

ASPSMS.COM

XML INTERFACE

TECHNICAL SPECIFICATIONS

Version 1.6
May 14th 2004

I. Table of contents

I. TABLE OF CONTENTS	II
II. ABBREVIATIONS AND TECHNICAL TERMS	IV
1. INTRODUCTION.....	1
2. OVERVIEW.....	2
2.1 Getting started	2
2.2 General procedure of XML request.....	3
3. COMMUNICATIONS	4
3.1 General	4
3.1 Examples.....	4
3.1.1 PHP example	4
3.1.2 Python example.....	5
3.1.3 ASP example (VBScript)	6
3.1.4 Java.....	7
4. ENCODING.....	8
5. CHARACTER SET	9
6. XML REQUEST	10
6.1 Definition	10
6.1.1 Processing Instruction (PI)	10
6.1.2 Content of the XML Document	10
6.1.3 Special Notice	10
6.2 Overview.....	11
6.3 Further information on individual tags.....	12
6.3.1 <Userkey>	12
6.3.2 <AffiliateId>	12
6.3.3 <Password>	12
6.3.4 <Originator>	12
6.3.4.1 Notice.....	12
6.3.4.2 Unlock procedure.....	12
6.3.4.3 Description.....	13
6.3.5 <UsedCredits>	13
6.3.6 <Recipient>	14
6.3.7 <DeferredDeliveryTime>	14
6.3.8 <MessageData>	14
6.3.9 <URLBinaryFile>	15
6.3.10 <FlashingSMS>	15
6.3.11 <BlinkingSMS>	15
6.3.12 <ReplaceMessage>	16

	II
6.3.13 <MCC>	16
6.3.14 <MNC>	16
6.3.15 <URLBufferedMessageNotification>	16
6.3.16 <URLDeliveryNotification>	17
6.3.17 <URLNonDeliveryNotification>	17
6.3.18 <TimeZone>	17
6.3.19 <XSer>	17
6.3.20 <BinaryFileDataHex>	18
6.3.21 <TransRefNumber>	18
6.3.22 <VCard>	18
6.3.23 <WAPPushDescription>	19
6.3.24 <WAPPushURL>	19
6.3.25 <OriginatorUnlockCode>	19
6.3.26 <Action>	19
6.3.26.1 SendRandomLogo	19
6.3.26.2 SendTextSMS	20
6.3.26.3 SendPictureMessage	20
6.3.26.4 SendLogo	20
6.3.26.5 SendGroupLogo	20
6.3.26.6 SendRingtone	20
6.3.26.7 InquireDeliveryNotifications	20
6.3.26.8 ShowCredits	20
6.3.26.9 SendVCard	20
6.3.26.10 SendBinaryData	20
6.3.26.11 SendWAPPushSMS	20
6.3.26.12 SendOriginatorUnlockCode	20
6.3.26.13 UnlockOriginator	21
6.3.26.14 CheckOriginatorAuthorization	21
7. XML REPLIES	22
7.1 Overview	22
7.2 Further information on individual tags	22
7.2.1 <ErrorCode>	22
7.2.2 <ErrorDescription>	22
7.2.3 <CreditsUsed>	22
7.2.4 <DeliveryNotification>	23
7.2.4.1 <TransRefNumber>	23
7.2.4.2 <DeliveryStatus>	23
7.2.4.3 <SubmissionDate>	23
7.2.4.4 <NotificationDate>	23
7.2.4.5 <ReasonCode>	23
7.2.5 <Credits>	23
7.2.6 <ParserErrorCode>	23
7.2.7 <ParserErrorDescription>	24
7.2.8 <ParserErrorFilePos>	24
7.2.9 <ParserErrorLine>	24
7.2.10 <ParserErrorLinePos>	24
7.2.11 <ParserErrorSrcText>	24
III. EXAMPLES	25
a) Simple text sms	25
b) Text sms with special features	25
c) Customized operator logo	25

	III
d) Random operator logo.....	26
e) Group Logo.....	26
f) Ring tone.....	27
g) Inquire delivery notification	27
h) Show Credits	27
i) VCard.....	28
IV. APPENDIX	29
a) Values of delivery status	29
b) Possible values of reason code.....	30
c) Most occurring reason code	31

II. Abbreviations and technical terms

Element	XML structural construct. An XML element consists of a start tag, an end tag, and the information between the tags, which is often referred to as the contents. Each element has a type, identified by name, sometimes called its “generic identifier” (GI), and may have a set of attribute specifications. Each attribute specification has a name and a value. An instance of an element is declared using <element> tags.
HTML	Hypertext Markup Language
Processing Instruction	Information contained in an XML structure that is intended to be interpreted by a specific application.
XML	Extended Markup Language. A subset of SGML that is optimized for delivery over the Web, XML provides a uniform method for describing and exchanging structured data that is independent of applications or vendors. At the time of this writing, XML 1.0 is a Worldwide Web Consortium Recommendation, which means that it is in the final stage of the approval process.
XML Document	A document object that is well formed, according to the XML recommendation, and that might (or might not) be valid. The XML document has a logical structure (composed of declarations, elements, comments, character references, and processing instructions) and a physical structure (composed of entities, starting with the root, or document entity).
XML Parser	A software module used to read XML documents and provide access to their content and structure. The XML parser generates a hierarchically structured tree, then hands off data to viewers and other applications for processing, and finally returns the results to the browser. A validating XML parser also checks the XML syntax and reports errors.
xmlSMS	XML interface of ASPSMS.COM

1. Introduction

xmlSMS allows you to send sms (short message service) by using a XML interface. The following features are included:

- **simple sms**
- **wap push sms**
- **multiple sms**
- **flashing sms**
- **blinking sms**
- **delivery notification**
- **alphanumeric originator**
- **ringtones**
- **operator logos**
- **vcards**
- **binary sms**
- **multiple concurrent users**
- **balance check**

