

# Especificaciones de la Interfaz HTTP

Altiria TIC, S.L.L.

Versión: 1.13

Copyright © Altiria TIC 2009

Este documento sólo puede ser reproducido por completo o en parte, almacenado, recuperado o transmitido por medios electrónicos, mecánicos, fotocopiado o cualquier otro medio con el consentimiento previo de los autores de acuerdo con los términos que estos indiquen.

## Historial de cambios

Versión	Cambios
1.13	Se añade una referencia a la sección de preguntas frecuentes de nuestro portal web (sección 1).
1.12	Se incluye un nuevo ejemplo en ASP 3.0 y se corrige una errata en el ejemplo en PHP (sección 2.13). Se detalla el modo de codificar los caracteres si los parámetros de la petición se envían en la URL (sección 2.2). Se precisa la información relativa a la especificación de una URL en mensajes WAP-PUSH (sección 2.8). Se añaden dos parámetros opcionales al comando “sendsms” para permitir especificar el puerto origen y destino del SMS a enviar (sección 2.5.1). Se añaden nuevos códigos de error correspondientes a los nuevos parámetros: 033 y 034 (sección 2.12).
1.11	Se incluye el enlace a la página que detalla la lista de países permitidos y las restricciones geográficas aplicables (sección 1). Se añaden las vocales con acento grave (à) a la lista de caracteres permitidos para los mensajes SMS de texto (sección 2.6). Se modifica la política ante mensajes con caracteres inválidos: a partir de ahora son reemplazados por una “?” y el mensaje es enviado, en lugar de generar el antiguo código de error 012 (secciones 2.6) y 2.7). Se actualiza la información relativa a la consulta de crédito disponible, previa el envío de mensajes (sección 2.4). Se incluye un capítulo con enlaces a las tarifas y a la información de cobertura internacional (sección 3).
1.10	Se actualiza la lista de caracteres permitidos para mensajes de texto, desaconsejando el de apertura de exclamación (cuadro 2.4). Se detalla la respuesta a la petición HTTP POST con datos de confirmación de entrega (sección 2.11).
1.9	Añadido un nuevo ejemplo de programación (sección 2.13). Modificados los posibles estados a recibir como información de confirmación de entrega (cuadro 2.8).
1.8	Se añade la sección 2.11 relativa a la nueva funcionalidad de la confirmación de entrega. Se reestructura la documentación del comando “sendwappush” (sección 2.5.2) y se añade la sección 2.10, aclaratoria a ese respecto. Se añaden nuevos parámetros a los comandos “sendsms” (sección 2.5.1) y “sendwappush” (sección 2.5.2) para gestionar la confirmación de entrega. Se añade limitación de longitud y de caracteres permitidos en el parámetro senderId del comando “sendsms” (sección 2.5.1). Añadido un nuevo código de error: 032 (sección 2.12).

**Historial de cambios**

<b>Versión</b>	<b>Cambios</b>
1.7	Añadidos nuevos ejemplos de programación (sección 2.13).
1.6	Nuevo comando para el envío de mensajes multimedia WAP-PUSH (sección 2.5.2). Nuevo apartado sobre los mensajes WAP-PUSH (sección 2.9). Añadidos nuevos código de error: 030, 031 y eliminado el 021 (sección 2.12). Apartado sobre las posibilidades de comprobar el credito disponible (sección 2.4). El comando “getcredit” devuelve un valor numérico con dos decimales (sección 2.5.3). Apartado sobre las reglas de composicion de una URL para los mensajes multimedia WAP-PUSH (sección 2.8).
1.5	Anotada la limitación en la longitud de la URL para el envío de los parámetros como parte de la misma (sección 2.1). Añadida la limitación de no poder enviar un mensaje vacío (sección 2.5.1). Añadido un nuevo código de error: 017 (sección 2.12). Añadidos nuevos ejemplos de programación (sección 2.13).
1.4	Agregada la opción de seleccionar remitente en el comando sendsms. Comprobación del crédito disponible antes de aceptar el comando sendsms. Nuevo comando getcredit. Añadidos nuevos códigos de error: 015, 016, 021 y 022. Reestructuración del documento.

