documentation>Used for wap push URL description
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
      <xsd:element name="recipient" maxOccurs="unbounded">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="msisdn" type="xsd:string"/>
               <xsd:element name="message">
                  <xsd:annotation>
                     <xsd:documentation>Used for sms content or wap push URL
                     </xsd:documentation>
                  </xsd:annotation>
                  <xsd:complexType>
                     <xsd:simpleContent>
                        <xsd:extension base="xsd:string">
                           <xsd:attribute name="deleteMessage" type="xsd:boolean"
                            use="optional"/>
                        </xsd:extension>
                     </xsd:simpleContent>
                  </xsd:complexType>
               </xsd:element>
               <xsd:element name="description" minOccurs="0">
                  <xsd:annotation>
                     <xsd:documentation>Used for wap push URL description
                     </xsd:documentation>
                  </xsd:annotation>
               </xsd:element>
               <xsd:element name="customData" type="customDataType"
                minOccurs="0"/>
```

```
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="customData" type="customDataType" minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:schema>
```

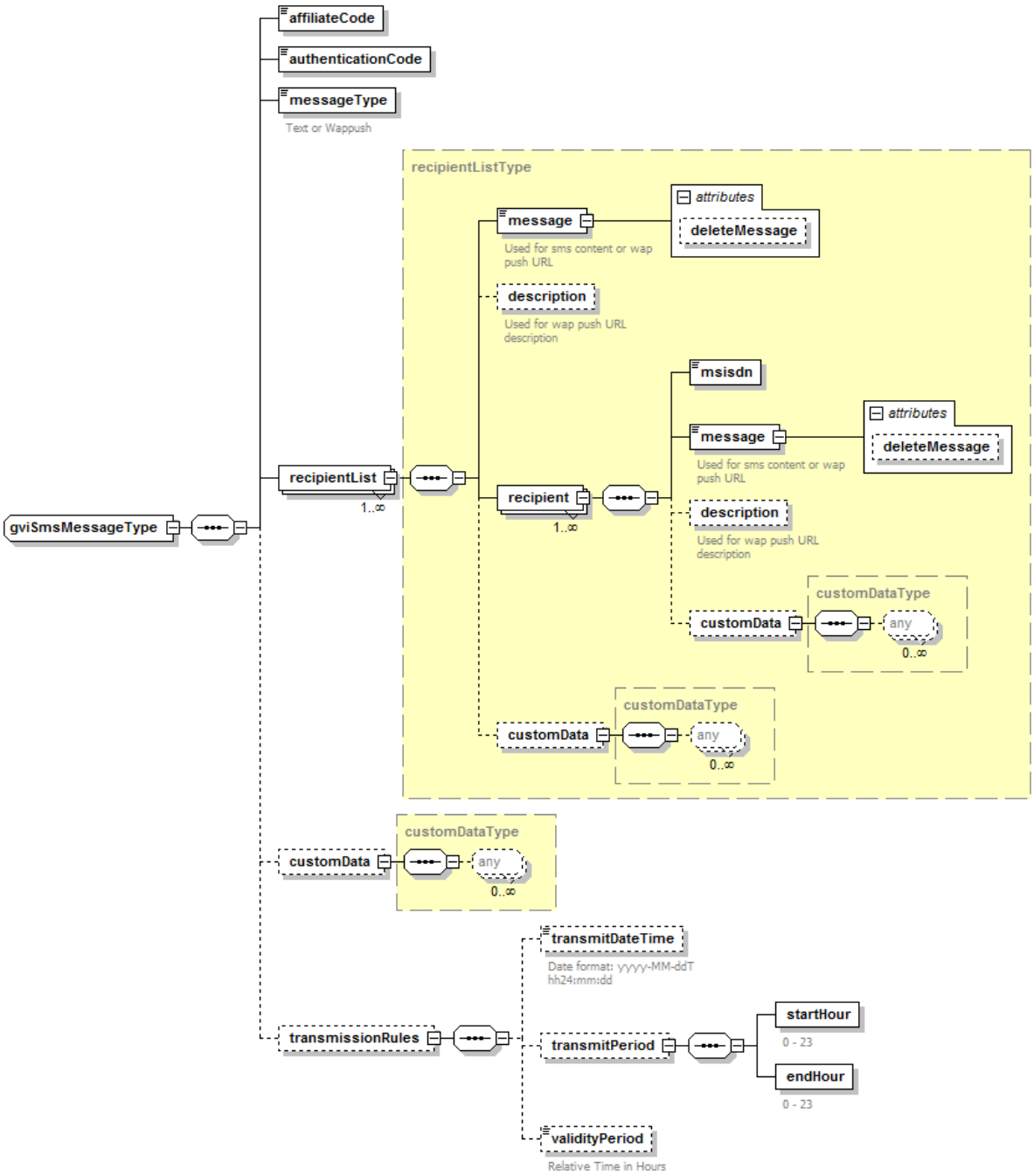

Figure 2: SMS AppLink XML Schema

APPENDIX B – SAMPLE XML (ONE MESSAGE TO MULTIPLE RECIPIENTS)

This example specifies a single message to be sent to a list of recipients. Note that a unique ID can be associated with each individual message; with each recipient list or with the entire batch as a whole using the `customData` element.

Note also that there can be more than one recipient list.

Note that the global `customData` element uses `ref` to distinguish this reference from the `listRef` element (child of `recipientList`) and from the `msisdnRef` element (child of `recipient`).