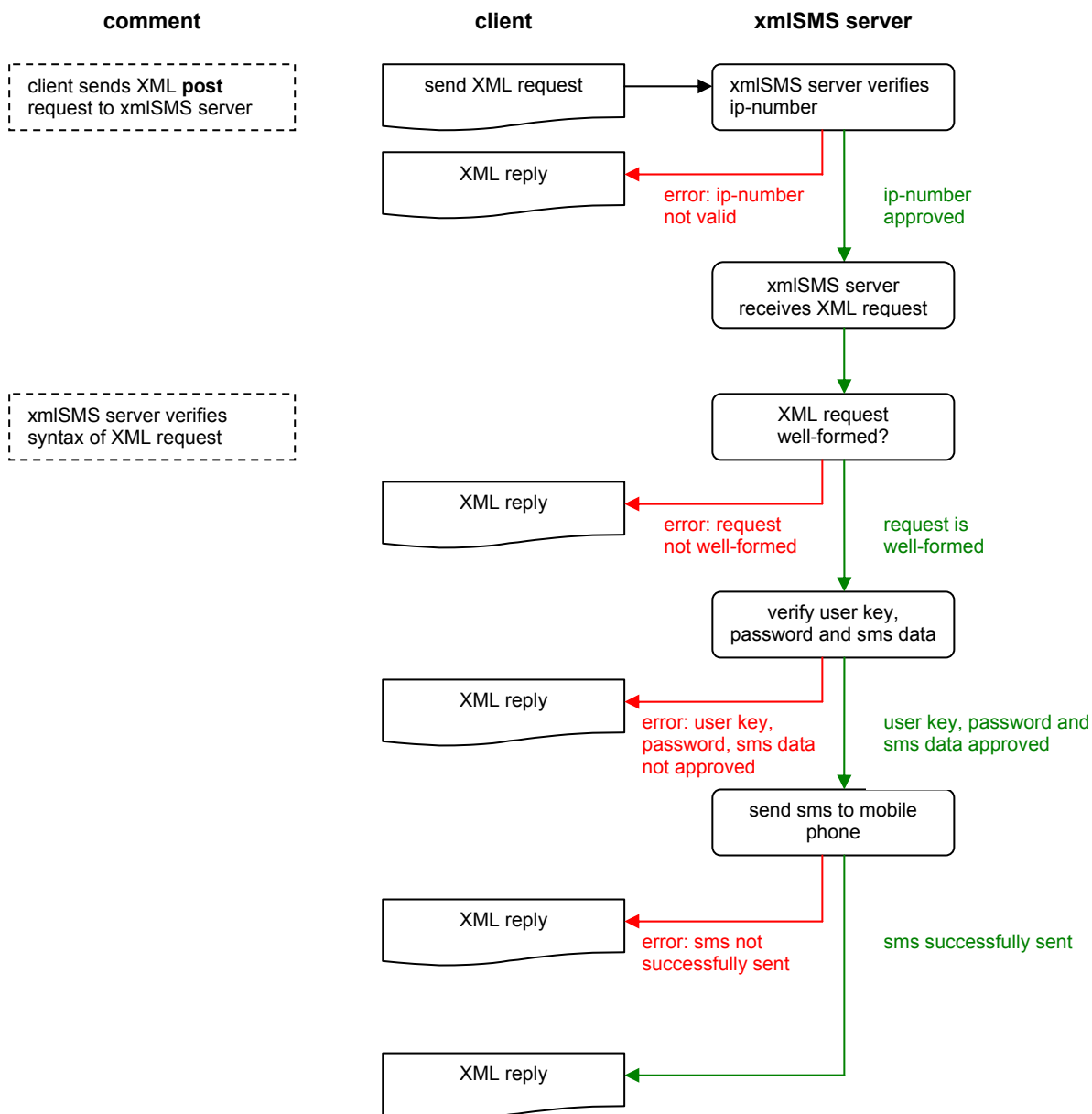
2. Overview

2.1 Getting started

1.	Access to the xmlSMS server	You need a user key and a password. Information are available at http://www.aspsms.com
2.	Register your ip address (optional)	<p>This step is <i>optional</i>.</p> <p>In order to protect your credits you can determine an ip address or a subnet in coherence with your user key. By using this feature, you are only able to use the interface from the determined ip address(es).</p> <p>If no ip address is determined you can access the interface from any address.</p> <p>You can determine your ip address(es) by login on https://www.aspsms.com with you login and password.</p>
3.	Preparation of XML request	Prepare your XML request according to the rules.
4.	Send your XML request	Send your XML request to the following address: http://xml1.aspsms.com:5061/xmlsvr.asp or http://xml1.aspsms.com:5098/xmlsvr.asp or http://xml2.aspsms.com:5061/xmlsvr.asp or http://xml2.aspsms.com:5098/xmlsvr.asp
5	Check for reply messages	xmlSMS server will reply messages to the address where the request was sent from. Reply messages are also sent as XML documents.
6.	Error messages	If you receive reply messages indicating error, correct your XML request and send it again.

2.2 General procedure of XML request

The following scheme describes the general procedure of an XML request you send to the xmlSMS server:



3. Communications

3.1 General

The xmlSMS server consists of two physical servers which are redundant and have different addresses:

- **xmlSMS Server 1: xml1.aspsms.com, port 5061**
- **xmlSMS Server 1: xml1.aspsms.com, port 5098**
- **xmlSMS Server 2: xml2.aspsms.com, port 5061**
- **xmlSMS Server 2: xml2.aspsms.com, port 5098**

In order to successfully make a XML request, the client has to open a socket, transport an XML string, read the reply and then close the socket. The XML string is part of an external message protocol defined by HTTP/1.0.

HTTP/1.0 is described in RFC 1945, „Hypertext Transfer Protocol – HTTP/1.0“. It is not really necessary to read the full specifications since the functionality needed for the xmlSMS service requires only a small subset of the protocol. To be precise, the xmlSMS server processes only **POST requests**.