# Índice general

<b>1. Introducción</b>	<b>4</b>
<b>2. Descripción de la API</b>	<b>5</b>
2.1. Envío de la petición . . . . .	5
2.2. Codificación de caracteres . . . . .	5
2.3. Respuesta a la petición . . . . .	6
2.4. Crédito disponible . . . . .	6
2.5. Comandos de la API . . . . .	7
2.5.1. Envío de un mensaje de texto . . . . .	7
2.5.2. Envío de un mensaje multimedia WAP-PUSH . . . . .	9
2.5.3. Consulta del crédito disponible . . . . .	11
2.6. Caracteres permitidos para mensajes de texto . . . . .	12
2.7. Caracteres permitidos para mensajes WAP-PUSH . . . . .	12
2.8. Especificación de una URL . . . . .	13
2.9. Mensajes multimedia WAP-PUSH . . . . .	14
2.9.1. Tipos de contenido . . . . .	14
2.9.2. Formato del contenido . . . . .	14
2.9.3. Dirección del contenido . . . . .	15
2.9.4. Envío del contenido . . . . .	15
2.10. Envío de mensajes WAP-PUSH . . . . .	15
2.11. Confirmación de entrega . . . . .	16
2.12. Códigos de error . . . . .	18
2.13. Ejemplos . . . . .	19
2.13.1. Envío de un mensaje en PHP . . . . .	19
2.13.2. Envío de un mensaje en JAVA . . . . .	20
2.13.3. Envío de un mensaje en Visual Basic . . . . .	21
2.13.4. Envío de un mensaje en .NET . . . . .	22
2.13.5. Envío de un mensaje en Borland C++ Builder . . . . .	24
2.13.6. Envío de un mensaje en Borland Delphi . . . . .	25
<b>3. Tarifas y cobertura internacional</b>	<b>27</b>

# Capítulo 1

## Introducción

En este documento se presenta la API disponible para el envío de mensajes cortos sobre la interfaz de *Altiria* a través de peticiones sobre el protocolo HTTP.

Para hacer uso de la interfaz HTTP el cliente enviará una petición HTTP POST y esperará la respuesta del servidor.

El servicio de envío de mensajes cortos está disponible en muchos países. Consultar el capítulo 3 para conocer los países permitidos, las operadoras válidas en cada país y las posibles **restricciones geográficas**, salvedades al funcionamiento general detallado en este documento, que pudieran aplicar en cada caso.

El servicio opcional de confirmación de entrega requiere que el cliente configure un servidor de peticiones HTTP para recibir la información de confirmación.

Si durante la integración de la API se presentan **problemas**, se recomienda revisar la sección de **preguntas frecuentes** ([FAQ]) de nuestro portal web.

## Capítulo 2

# Descripción de la API

### 2.1. Envío de la petición

Cada petición HTTP POST enviada se corresponde con un comando de la API.

El cuerpo de cada petición HTTP está compuesto por una lista de pares [nombre,valor], según la norma *application/x-www-form-encoded*.

Cada par representa un parámetro del comando. Cada comando tiene un conjunto de parámetros diferente, como se verá a continuación. Todos ellos comparten el primer parámetro, de nombre “cmd”, referido al tipo de comando que se está empleando.

La **URL** sobre la que enviar las peticiones HTTP debe ser suministrada por Altiria.

También es posible enviar la lista de pares [nombre,valor] como parte de la cadena de caracteres que conforma la URL de la petición POST, siguiendo el siguiente esquema:

```
http://url_del_servidor?nombre1=valor1&nombre2=valor2&nombre3=valor3
```

Debido a las limitaciones en el número máximo de caracteres de la URL, dependientes de numerosos factores, se desaconseja este método de envío de los parámetros.

En ningún caso se permitirán peticiones HTTP GET.

### 2.2. Codificación de caracteres

Los parámetros enviados en el cuerpo de la petición HTTP POST deben codificarse con el juego de caracteres “UTF-8”. En el apartado 2.13 se dan varios ejemplos.

En caso de enviar los parámetros como parte de la URL, será preciso codificar su valor previamente a su inclusión en la misma. Esto consiste en obtener la representación en hexadecimal en UTF-8 de cada carácter y añadirlo a la URL con un “%” para cada par de dígitos hexadecimales.