```
<?xml version="1.0" encoding="UTF-8"?>
<gviSmsMessage xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <!-- affiliate code and authentication codes are supplied by Grapevine -->
   <affiliateCode>NewAffiliate</affiliateCode>
   <authenticationCode>NEW001</authenticationCode>
   <submitDateTime>2007-05-20T11:30:20</submitDateTime>
   <!-- The message type can currently only be set to text. -->
   <messageType>text</messageType>
   <!--List of recipients to receive the message. -->
   <recipientList>
      <message>This is test message 1</message>
      <recipient>
         <msisdn>082123456</msisdn>
         <!-- Optional tag which can have any custom tags for each msisdn. -->
         <customData>
            <msisdnRef>413909</msisdnRef>
         </customData>
      </recipient>
      <recipient>
         <msisdn>0839876543</msisdn>
         <customData>
            <msisdnRef>5004802</msisdnRef>
         </customData>
      </recipient>
   </recipientList>
   <recipientList>
      <message>This is test message 2</message>
      <recipient>
         <msisdn>084123456</msisdn>
      </recipient>
      <!-- Optional tag which can have any custom tags for a recipient list -->
      <customData>
         <listRef>ABCD</listRef>
         <listData>Sales Shot</listData>
      </customData>
   </recipientList>
   <!-- Optional tag which can have any custom tags for a smsMessage send -->
   <customData>
      <ref>500000000061</ref>
      <otherData>any ...</otherData>
   </customData>
   <transmissionRules>
      <transmitDateTime>2007-05-20T11:30:20</transmitDateTime>
      <transmitPeriod>
         <startHour>08</startHour>
         <endHour>17</endHour>
      </transmitPeriod>
   </transmissionRules>
</gviSmsMessage>
```

APPENDIX C – SAMPLE XML (MESSAGE PER RECIPIENT)

This example shows how to send a unique message to each individual in the batch. Note that you can combine both types of message in a single batch i.e. a single message to multiple recipients and single message to single recipient.

Note that the global `customData` element uses `ref` to distinguish this reference from the `listRef` element (child of `recipientList`) and from the `msisdnRef` element (child of `recipient`).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<gviSmsMessage xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <affiliateCode>411</affiliateCode>
  <authenticationCode>411</authenticationCode>
  <submitDateTime>2007-05-20T11:30:20</submitDateTime>
  <messageType>text</messageType>
  <recipientList>
    <message>Test Message 1</message>
    <recipient>
      <msisdn>0823398094</msisdn>
      <customData>
        <msisdnRef>413909</msisdnRef>
      </customData>
    </recipient>
    <customData>
      <listRef>ABCD</listRef>
    </customData>
  </recipientList>
  <recipientList>
    <message>Test Message 2</message>
    <recipient>
      <msisdn>0834451878</msisdn>
      <customData>
        <msisdnRef>5004802</msisdnRef>
      </customData>
    </recipient>
    <customData>
      <listRef>XYZ</listRef>
    </customData>
  </recipientList>
  <customData>
    <ref>500000000061</ref>
  </customData>
  <transmissionRules>
    <transmitDateTime>2007-05-20T11:30:20</transmitDateTime>
    <transmitPeriod>
      <startHour>08</startHour>
      <endHour>17</endHour>
    </transmitPeriod>
  </transmissionRules>
</gviSmsMessage>
```