We do strongly recommend that you program your applications redundant. That means that in case xml1.aspsms.com does not respond your application tries xml2.aspsms.com.

3.1 Examples

Due to many inquiries in the past, we decided to supply some examples how to connect to xmlSvr. We provide examples for PHP, Python and ASP (Active Server Pages) based on VB Script.

3.1.1 PHP example

Please notice that this is not a complete script. It only shows the method of how the connect can be made.

```
<?php
$content = "<?xml version...";
$len = strlen($content);
$fp = fsockopen( "xml1.aspsms.com", 5061 , &$errno, &$errdesc );
    if ( ! $fp ) die ( "Couldn't connect: $errno → $errdesc\n" );
fputs( $fp, "POST /xmlsvr.asp HTTP/1.0\r\n");
fputs( $fp, "Content-Type: text/xml\r\n");
fputs( $fp, "Content-Length: $len\r\n\r\n");
fputs( $fp, $content);
while ( ! feof( $fp ) )
    $reply[] = fgets( $fp, 1024 );
fclose( $fp );
?>
```

3.1.2 Python example

```
#!/usr/bin/python2
print 'Content-Type: text/html'
print # Blank line marking end of HTTP headers

import socket

HOST = 'xml1.aspsms.com' # The remote host
PORT = 5061 # The same port as used by the server
userkey = #please fill in your USERKEY
password = #please enter your PASSWORD
originator = #please fill in the ORIGINATOR
recipient = #please insert the RECIPIENT here
text = #Your SMS Text

CONTENT="""<?xml version="1.0" encoding="ISO-8859-1"?>
<aspsms>
  <Userkey>"""+str(userkey)+"""</Userkey>
  <Password>"""+str(password)+"""</Password>
  <Originator>"""+ str(originator) +""""</Originator>
  <Recipient>
  <PhoneNumber>"""+ str(recipient) +""""</PhoneNumber>
</Recipient>
  <MessageData>"""+ str(text) +""""</MessageData>
  <Action>SendTextSMS</Action>
</aspsms>"""

length=len(CONTENT)
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))
s.send("POST /xmlsvr.asp HTTP/1.0\r\n")
s.send("Content-Type: text/xml\r\n")
s.send("Content-Length: "+str(length)+"\r\n\r\n")
s.send(CONTENT)
datarecv=s.recv(1024)
print "Reply Received: "+ str(datarecv)
s.close()
```

3.1.3 ASP example (VBScript)

```

<%
Option Explicit

'-----
' Define Variables
'-----
Dim xmlExample, XMLhttp, xmlDoc2, Response_Doc

'-----
' HTML
'-----
%>
<html>
<head>
<title>XML Post</title>
</head>
<body>
<pre>
<%
'-----
' Open XML document
'-----
Set xmlExample = CreateObject("MSXML2.DOMDocument")
xmlExample.SetProperty "ServerHTTPRequest", False
xmlExample.Async = False
xmlExample.Load "E:\aspsms\xml\request1.xml"

'-----
' Check XML document
'-----
If Not xmlExample.ParseError = 0 Then
    Response.Write "<b>Error Code:</b> " & xmlExample.ParseError & "<br>"
    Response.Write "<b>Error Description:</b> " & xmlExample.ParseError.reason & "<br>"
    Response.Write "<b>Error File Position:</b> " & xmlExample.ParseError.filepos & "<br>"
    Response.Write "<b>Error Line:</b> " & xmlExample.ParseError.line & "<br>"
    Response.Write "<b>Error Line Position:</b> " & xmlExample.ParseError.linepos & "<br>"
    Response.Write "<b>Error Source Text:</b> " & xmlExample.ParseError.srcText & "<br>"
Else
    '-----
    ' Send XML request
    '-----
    Set XMLhttp = CreateObject("MSXML2.ServerXMLHTTP")
    XMLhttp.Open "POST", "http://xml1.aspsms.com:5061/xmlsvr.asp", False
    XMLhttp.Send xmlExample.xml

    '-----
    ' Get server status
    '-----
    Response.Write "<br>"
    Response.Write "<b>xmlSvr Server Status:</b><br>"
    Response.Write "-----<br>"
    Response.Write "<b>Status (Value must be 200): </b>" & XMLhttp.status & "<br>"
    Response.Write "<b>ReadyState (Value must be 4): </b>" & XMLhttp.ReadyState & "<br>"
    Response.Write "<b>StatusText (Value must be OK): </b>" & XMLhttp.StatusText & "<br>"
    Response.Write "<b>AllResponseHeaders:</b><br>" & XMLhttp.GetAllResponseHeaders & "<br>"

    '-----
    ' Get XML response from xmlSvr
    '-----
    Set xmlDoc2 = CreateObject("MSXML2.DOMDocument")
    xmlDoc2.setProperty "ServerHTTPRequest", True
    xmlDoc2.async = False
    xmlDoc2.LoadXML XMLhttp.ResponseXML.xml
    Response.Write "<br>"
    Response.Write "<b>xmlSvr XML Response:</b><br>"
    Response.Write "-----<br>"
    Response_Doc = xmlhttp.responseXML.xml
    Response_Doc = Replace (Response_Doc, "<", "&lt;")
    Response_Doc = Replace (Response_Doc, ">", "&gt;")
    Response.Write Response_Doc & "<br>"
End If
%>
</pre>
</body>
</html>

```

This script is based on Microsoft Windows 2000 Server, Microsoft Internet Information Server 5.0 (IIS), Microsoft XML 3.0/Service Pack 1 (MSXML) and ASPSMS Active X Component Version 3.0. Make sure that you modify the path for your XML document.