Como se ha visto la URL se compondrá a partir de varios parámetros del estilo

```
nombre1=valor1&nombre2=valor2&nombre3=valor3
```

De acuerdo a lo indicado cada uno de los valores de los parámetros deberá codificarse en UTF-8 antes de incluirlo en la URL. Por ejemplo una “ñ” se codificaría como “%C3%B1”.

Esto es especialmente necesario en los siguientes casos (se indica el carácter y su codificación en la URL en UTF-8):

+ => %2B  
% => %25  
& => %26

De cualquier modo es fundamental codificar los datos de la petición según se ha detallado, de lo contrario la interfaz HTTP podría recibir caracteres incorrectos, siendo esto especialmente grave en aquellos que conforman el mensaje corto a enviar.

## 2.3. Respuesta a la petición

Cada petición HTTP lleva asociada una respuesta desde el servidor HTTP de Altiria, variable en función del comando enviado.

En los siguientes apartados se detalla la respuesta para cada comando. De todos modos, antes de analizar la respuesta es preciso comprobar que el código de estatus devuelto por el servidor HTTP es 200, de lo contrario el resto de la respuesta no se ajustará a los patrones esperados.

Como mecanismo preventivo se recomienda establecer un tiempo máximo de espera, de modo que si la respuesta no llega antes de su vencimiento se cierre la conexión HTTP establecida y se reintente la petición de nuevo.

La respuesta a cada petición HTTP se remite codificada con el juego de caracteres “UTF-8”.

## 2.4. Crédito disponible

La única forma de averiguar si se tiene crédito suficiente para enviar los mensajes, aparte de llevar un contador propio de saldo disponible, es mediante una consulta previa a través del comando “getcredit” (ver sección 2.5.3).

Este comando ofrece información del crédito disponible en un momento dado. Puesto que el sistema decrementa el crédito justo al enviar el mensaje al destinatario, es necesario seguir el siguiente esquema para utilizar adecuadamente el comando de consulta de crédito disponible:

- Antes de comenzar con los envíos, se calcula cuantos mensajes en total se desean enviar, por ejemplo 5000.
- Se consulta el valor del crédito disponible una única vez.
- A partir del coste en créditos de cada mensaje a enviar y del saldo disponible se estima si se podrán enviar o no todos los mensajes.
- En caso positivo, se usan los comandos de envío de mensajes. En caso negativo se debe adquirir más crédito

Cuando el sistema haya finalmente enviado todos los mensajes, una nueva consulta del crédito disponible ofrecerá el valor actualizado.

En cualquier caso la comprobación efectiva del saldo disponible para efectuar un envío se realiza en un proceso interno justo antes de efectuar el envío. En caso de que no se disponga de crédito suficiente, el mensaje no será enviado y el cliente será informado a través de correo electrónico. Si posteriormente se adquiere más crédito disponible, se podrá avisar a Altiria para reintentar los envíos pendientes.

## 2.5. Comandos de la API

A continuación se detallan los comandos disponibles de la API de programación. Para cada comando se representa un cuadro con los parámetros que lo componen. Cada parámetro puede ser obligatorio u opcional y en algunos casos puede aparecer múltiples veces.

### 2.5.1. Envío de un mensaje de texto

Permite enviar un mensaje corto de texto a uno o a varios teléfonos destinatarios.

Este comando tiene la lista de parámetros del cuadro 2.1.

Para enviar el mensaje a varios destinatarios, basta repetir el parámetro "dest" tantas veces como sea preciso, asignándole cada vez el valor de un número de teléfono distinto. El número máximo de destinatarios en cada petición no está limitado, sin embargo, para minimizar los efectos de un eventual error en el protocolo HTTP, se recomienda limitar el valor en torno a los 100 destinatarios.

Las respuestas a este comando pueden ser:

- Una línea para cada destinatario, indicando alguna de las siguientes informaciones:

- En caso de éxito y de solicitar confirmación de entrega (ver sección 2.11):

```
OK dest:xxxxxxxxxxx idAck:wwwwwwwww
```

Si no se solicita la confirmación de entrega o si la petición de confirmación de entrega no es aceptada, no aparecerá la parte del "idAck".