APPENDIX D – SAMPLE XML (WAP PUSH)

```
<?xml version="1.0" encoding="UTF-8"?>
<gviSmsMessage xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <!-- affiliate code and authentication codes are supplied by Grapevine -
->
  <affiliateCode>NewAffiliate</affiliateCode>
  <authenticationCode>NEW001</authenticationCode>
  <submitDateTime>2007-05-20T11:30:20</submitDateTime>
  <!-- The message type can currently only be set to text. -->
  <messageType>Wappush</messageType>
  <!--List of recipients to receive the message. -->
  <recipientList>
    <message>http://www.example.com/page1</message>
    <description>Visit our website</description>
    <recipient>
      <msisdn>082123456</msisdn>
      <!-- Optional tag which can have any custom tags for each msisdn.-->
      <customData>
        <msisdnRef>413909</msisdnRef>
      </customData>
    </recipient>
    <recipient>
      <msisdn>0839876543</msisdn>
      <customData>
        <msisdnRef>5004802</msisdnRef>
      </customData>
    </recipient>
  </recipientList>
  <recipientList>
    <message>http://www.example.com/page2</message>
    <description>Visit our website</description>
    <recipient>
      <msisdn>084123456</msisdn>
    </recipient>
    <!-- Optional tag which can have any custom tags for a recipient list
     -->
    <customData>
      <listRef>ABCD</listRef>
      <listData>Sales Shot</listData>
    </customData>
  </recipientList>
  <!-- Optional tag which can have any custom tags for a smsMessage send
   -->
  <customData>
    <ref>500000000061</ref>
    <otherData>any ...</otherData>
  </customData>
  <transmissionRules>
    <transmitDateTime>2007-05-20T11:30:20</transmitDateTime>
    <transmitPeriod>
      <startHour>08</startHour>
      <endHour>17</endHour>
    </transmitPeriod>
  </transmissionRules>
</gviSmsMessage>
```

APPENDIX E – WEB SERVICE

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:axis2="http://tempuri.org/" xmlns:ns1="http://org.apache.axis2/xsd"
 xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
 xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
 xmlns:ns0="http://tempuri.org/xsd"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
 targetNamespace="http://tempuri.org/">
  <wsdl:types>
    <xs:schema xmlns:xsd="http://tempuri.org/xsd"
     attributeFormDefault="qualified" elementFormDefault="qualified"
     targetNamespace="http://tempuri.org/xsd">
      <xs:element name="upload">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="0" name="xmlString" nillable="true"
             type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="uploadResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="0" name="return" nillable="true"
             type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:schema>
  </wsdl:types>
  <wsdl:message name="uploadRequest">
    <wsdl:part name="parameters" element="ns0:upload"/>
  </wsdl:message>
  <wsdl:message name="uploadResponse">
    <wsdl:part name="parameters" element="ns0:uploadResponse"/>
  </wsdl:message>
  <wsdl:portType name="ApplinkUploadPortType">
    <wsdl:operation name="upload">
      <wsdl:input message="axis2:uploadRequest" wsaw:Action="urn:upload"/>
      <wsdl:output message="axis2:uploadResponse"
       wsaw:Action="urn:uploadResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="ApplinkUploadSOAP11Binding"
   type="axis2:ApplinkUploadPortType">
    <soap:binding style="document"
     transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="upload">
      <soap:operation soapAction="urn:upload" style="document"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
```

```
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="ApplinkUploadSOAP12Binding"
   type="axis2:ApplinkUploadPortType">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"
     style="document"/>
    <wsdl:operation name="upload">
      <wsdl:input>
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="ApplinkUploadHttpBinding"
   type="axis2:ApplinkUploadPortType">
    <http:binding verb="POST"/>
    <wsdl:operation name="upload">
      <http:operation location="ApplinkUpload/upload"/>
      <wsdl:input>
        <mime:content part="upload" type="text/xml"/>
      </wsdl:input>
      <wsdl:output>
        <mime:content part="upload" type="text/xml"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="ApplinkUpload">
    <wsdl:port name="ApplinkUploadSOAP11port_http"
     binding="axis2:ApplinkUploadSOAP11Binding">
      <soap:address
       location="http://[fqdn]/webservice/services/ApplinkUpload"/>
    </wsdl:port>
    <wsdl:port name="ApplinkUploadSOAP12port_http"
     binding="axis2:ApplinkUploadSOAP12Binding">
      <soap12:address
       location="http://[fqdn]/webservice/services/ApplinkUpload"/>
    </wsdl:port>
    <wsdl:port name="ApplinkUploadHttpport"
     binding="axis2:ApplinkUploadHttpBinding">
      <http:address
       location="http://[fqdn]/webservice/services/ApplinkUpload"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Note
[fqdn] in the location attributes refers to the fully qualified domain name that you received from Grapevine when you first registered for SMS AppLink.

APPENDIX F – RESPONSE XML SCHEMA

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xjc="http://java.sun.com/xml/ns/jaxb/xjc"
xmlns:jaxb="http://java.sun.com/xml/ns/jaxb"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
jaxb:extensionBindingPrefixes="xjc">
  <xsd:element name="gviSmsResponse" type="gviResponseType"/>
  <xsd:complexType name="gviResponseType">
    <xsd:sequence>
      <xsd:element name="responseDateTime" type="xsd:dateTime"/>
      <xsd:element name="submitDateTime" type="xsd:dateTime"
       minOccurs="0"/>
      <xsd:element name="recipient">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="msisdn">
              <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="11"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="responseType">
        <xsd:annotation>
          <xsd:documentation>Receipt, Reply and Error</xsd:documentation>
        </xsd:annotation>
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:maxLength value="10"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
      <xsd:element name="status" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="code">
              <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="3"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="reason" minOccurs="0">
              <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                  <xsd:maxLength value="250"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="response" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Reply content</xsd:documentation>
        </xsd:annotation>
```