For Microsoft Windows users we have developed a VBScript Class that makes it very easy to use the XML Interface. Further information is available at http://www.aspsms.com/vbscript_class/

3.1.4 Java

```

import java.net.*;
import java.io.*;

public class ASPSMS {

    String xmlURL = "http://xml2.aspsms.com:5061/xmlsvr.asp";
    // insert required userkey,password and originator
    String userkey = "";
    String password = "";
    String originator = "";

    ASPSMS() {
    }

    public String send( int idsms,
                       String number,
                       String message,
                       int flashing,
                       String URLLeliveryNotification,
                       String URLNonDeliveryNotification) {

        String content =
            "<?xml version='1.0' encoding='ISO-8859-1'?'>\r\n"
            + "<aspsms>\r\n"
            + "<Userkey>" + userkey + "</Userkey>\r\n"
            + "<Password>" + password + "</Password>\r\n"
            + "<Originator>" + originator + "</Originator>\r\n"
            + "<Recipient>\r\n"
            + "<PhoneNumber>"
            + number
            + "</PhoneNumber>\r\n"
            + "<TransRefNumber>"
            + idsms
            + "</TransRefNumber>\r\n"
            + "</Recipient>\r\n"
            + "<MessageData>"
            + message
            + "</MessageData>\r\n"
            + "<FlashingSMS>"
            + flashing
            + "</FlashingSMS>\r\n"
            + "<URLDeliveryNotification>"
            + URLLeliveryNotification
            + "<id=</URLDeliveryNotification>\r\n"
            + "<URLNonDeliveryNotification>"
            + URLNonDeliveryNotification
            + "<id=</URLNonDeliveryNotification>\r\n"
            + "<Action>SendTextSMS</Action>\r\n"
            + "</aspsms>\r\n";

        InetAddress inetAddr = null;
        String xmlResult = "";

        try {
            URL aspsmsURL = new URL(xmlURL);
            URLConnection aspsmsCon = aspsmsURL.openConnection();

            aspsmsCon.setRequestProperty("Content-Type", "text/xml");
            aspsmsCon.setDoOutput(true);
            aspsmsCon.setDoInput(true);

            PrintWriter out = new PrintWriter(aspsmsCon.getOutputStream());

            char[] buffer = new char[1024*10];
            buffer = content.toCharArray();

            out.write(buffer,0,content.length());
            out.close();

            BufferedReader in = new BufferedReader(
                new InputStreamReader(
                    aspsmsCon.getInputStream()));

            String inputLine = null;
            while ((inputLine = in.readLine()) != null)
            {
                xmlResult = xmlResult + inputLine;
                System.out.println(inputLine);
            }

            in.close();
        } catch (Exception ex) {
            System.out.println(ex.getMessage());
        }
        return xmlResult;
    }

    public static void main(String[] args) {
        ASPSMS testSMS = new ASPSMS();
        String xmlResult =
            testSMS.send(refnumber,handynumber,message,flashing,urldelivery_yes,urldelivery_no);
    }
}

```

4. Encoding

The XML declaration is a processing instruction (PI) that identifies the document as being XML. All XML documents have to begin with the following XML declaration:

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

If a request does not contain exactly this declaration, it may be refused.

The following encoding rules must be obeyed:

- The one and only encoding declaration that can be used is ISO-8859-1. The xmlSMS server only accepts ISO-8859-1. Others will be refused.
- The value of special characters (38, 60, 62, 128 - 255) must be ISO-encoded, i.e. 'ä' becomes ä. Use of HTML-encoding (e.g. ü for ä) is not allowed.

5. Character Set

Basically we use the ISO-8859-1 char set but you do not have to encode all characters with ISO-8859-1 numbers. The following range within ISO-8859-1 characters (decimal value) have to be encoded by ISO-Numbers:

- 38
- 60
- 62
- 128 till 255

Examples:

& becomes &

< becomes <

> becomes >

Ä becomes Ä

ü becomes ü

Notice that you only have to replace values. Do **not** use < to open a tag (e.g. <aspsms>).

6. XML Request

6.1 Definition

An XML request must contain the following two parts:

- Processing Instruction (PI)
- Content of Document

6.1.1 Processing Instruction (PI)

XML structural construct. A mechanism for embedding information in a file intended for proprietary applications rather than the XML parser or browser. The XML parser passes the instructions to the application.

A processing instruction is a string of text included almost anywhere in an XML document's character data between `<?>` and `?>` marks. It begins with the name of the application for which the PI is intended, followed by the data for the instruction.

The processing instruction always has to contain the following information:

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

6.1.2 Content of the XML Document

The XML document has to start by `<aspsms>` tag and must end by `</aspsms>` tag. Between the so-called root tags you have to insert element tags and child element tags. Let us start with an example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<aspsms>
  <Userkey>I3QHMYKEY6E</Userkey>
  <Password>mypassword</Password>
  <Recipient>
    <PhoneNumber>0041795555555</PhoneNumber>
  </Recipient>
  <MessageData>Test</MessageData>
  <Action>SendTextSMS</Action>
</aspsms>
```

6.1.3 Special Notice

Notice that you send a CRLF after every line. Important: The web server on which xmlSMS server is running classifies a CRLF as 2 characters. To be precise as CR and LF character. Make sure that you modify the content length. Thanks to Simon Vogl for this reference.

6.2 Overview

The following table shows you all tags which can be used:

Root	Element	Child Element	Allowed Chars
<aspsms>	<Userkey>		[a-zA-Z0-9]
	<AffiliateId>		[0-9]
	<Password>		[a-zA-Z0-9]
	<Originator>		[a-zA-Z0-9!\@]
	<UsedCredits>		0 1
	<Recipient>		
		<PhoneNumber>	[0-9\+]
		<TransRefNumber>	[a-zA-Z0-9]
	<DeferredDeliveryTime>		[0-9]
	<MessageData>		ISO-8859-1
	<URLBinaryFile>		ISO-8859-1
	<FlashingSMS>		0 1
	<BlinkingSMS>		0 1
	<MCC>		[0-9]
	<MNC>		[0-9]
	<URLBufferedMessageNotification>		ISO-8859-1
	<URLDeliveryNotification>		ISO-8859-1
	<URLNonDeliveryNotification>		ISO-8859-1
	<TimeZone>		[0-9\+]
	<XSer>		[0-9]
	<BinaryFileDataHex>		ISO-8859-1
	<TransRefNumber>		[a-zA-Z0-9]
	<ReplaceMessage>		1 2 3 4 5 6 7
	<VCard>		
		<VName>	ISO-8859-1
		<VPhoneNumber>	[0-9\+]
	<WAPPushDescription>		ISO-8859-1
	<WAPPushURL>		ISO-8859-1
	<OriginatorUnlockCode>		[0-9]
	<Action>		SendRandomLogo SendTextSMS SendPictureMessage SendLogo SendGroupLogo SendRingtone InquireDeliveryNotifications ShowCredits SendVCard SendBinaryData SendWAPPushSMS SendOriginatorUnlockCode UnlockOriginator CheckOriginatorAuthorization

6.3 Further information on individual tags

6.3.1 <Userkey>

The user key is needed every time your make a request. The xmlSMS server always verifies authentication.

Example:

```
<Userkey>IWHGETG3I</Userkey>
```

6.3.2 <AffiliateId>

If you are registered as an affiliate member you can send the corresponding affiliate id. More information about the affiliate program is available at <http://www.aspsms.com/affiliates/>.

Example:

```
<AffiliateId>2</AffiliateId>
```

6.3.3 <Password>

The password is needed every time your make a request. The xmlSMS server always verifies authentication.

Example:

```
<Password>3424iew3</Password>
```

6.3.4 <Originator>

6.3.4.1 Notice

Please be aware, that using this feature is delicate. ***Any abuse of this nice feature will be tracked and prosecuted according to serious civil- and / or criminal law regulations.*** We will close such accounts immediately without prior notification. Thank you for your understanding.

6.3.4.2 Unlock procedure

Due to potential abuse of the originator feature there are some restrictions. These restrictions only refer to numeric and not to alphanumeric originators. To enable a numeric originator the following steps have to be taken:

1. Request an unlock code for a specific numeric originator by using action
SendOriginatorUnlockCode (see 6.3.26.12). An unlock code will be sent to the mobile phone.

Example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<aspsms>
  <Userkey>LLLLLLLLLLLL</Userkey>
  <Password>password</Password>
  <Originator>0041790000000</Originator>
  <Action>SendOriginatorUnlockCode</Action>
</aspsms>
```

2. Unlock the numeric originator by using action UnlockOriginator (see 6.3.26.13).

Example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<aspsms>
  <Userkey>LLLLLLLLLLLL</Userkey>
  <Password>password</Password>
  <Originator>0041790000000</Originator>
  <OriginatorUnlockCode>65164424</OriginatorUnlockCode>
  <Action>UnlockOriginator</Action>
</aspsms>
```

3. Check if the numeric originator has been unlocked by using action CheckOriginatorAuthorization (see 6.3.26.14)

Example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<aspsms>
  <Userkey>LLLLLLLLLLLL</Userkey>
  <Password>password</Password>
  <Originator>0041790000000</Originator>
  <Action>CheckOriginatorAuthorization</Action>
</aspsms>
```

6.3.4.3 Description

Specifies an alphanumeric or numeric originator. It can be a mobile phone number or a short text with a maximum of 11 characters.

Example:

```
<Originator>aspsms.com</Originator>
```

6.3.5 <UsedCredits>

This feature provides you information about used credits during a transaction. If you set the value equal 0 the credits used will not be returned. By setting the value equal 1 the tag <CreditsUsed> (see 7.2.3) will be returned.

Example:

```
<UsedCredits>1</UsedCredits>
```

6.3.6 <Recipient>

Set a recipient and a transaction reference number (optional). With this feature, it is possible to send messages to a maximum of 1000 recipients at the same time. The transaction reference number is used to inquire delivery notifications. If used, the transaction reference number should be unique to ensure predictable results.

Examples:

```
<!--
Example 1: 1 recipient without transaction number
-----
-->
```

```
<Recipient>
  <PhoneNumber>0041795555555</PhoneNumber>
  <TransRefNumber/>
</Recipient>
```

```
<!--
Example 2: 2 recipients with transaction numbers
-----
-->
```

```
<Recipient>
  <PhoneNumber>0041795555555</PhoneNumber>
  <TransRefNumber>1234</TransRefNumber>
</Recipient>
```

```
<Recipient>
  <PhoneNumber>0041796666666</PhoneNumber>
  <TransRefNumber>1235</TransRefNumber>
</Recipient>
```

6.3.7 <DeferredDeliveryTime>

If set, specifies when a submitted message should be sent. The format is **ddmmyyyyhhmmss**. If this feature is used outside the time zone GMT+1, the time zone should also be set.

Example:

```
<TimeZone>-5</TimeZone >
```

6.3.8 <MessageData>

The message can be text or binary hex-coded.

Examples:

```
<!--
Example 1: Text message
-----
-->
```

```
<MessageData>This is a Textmessage</MessageData>
```



```

<!--
Example 2: Message "I love you!", but only the word "love" is blinking
-----
-->

<BlinkingSMS>1</BlinkingSMS>
<MessageData>I &#60;BLINK&#62; love &#60;/BLINK&#62; you!</MessageData>
<!--
Notice that <BLINK> and </BLINK> have to be ISO-8859-1-encoded.
<BLINK> becomes &#60;BLINK&#62;
</BLINK> becomes &#60;/BLINK&#62;
-->

```

6.3.12 <ReplaceMessage>

If set in a range from 1 to 7, this feature allows you to overwrite existing messages on a handset with the same replace message value. Notice: The message you want to overwrite has already to be indicated by a replace message.