- En caso de error:

```
ERROR dest:xxxxxxxxxxx errNum:yyy
```

- Una única línea informando de algún error general que afecta a todos los envíos. Tendrá el siguiente formato:

```
ERROR errNum:yyy
```

El valor de "dest" se corresponde con el número de teléfono del destinatario.

El valor de "errNum" se corresponde con uno de los códigos de error del apartado 2.12.

El valor de "idAck" se corresponde con el código de identificación asociado a la solicitud de confirmación de entrega (ver sección 2.11).

La información de éxito para un destinatario concreto implica que el mensaje ha sido aceptado por la pasarela, no que haya sido enviado y recibido por el destinatario. Un mensaje puede ser aceptado aún cuando no se disponga de crédito suficiente para su envío (ver sección 2.4).

Para asegurar el adecuado funcionamiento de este comando se recomienda probar la correcta recepción de todos los caracteres permitidos en un teléfono móvil antes de poner el sistema en producción.

Nombre	Valor	Obligatorio	Multiple
cmd	"sendsms"	sí	no
domainId	Identificador suministrado por <i>Altiria</i> al cliente.	sí	no
login	Identificador de usuario suministrado por <i>Altiria</i> al cliente.	sí	no
passwd	Clave del usuario suministrada por <i>Altiria</i> al cliente.	sí	no
dest	Número de teléfono móvil del destinatario del mensaje. Se especificará en formato de numeración internacional sin prefijo '00' ni el signo '+'. Ej: 34645852126. Es fundamental incluir el prefijo del país (34 para España) para que el mensaje llegue al destino esperado. No debe superar los 16 dígitos.	sí	sí
msg	Mensaje a enviar. Debería contener exclusivamente caracteres incluidos en la lista del apartado 2.6. No debe sobrepasar de 160 caracteres y tampoco debe estar vacío (cadena vacía).	sí	no
senderId	Remitente del mensaje a enviar, autorizado por <i>Altiria</i> . Valor alfanumérico de hasta 11 caracteres (números, letras de la "a" a la "z" tanto mayúsculas como minúsculas sin incluir ni "Ñ" ni "ñ" y los caracteres "." y "-"). Si no se incluye, el mensaje se enviará con el remitente por defecto seleccionado por <i>Altiria</i> .	no	no
ack	Solicitud de confirmación de entrega de los mensajes enviados (ver sección 2.11). Valores permitidos: "true" para solicitar confirmación de entrega y "false" para lo opuesto. Si aparece otro valor o no se envía este parámetro, se interpretará "false".	no	no
idAck	Código identificativo para la confirmación de entrega (ver sección 2.11). Valor alfanumérico de hasta 10 caracteres (números y letras de la "a" a la "z" tanto mayúsculas como minúsculas sin incluir ni "Ñ" ni "ñ"). De rebasar la longitud máxima permitida será truncado. Los caracteres no permitidos serán eliminados. Solo será considerado si el parámetro ack se envía con valor "true". Si se incluye explícitamente este parámetro y toma como valor cadena vacía, anula la solicitud de confirmación de entrega.	no	no
dPort	Puerto destino del SMS a enviar. Valor numérico entre 1 y 65535 (solo dígitos decimales). Solo necesario si se desea cambiar el valor habitual de esta propiedad. Si se define, la longitud del parámetro "msg" se reduce a 152 caracteres. Si solo se define "sPort", este tomará el valor 0.	no	no
sPort	Puerto origen del SMS a enviar. Valor numérico entre 1 y 65535 (solo dígitos decimales). Solo necesario si se desea cambiar el valor habitual de esta propiedad. Si se define, la longitud del parámetro "msg" se reduce a 152 caracteres. Si solo se define "dPort", este tomará el valor 0.	no	no

Cuadro 2.1: Lista de parámetros del comando sendsms

## 2.5.2. Envío de un mensaje multimedia WAP-PUSH

Permite enviar un mensaje multimedia a través de WAP-PUSH a uno o a varios teléfonos destinatarios.

Este comando tiene la lista de parámetros del cuadro 2.2.