```
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:maxLength value="160"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="customData" type="customDataType"
         form="qualified" minOccurs="0"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="customDataType">
      <xsd:sequence>
        <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded">
          <xsd:annotation>
            <xsd:appinfo>
              <xjc:dom/>
            </xsd:appinfo>
          </xsd:annotation>
        </xsd:any>
      </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

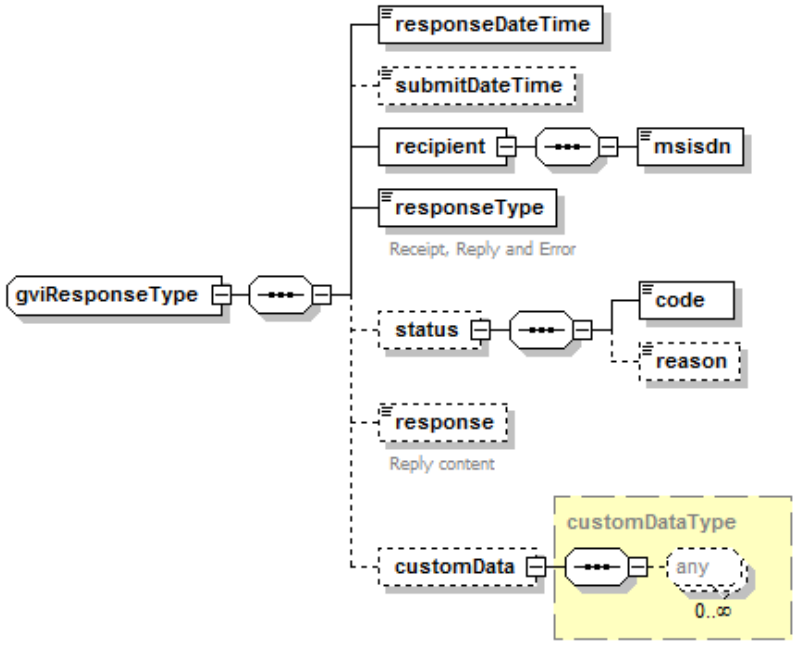

Figure 3: Response XML Schema

APPENDIX G – EXAMPLE RECEIPT XML

The following listing shows example XML generated for a successful delivery receipt.

Note that the receipt has only one `customData` element and it includes the `ref` element, the `listRef` element and the `msisdnRef` element from the posted XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<gviSmsResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <responseDateTime>2007-03-02T13:16:20</responseDateTime>
  <submitDateTime>2007-03-02T13:14:12</submitDateTime>
  <recipient>
    <msisdn>27834451878</msisdn>
  </recipient>
  <responseType>receipt</responseType>
  <status>
    <code>0</code>
    <reason>Message is delivered to destination. stat:DELIVRD</reason>
  </status>
  <customData>
    <ref>500000000061</ref>
    <listRef>XYZ</listRef>
    <msisdnRef>5004802</msisdnRef>
  </customData>
</gviSmsResponse>
```

APPENDIX H – EXAMPLE REPLY XML

The following listing shows example XML generated for a reply.

Note that the reply has only one `customData` element and it includes the `ref` element, the `listRef` element and the `msisdnRef` element from the posted XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<gviSmsResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <responseDateTime>2007-03-02T13:16:20</responseDateTime>
  <recipient>
    <msisdn>27834451878</msisdn>
  </recipient>
  <responseType>reply</responseType>
  <response>Thanks</response>
  <customData>
    <ref>500000000061</ref>
    <listRef>XYZ</listRef>
    <msisdnRef>5004802</msisdnRef>
  </customData>
</gviSmsResponse>
```

APPENDIX I – EXAMPLE ERROR XML

The following listing shows example XML generated for a failed delivery receipt.

Note that the error has only one `customData` element and it includes the `ref` element, the `listRef` element and the `msisdnRef` element from the posted XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<gviSmsResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <responseDateTime>2002-12-19T12:53:29</responseDateTime>
  <submitDateTime>2002-12-19T12:53:48</submitDateTime>
  <recipient>
    <msisdn>27823398094</msisdn>
  </recipient>
  <responseType>error</responseType>
  <status>
    <code>-1</code>
    <reason>SMSC Error: Message is undeliverable. stat:UNDELVR</reason>
  </status>
  <customData>
    <ref>500000000061</ref>
    <listRef>ABCD</listRef>
    <msisdnRef>413909</msisdnRef>
  </customData>
</gviSmsResponse>
```