Example:

```
<ReplaceMessage>5</ReplaceMessage>
```

6.3.13 <MCC>

MCC means mobile country code. This feature should be specified sending operator logos. The complete list of operator codes is available at <http://www.gsmworld.com/gsminfo/>

Example:

```
<MCC>228</MCC>
```

6.3.14 <MNC>

MNC means mobile network code. This feature should be specified sending operator logos. The complete list of operator codes is available at <http://www.gsmworld.com/gsminfo/>

Example:

```
<MNC>1</MNC>
```

6.3.15 <URLBufferedMessageNotification>

URL the xmlSMS server is going to request if a message was not delivered instantly and was buffered. The value of the submitted transaction reference number will be the argument of the URL.

Example:

```
<URLBufferedMessageNotification>http://www.yoursite.com/buffered.php4?ID=
</URLBufferedMessageNotification>
```

```
<!--
```

```

The transaction reference number was, for example, set 1234. In this case, the xmlSMS server makes
the following request: http://www.yoursite.com/buffered.php4?ID=1234
-->

```

6.3.16 <URLDeliveryNotification>

URL the xmlSMS server is going to request if a message was delivered instantly. The value of the submitted transaction reference number will be the argument of the URL.

Example:

```
<URLDeliveryNotification>http://www.yoursite.com/delivered.php4?ID=</URLDeliveryNotificatio>
```

```
<!--
```

```
The transaction reference number was, for example, set 1234. In this case, the xmlSMS server makes
the following request: http://www.yoursite.com/delivered.php4?ID=1234
```

```
-->
```

6.3.17 <URLNonDeliveryNotification>

URL the xmlSMS server is going to request if a message was not delivered. The value of the submitted transaction reference number will be the argument of the URL.

Example:

```
<URLNonDeliveryNotification>http://www.yoursite.com/fail.php4?ID=</URLNonDeliveryNotification>
```

```
<!--
```

```
The transaction reference number was, for example, set 1234. In this case, the xmlSMS server makes
the following request: http://www.yoursite.com/fail.php4?ID=1234
```

```
-->
```

6.3.18 <TimeZone>

Specifies the time zone where the xmlSMS service is used. It is only necessary to set this feature if the feature deferred delivery time is used and the service is used outside the time zone GMT +1.

Example:

```
<!--
```

```
Example 1: User is in the time zone GMT+3
```

```
-----
```

```
-->
```

```
<TimeZone>2</TimeZone>
```

```
<!--
```

```
Example 2: User is in the time zone GMT-4
```

```
-----
```

```
-->
```

```
<TimeZone>-5</TimeZone>
```

6.3.19 <XSer>

Additional information for binary messages.

Example:

```
<XSer>010706050415821582</XSer>
```


Example:

```
<!--
Example: A phone book record with name "Tarzan" and phone number "0041796666666" will
        be sent to a mobile phone.
-----
-->
```

```
<VCard>
  <VName>Tarzan</VName>
  <VPhoneNumber>0041796666666</VPhoneNumber>
</VCard>
```

6.3.23 <WAPPushDescription>

The wap push description describes the content of a wap sms.

Example:

```
<WAPPushDescription>Nice Gils at the beach</WAPPushDescription>
```

6.3.24 <WAPPushURL>

Specifies the url of the file to be transferred by wap protocol. This tag has to be set if using the SendWAPPushSMS action.

Example:

```
<WAPPushURL>http://www.yourserver.com/img/beachgirl.gif</WAPPushURL>
```

6.3.25 <OriginatorUnlockCode>

Specifies the unlock code sent to a specific handy to unlock a numeric originator. For more information about the unlock procedure see 6.3.4.

Example:

```
<OriginatorUnlockCode>65164424</OriginatorUnlockCode>
```

6.3.26 <Action>

The action element instructs the xmlSMS server what to do. Notice that only one action can be called within one request. Action values are case sensitive.

Further information about individual values:

6.3.26.1 SendRandomLogo

Sends a random operator logo out of a collection of several hundred logos. This action costs one credit per recipient.

6.3.26.2 SendTextSMS

Sends a text sms. This action costs one credit per recipient.

6.3.26.3 SendPictureMessage

Sends a picture message. This action costs three credits per recipient.

6.3.26.4 SendLogo

Sends an operator logo. This action costs one credit per recipient.

6.3.26.5 SendGroupLogo

Sends a group logo. This action costs one credit per recipient.

6.3.26.6 SendRingtone

Sends a ring tone. This action costs one credit per recipient.

6.3.26.7 InquireDeliveryNotifications

Inquires the status of sent messages and returns a string, which contains the query result. This action costs 0.25 credits per transaction reference number.

6.3.26.8 ShowCredits

Number of messages, that can be sent with the current account balance.

6.3.26.9 SendVCard

Sends a vcard.

6.3.26.10 SendBinaryData

Sends binary-data. XSer has to be specified.

6.3.26.11 SendWAPPushSMS

Sends a wap push sms. Notice, that the following two tags have to be specified:

WAPPushDescription (see 6.3.23) and WAPPushURL (see 6.3.24).

6.3.26.12 SendOriginatorUnlockCode

Initiates the sending of the unlock code to a mobile phone. This action has to be used in combination with <Userkey>, <Password> and <Originator>. For more information about the unlock procedure see 6.3.4.2.

6.3.26.13 UnlockOriginator

Sends the unlock code received by mobile phone back to our servers to verify the numeric originator. This action has to be combined with <Userkey>, <Password>, <Originator> and <OriginatorUnlockCode>. For more information about the unlock procedure see 6.3.4.2.

6.3.26.14 CheckOriginatorAuthorization

Verifies if a specific numeric originator is enabled on your account. This action has to be combined with <Userkey>, <Password> and <Originator>. For more information about the unlock procedure see 6.3.4.2.