Para conocer en qué consisten los mensajes de este tipo se aconseja leer la sección 2.9.

Para saber más sobre el proceso de envío de mensajes WAP-PUSH a través de la pasarela se aconseja leer la sección 2.10.

Para enviar el mensaje a varios destinatarios, basta repetir el parámetro "dest" tantas veces como sea preciso, asignándole cada vez el valor de un número de teléfono distinto. El número máximo de destinatarios en cada petición no está limitado, sin embargo, para minimizar los efectos de un eventual error en el protocolo HTTP, se recomienda limitar el valor en torno a los 100 destinatarios.

Las respuestas a este comando pueden ser:

- Una línea para cada destinatario, indicando alguna de las siguientes informaciones:
  - En caso de éxito, de solicitar clave para cada destinatario (ver sección 2.10) y de solicitar confirmación de entrega (ver sección 2.11):

```
OK dest:xxxxxxxxx key:xxxxxxxxx-zzzzzzzzz idAck:wwwwwwwww
```

Si no se solicita la clave para cada destinatario, no aparecerá la parte del "key".

Si no se solicita la confirmación de entrega o si la petición de confirmación de entrega no es aceptada, no aparecerá la parte del "idAck".

- En caso de error:

```
ERROR dest:xxxxxxxxx errNum:yyy
```

- Una única línea informando de algún error general que afecta a todos los envíos. Tendrá el siguiente formato:

```
ERROR errNum:yyy
```

El valor de "dest" se corresponde con el número de teléfono del destinatario.

El valor de "key" se corresponde con la clave única asociada al destinatario (ver sección 2.10).

El valor de "idAck" se corresponde con el código de identificación asociado a la solicitud de confirmación de entrega (ver sección 2.11).

El valor de "errNum" se corresponde con uno de los códigos de error del apartado 2.12.

La información de éxito para un destinatario concreto implica que el mensaje ha sido aceptado por la pasarela, no que haya sido enviado y recibido por el destinatario. Un mensaje puede ser aceptado aún cuando no se disponga de crédito suficiente para su envío (ver sección 2.4).

Para asegurar el adecuado funcionamiento de este comando se recomienda probar un ciclo completo de servicio, desde el envío del mensaje WAP-PUSH hasta la descarga del contenido en el teléfono móvil, como paso previo a poner el sistema en producción.

Nombre	Valor	Obligatorio	Múltiple
cmd	"sendwappush"	sí	no
domainId	Identificador suministrado por <i>Altiria</i> al cliente.	sí	no
login	Identificador de usuario suministrado por <i>Altiria</i> al cliente.	sí	no
passwd	Clave del usuario suministrada por <i>Altiria</i> al cliente.	sí	no
dest	Número de teléfono móvil del destinatario del mensaje. Se especificará en formato de numeración internacional sin prefijo '00' ni el signo '+'. Ej: 34645852126. Es fundamental incluir el prefijo del país (34 para España) para que el mensaje llegue al destino esperado. No debe superar los 16 dígitos.	sí	sí
msg	Texto a enviar adjunto al mensaje WAP-PUSH. Representa una breve descripción del contenido multimedia. Debería contener exclusivamente caracteres incluidos en la lista del apartado 2.7. No debe sobrepasar los 115 caracteres junto a la longitud del parámetro "url" o bien los 88 caracteres si se opta por el envío de clave en el parámetro "url" (ver sección 2.10). No puede estar vacío.	sí	no
url	Dirección de Internet desde donde el teléfono móvil se descargará el contenido. Debe contener caracteres válidos en una URL (ver sección 2.8). No debe sobrepasar los 115 caracteres junto a la longitud del parámetro "msg" o bien los 88 caracteres si se opta por el envío de clave en este parámetro (ver sección 2.10). No puede estar vacío. No es posible seleccionar un puerto de conexión diferente al 80, el habitual en la navegación WEB.	sí	no
ack	Solicitud de confirmación de entrega de los mensajes enviados (ver sección 2.11). Valores permitidos: "true" para solicitar confirmación de entrega y "false" para lo opuesto. Si aparece otro valor o no se envía este parámetro, se interpretará "false".	no	no
idAck	Código identificativo para la confirmación de entrega (ver sección 2.11). Valor alfanumérico de hasta 10 caracteres (números y letras de la "a" a la "z" tanto mayúsculas como minúsculas sin incluir ni "Ñ" ni "ñ"). De rebasar la longitud máxima permitida será truncado. Los caracteres no permitidos serán eliminados. Solo será considerado si el parámetro ack se envía con valor "true". Si se incluye explícitamente este parámetro y toma como valor cadena vacía, anula la solicitud de confirmación de entrega.	no	no