7. XML replies

7.1 Overview

Every request you send will be answered by xmlSMS server returning a XML document. The following table shows you all tags which can occur:

Root	Element	Child Element	Allowed Chars
<aspsms>	<ErrorCode>		[0-9]
	<ErrorDescription>		[a-zA-Z0-9]
	<CreditsUsed>		[0-9\.]
	<DeliveryNotification>		
		<TransRefNumber>	ISO-8859-1
		<DeliveryStatus>	[0-9]
		<SubmissionDate>	[0-9]
		<NotificationDate>	[0-9]
		<ReasonCode>	[0-9]
	<Credits>		[0-9]
	<ParserErrorCode>		[0-9\.]
	<ParserErrorDescription>		[a-zA-Z0-9\.\,]
	<ParserErrorFilePos>		[0-9]
	<ParserErrorLine>		[0-9]
	<ParserErrorLinePos>		[0-9]
	<ParserErrorSrcText>		ISO-8859-1

7.2 Further information on individual tags

7.2.1 <ErrorCode>

After invoking an action an error code is going to be returned. If the operation was successful, the error code is 1. If there was an error, the value is different from 1.

Example:

```
<ErrorCode>1</ErrorCode>
```

7.2.2 <ErrorDescription>

The error description gives specific information, why an error has occurred.

Example:

```
<ErrorDescription>Ok</ErrorDescription>
```

7.2.3 <CreditsUsed>

This tag will be returned if your request contains the tag <UsedCredits/>.

Example:

```
<CreditsUsed>2</CreditsUsed>
```

7.2.4 <DeliveryNotification>

Inquires the status of sent messages and returns an XML document, which contains the query result.

Example:

```
<DeliveryNotification>
  <TransRefNumber>1234</TransRefNumber>
  <DeliveryStatus>0</DeliveryStatus>
  <SubmissionDate>13062001182055</SubmissionDate>
  <NotificationDate>13062001182058</NotificationDate>
  <ReasonCode>0</ReasonCode>
</DeliveryNotification>
```

7.2.4.1 <TransRefNumber>

The represented value is the transaction reference number you requested.

7.2.4.2 <DeliveryStatus>

The delivery status represents the current status of the sent sms. If the delivery status is equals to 0 the sms was successfully delivered to the mobile phone. There are other possibilities for delivery status. See appendix for further information.

7.2.4.3 <SubmissionDate>

The value of the submission date is in the format ddmmyyyyhhmmss.

7.2.4.4 <NotificationDate>

The value of the notification date is in the format ddmmyyyyhhmmss.

7.2.4.5 <ReasonCode>

The reason code tells you more about a specific sms. See appendix for further information.

7.2.5 <Credits>

This feature gives you information about your accounts balance.

Example:

```
<Credits>526</Credits>
```

7.2.6 <ParserErrorCode>

The parser error code does only occur if there is a problem with the parsing process caused for example by a bad structured XML document. In this case a numeric value will be returned.

Example:

```
<aspsms>
  <ParserErrorCode>-1072896680</ParserErrorCode>
</aspsms>
```

7.2.7 <ParserErrorDescription>

The parser error description does only occur if there is a problem with the parsing process caused for example by a bad structured XML document. In this case an alphanumeric value will be returned. This tells you the reason why the process failed.

Example:

```
<aspsms>
  <ParserErrorDescription>XML document must have a top level element.</ParserErrorDescription>
</aspsms>
```

7.2.8 <ParserErrorFilePos>

Contains the absolute file position where the error occurred.

Example:

```
<aspsms>
  <ParserErrorFilePos>261</ParserErrorFilePos>
</aspsms>
```

7.2.9 <ParserErrorLine>

Specifies the line number that contains the error.

Example:

```
<aspsms>
  <ParserErrorLine>9</ParserErrorLine>
</aspsms>
```

7.2.10 <ParserErrorLinePos>

Contains the character position within the line where the error occurred.

Example:

```
<aspsms>
  <ParserErrorLinePos>18</ParserErrorLinePos>
</aspsms>
```

7.2.11 <ParserErrorSrcText>

Returns the text of the line containing the error.

Example:

```
<aspsms>
  <ParserErrorSrcText>&</ParserErrorSrcText>
</aspsms>
```

III. Examples

a) Simple text sms

In this first example a simple text sms without any special features will be sent.

```
<!--
XML Request
-----
-->

<?xml version="1.0" encoding="ISO-8859-1"?>
<aspsms>
  <Userkey>I3QHMYKEY6E</Userkey>
  <Password>mypassword</Password>
  <Recipient>
    <PhoneNumber>0041795555555</PhoneNumber>
  </Recipient>
  <MessageData>Hello Jane, i got the tickets. See you. Tarzan</MessageData>
  <Action>SendTextSMS</Action>
</aspsms>
```

b) Text sms with special features

In this example it is getting much more complicated. An originator is set. The user who receives this sms sees “aspsms.com” as originator. A transaction reference number is set to enable to verify the delivery status of the sent sms at a later time. This sms will not be sent immediately because a deferred delivery time is set. In this example the sms will be sent on June, 15th 2001 at 8 pm. In the existing example the sms will be sent as flashing sms with a blinking component. Only the word “love” is set to be blinking. The delivery status is going to be posted to the pre defined URL. The last feature of this request is the time zone setting. If the time zone is not set the xmlSMS server assumes GMT+1. This feature is useful for users they are not in the GMT+1 time zone.

```
<!--
XML Request
-----
-->

<?xml version="1.0" encoding="ISO-8859-1"?>
<aspsms>
  <Userkey>I3QHMYKEY6E</Userkey>
  <Password>mypassword</Password>
  <Originator>aspsms.com</Originator>
  <Recipient>
    <PhoneNumber>0041795555555</PhoneNumber>
    <TransRefNumber>1234</TransRefNumber>
  </Recipient>
  <DeferredDeliveryTime>15062001200000</DeferredDeliveryTime>
  <MessageData>I &#60;BLINK&#62; love &#60;BLINK&#62; you!</MessageData>
  <FlashingSMS>1</FlashingSMS>
  <BlinkingSMS>1</BlinkingSMS>
  <URLBufferedMessageNotification>http://www.mysite.com/sms/buffered.asp?id=
</URLBufferedMessageNotification>
  <URLDeliveryNotification>http://www.mysite.com/sms/delivered.asp?id=
</URLDeliveryNotification>
  <URLNonDeliveryNotification>http://www.mysite.com/sms/nondelivered.asp?id=
</URLNonDeliveryNotification>
  <TimeZone>-5</TimeZone>
  <Action>SendTextSMS</Action>
</aspsms>
```

c) Customized operator logo

This example shows you how to send an customized operator logo to a mobile phone. You have to tell the xmlSMS server where to get the picture data from by setting an URL.


```

<!--
XML Request
-----
-->

<?xml version="1.0" encoding="ISO-8859-1"?>
<aspsms>
  <Userkey>I3QHMYKEY6E</Userkey>
  <Password>mypassword</Password>
  <Originator>aspsms.com</Originator>
  <Recipient>
    <PhoneNumber>0041795555555</PhoneNumber>
  </Recipient>
  <URLBinaryFile>http://www.mysite.com/sms/handylogos/logo1.bmp</URLBinaryFile>
  <Action>SendGroupLogo</Action>
</aspsms>

```

f) Ring tone

A very interesting feature is the following one. It allows you to send ring tones to a mobile phone. The xmlSMS server will get the *.ott or *.txt file directly from the pre defined URL.

```

<!--
XML Request
-----
-->

<?xml version="1.0" encoding="ISO-8859-1"?>
<aspsms>
  <Userkey>I3QHMYKEY6E</Userkey>
  <Password>mypassword</Password>
  <Originator>aspsms.com</Originator>
  <Recipient>
    <PhoneNumber>0041795555555</PhoneNumber>
  </Recipient>
  <URLBinaryFile>http://www.mysite.com/sms/handylogos/tonel.ott</URLBinaryFile>
  <Action>SendRingtone</Action>
</aspsms>

```

g) Inquire delivery notification

Under III b) we have set a transaction reference number. After having sent this sms we want to verify the delivery status.

```

<!--
XML Request
-----
-->

<?xml version="1.0" encoding="ISO-8859-1"?>
<aspsms>
  <Userkey>I3QHMYKEY6E</Userkey>
  <Password>mypassword</Password>
  <TransRefNumber>1234</TransRefNumber>
  <TransRefNumber>1235</TransRefNumber>
  <Action>InquireDeliveryNotifications</Action>
</aspsms>

```

h) Show Credits

This example allows you to verify your current balance.

```

<!--
XML Request
-----
-->

<?xml version="1.0" encoding="ISO-8859-1"?>
<aspsms>
  <Userkey>I3QHMYKEY6E</Userkey>
  <Password>mypassword</Password>
  <Action>ShowCredits</Action>
</aspsms>

```


i) VCard

A vcard is a kind of a phone book record for mobile phones. The following example shows you how to send a vcard to mobile phone.

```
<!--
XML Request
-----
-->

<?xml version="1.0" encoding="ISO-8859-1"?>
<aspsms>
  <Userkey>I3QHMYKEY6E</Userkey>
  <Password>mypassword</Password>
  <Originator>aspsms.com</Originator>
  <Recipient>
    <PhoneNumber>0041795555555</PhoneNumber>
  </Recipient>
  <VCard>
    <VName>Tarzan</VName>
    <VPhoneNumber>0041796666666</VPhoneNumber>
  </VCard>
  <Action>SendVCard</Action>
</aspsms>
```

IV. Appendix

a) Values of delivery status

Value	Reason
-1	Not yet submitted or rejected
0	Delivered
1	Buffered
2	Not Delivered

b) Possible values of reason code

0	Unknown subscriber	112	Unknown SC
1	Service temporary not available	113	SC congestion
2	Service temporary not available	114	Illegal MS
3	Service temporary not available	115	MS not a subscriber
4	Service temporary not available	116	Error in MS
5	Service temporary not available	117	SMS lower layer not provisioned
6	Service temporary not available	118	System fail
7	Service temporary not available	119	PLMN system failure
8	Service temporary not available	120	HLR system failure
9	Illegal error code	121	VLR system failure
10	Network time-out	122	Previous VLR system failure
30	Originator not Authorized	123	Controlling MSC system failure
31	Originator already Authorized	124	VMSC system failure
100	Facility not supported	125	EIR system failure
101	Unknown subscriber	126	System failure
102	Facility not provided	127	Unexpected data value
103	Call barred	200	Error in address service centre
104	Operation barred	201	Invalid absolute Validity Period
105	SC congestion	202	Short message exceeds maximum
106	Facility not supported	203	Unable to Unpack GSM message
107	Absent subscriber	204	Unable to convert to IA5 ALPHABET
108	Delivery fail	205	Invalid validity period format
109	SC congestion	206	Invalid destination address
110	Protocol error	207	Duplicate message submit
111	MS not equipped	208	Invalid message type indicator

c) Most occurring reason code

Reason Code	Reason	Possible explanation
0	Unknown Subscriber	Not existing recipient number used
103	Call Barred	Blocked mobilephone, occurs in cases of stolen phones or unpaid bills
107	Absent Subscriber	Mobilephone switched off or out of memory
108	Delivery Fail	External partner has definitely given up to deliver message after a maximum period of 72 hours
110	Protocol Error	Mobilephone damaged or not capable to receive message, e.g. if a logo has been sent
111	MS not equipped	Mobilephone damaged or not capable to receive message, e.g. if a logo has been sent
118	System fail	Indicates serious problems within mobilephone and/or network of the recipient
119	PLMN Failure	Public Land Mobile Network Failure - Indicates serious problems within network of the recipient
120	HLR system Failure	Home Location Register Failure - Indicates serious problems within network of the recipient, also possible that terminating network has blocked our external partners
121	VLR system Failure	Visiting Location Register Failure - Indicates problems within network of the recipient, occurs often if recipient is moving very fast, e.g. in a driving car