Cuadro 2.2: Lista de parámetros del comando sendwappush

### 2.5.3. Consulta del crédito disponible

Permite conocer el crédito instantáneo disponible para enviar mensajes. Consultar la sección 2.4 para conocer la forma de utilizarlo.

Este comando tiene la lista de parámetros del cuadro 2.3:

La respuesta a este comando es una única línea con el siguiente formato:

- En caso de éxito:

```
OK credit(0):xxxx
```

- En caso de error:

```
ERROR errNum:yyyy
```

El valor de “credit(0)” se corresponde con el crédito disponible. Será un número con dos decimales.

El valor de “errNum” se corresponde con uno de los códigos de error del apartado 2.12.

Nombre	Valor	Obligatorio	Multiple
cmd	”getcredit”	sí	no
domainId	Identificador suministrado por <i>Altiria</i> al cliente.	sí	no
login	Identificador de usuario suministrado por <i>Altiria</i> al cliente.	sí	no
passwd	Clave del usuario suministrada por <i>Altiria</i> al cliente.	sí	no

Cuadro 2.3: Lista de parámetros del comando getcredit

## 2.6. Caracteres permitidos para mensajes de texto

El cuadro 2.4 detalla los caracteres admisibles para los mensajes de texto.

@	(	4	-	L	W	h	s	Ú	ù
cr <sup>1</sup>	)	5	A	M	X	i	t	á	
lf <sup>2</sup>	*	6	B	N	Y	j	u	é	
Ç	+	7	C	Ñ	Z	k	v	í	
sp <sup>3</sup>	,	8	D	O	¿	l	w	ó	
!	-	9	E	P	a	m	x	ú	
”	.	:	F	Q	b	n	y	Û	
#	/	;	G	R	c	ñ	z	ü	
\$	0	<	H	S	d	o	Á	à	
%	1	=	I	T	e	p	É	è	
&	2	>	J	U	f	q	Í	ì	
'	3	?	K	V	g	r	Ó	ò	

Cuadro 2.4: Lista de caracteres permitidos para mensajes de texto

Las vocales con tilde o acento agudo (á) son aceptadas pero se enviarán al teléfono móvil sin acentuar.

En caso de que el mensaje a enviar contenga caracteres fuera de la lista presentada, estos serán reemplazados por el caracter “?” y el mensaje será enviado.

## 2.7. Caracteres permitidos para mensajes WAP-PUSH

El cuadro 2.5 detalla los caracteres admisibles para los mensajes WAP-PUSH.

cr <sup>1</sup>	lf <sup>2</sup>	sp <sup>3</sup>	!	”	#	&
'	(	)	*	+	,	-
.	/	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
Á	É	Í	Ó	Û	á	é
í	ó	ú				

Cuadro 2.5: Lista de caracteres permitidos para los mensajes WAP-PUSH

Las vocales con tilde o acento agudo (á) son aceptadas pero se enviarán al teléfono móvil sin acentuar.

<sup>1</sup>Retorno de carro

<sup>2</sup>Nueva línea

<sup>3</sup>Espacio blanco

En caso de que el mensaje a enviar contenga caracteres fuera de la lista presentada, estos serán reemplazados por el caracter “?” y el mensaje será enviado.

## 2.8. Especificación de una URL

La URL de descarga de los contenidos multimedia suministrados a través de mensajes WAP-PUSH (parámetro “url”) debe seguir las siguientes normas de composición:

- Los caracteres del cuadro 2.6 son seguros y se pueden incluir sin codificar.
- Los caracteres del cuadro 2.7 son reservados y se pueden incluir sin codificar si se emplean dentro de la URL de acuerdo a su uso reservado. Por ejemplo el caracter “&” se usa para separar los parametros de un formulario. Si estos caracteres se emplean de otro modo se deben codificar.
- Otros caracteres se pueden incluir previa codificación. De todos modos pueden no ser seguros y es posible que algunos presenten problemas en algunos teléfonos. Se recomienda prescindir de ellos siempre que sea posible.

La codificación de un carácter se logra a partir de su representación en hexadecimal en un determinado juego de caracteres, insertando el simbolo “%” por cada par de dígitos hexadecimales. Por ejemplo la “ñ” en UTF-8 se codificaría como “%C3%B1”.

El juego de caracteres a escoger debería ser el del servidor que albergue el contenido a descargar mediante el mensaje WAP-PUSH.

Según lo visto si se desea permitir la descarga de un contenido de la URL:

```
http://www.miempresa.com/contenidos/imagen[1].jpg
```

se debe enviar como (usando ISO8859-1):

```
http://www.miempresa.com/contenidos/imagen%5B1%5D.jpg
```

Es importante reseñar que cada carácter codificado ocupa un número mayor de caracteres en el cómputo de la longitud completa de la URL.

Si además el parámetro “url” se enviase como parte de la URL de la petición POST, habrá que codificarlo adecuadamente (ver sección 2.2). Para el ejemplo sería:

```
http://www.miempresa.com/contenidos/imagen%255B1%255D.jpg
```

En cualquier caso se recomienda probar la correcta descarga de los contenidos desde la URL seleccionada para comprobar que todo el proceso se efectúa correctamente.

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	0	1	2	3
4	5	6	7	8	9	-	-	.	!	*	'	(	)

Cuadro 2.6: Lista de caracteres seguros

\$	&	+	,	/	:	;	=	?	@
----	---	---	---	---	---	---	---	---	---

Cuadro 2.7: Lista de caracteres reservados

## 2.9. Mensajes multimedia WAP-PUSH

La interfaz HTTP de *Altiria* permite el envío de mensajes multimedia (imágenes, sonidos, juegos, etc) mediante la tecnología de los mensajes WAP-PUSH.

Los mensajes WAP-PUSH incluyen información sobre la ubicación de un determinado contenido multimedia, una dirección de Internet. En este sentido son completamente diferentes a los mensajes de texto normales, puesto que en estos el contenido relevante es el propio texto.

Básicamente un mensaje de este tipo se compone de un pequeño texto a modo de presentación del contenido que se ofrece y la dirección de Internet donde se ubica dicho contenido.

Cuando un teléfono móvil recibe un mensaje WAP-PUSH, le presenta al usuario la breve descripción mencionada junto con la posibilidad de descargarse el contenido multimedia referenciado. Si el usuario acepta, el teléfono de manera automática accede al contenido a través de HTTP, se lo descarga como si de un navegador WEB se tratara y lo almacena, mostrando además otras opciones en función del tipo de contenido (ver una imagen, reproducir un sonido...).

### 2.9.1. Tipos de contenido

El contenido más general que se puede enviar es una página “wml”. Las páginas “wml” son similares a las conocidas páginas web, adaptadas a los requisitos de un teléfono móvil.

De este modo se podrá enviar un contenido formado por texto y otros tipos de archivos como imágenes (ej: jpg y gif) o sonidos (ej: midi), incluidos en la propia página.

También es posible enviar directamente el archivo multimedia al teléfono, evitando incluirlo en una página “wml”.

Actualmente hay mucha diversidad de teléfonos móviles, cada uno con sus propias capacidades multimedia. Es posible que determinados teléfonos no sean capaces de manejar algunos tipos de archivos. Para esas situaciones, la posibilidad de incluir texto en una página “wml”, un recurso manejado por todos los terminales WAP, permite al menos que el teléfono acceda a parte de la información. A este respecto una buena práctica es incluir en el texto información para el destinatario sobre la opción de acceder al fichero multimedia a través de un navegador web convencional, adjuntando la información relativa a la dirección de Internet.

En caso de optar por la inclusión de texto en una página “wml” se recomienda emplear un juego de caracteres sencillo, reconocible por la mayoría de los teléfonos. Como referencia se puede usar el detallado en el cuadro 2.5, sin incluir las vocales acentuadas.

### 2.9.2. Formato del contenido

Independientemente del tipo de contenido escogido, siempre se debería considerar que el medio habitual de acceso al mismo será un teléfono móvil.

Esto tiene importantes incidencias en cuanto al tamaño máximo de la información suministrada. Se recomienda no enviar contenidos que ocupen más de 10kB, sobre todo si se suministran embebidos en páginas “wml”.

Para optimizar el tamaño, se sugiere adaptar los contenidos a los requerimientos de un teléfono móvil. Por ejemplo si se trata de una imagen es conveniente ajustar su tamaño al habitual de la pantalla, guardando además una relación de aspecto adecuada para que al recibirla ocupe el máximo en todas las direcciones. Una buena medida como referencia pueden ser 120 x 120 “pixels”.

### 2.9.3. Dirección del contenido

El teléfono móvil conoce la ubicación del contenido multimedia mediante la información de dirección que le llega en el mensaje WAP-PUSH.

Es obvio que para que el teléfono pueda descargarse la información la dirección debe respresentar la ubicación de un recurso accesible públicamente a través de HTTP, mediante navegación WEB.

Un detalle importante asociado a la dirección del contenido es que muchos teléfonos la emplean como identificador de los mensajes WAP-PUSH recibidos. Esto supone que si se recibe un mensaje WAP-PUSH con la misma dirección del contenido asociado que un mensaje ya recibido y almacenado en el teléfono, el nuevo mensaje reemplazará al antiguo.

Existen sin embargo teléfonos que no siguen este patrón y almacenan los dos mensajes con idéntica dirección del contenido de forma independiente.

### 2.9.4. Envío del contenido

Cuando el teléfono móvil solicita el contenido referenciado en el mensaje WAP-PUSH, envía una petición HTTP GET (en algunos casos se envía un HTTP HEAD previamente) a la dirección apropiada.

Es necesario entonces un servidor HTTP que atienda la petición y entregue el contenido apropiadamente.

## 2.10. Envío de mensajes WAP-PUSH

La URL indicada en el campo “url” del comando “sendwappush” será enviada en el mensaje WAP-PUSH a cada destinatario para que los interesados se descarguen el contenido ahí alojado. Para poder distinguir qué destinatarios han accedido realmente al contenido *Altiria* ofrece la posibilidad de enviar un mensaje diferente a cada uno de ellos. A la URL a suministrar al teléfono se le añadirá un parámetro identificador, una clave. Este parámetro estará unívocamente asociado al teléfono del destinatario mediante la respuesta generada por la pasarela al comando “sendwappush”. Cuando se reciba una petición de descarga se podrá extraer la clave y de esta manera obtener información del destinatario.

Si se desea solicitar el envío de clave es preciso que la URL suministrada en el parámetro “url” termine con la subcadena “k=”. *Altiria* agregará en ese caso a la URL enviada en cada mensaje la clave única con el formato: “xxxxxxxxxxx-zzzzzzzzzz”; “xxxxxxxxxxx” representa el número de teléfono del destinatario en formato internacional y “zzzzzzzzzz” representa un número asociado a cada comando “sendwappush”, como máximo de diez cifras.

Es necesario resaltar que si se opta por este método la longitud total para los parametros “msg” y “url” pasará de 115 caracteres a 88.

Finalmente para clarificar todos los elementos del servicio de envío de mensajes multimedia a través del comando “sendwappush” se esquematizan los procesos involucrados en el envío de un mensaje a dos destinatarios solicitando claves independientes:

1. El cliente efectúa una petición HTTP POST con el comando “sendwappush” a la pasarela de *Altiria* , incluyendo los siguientes parámetros:
  - dest=346xxxxxxxx.
  - dest=346yyyyyyyy.
  - url=www.miempresa.com/contenidos/descarga.php?k=

Se observa que se desea enviar el contenido multimedia a dos destinatarios y que además la URL acaba con la subcadena “k=”, solicitando entonces la generación de claves identificadoras.

2. La pasarela de *Altiria* recibe la petición y remite dos mensajes WAP-PUSH individuales a los destinatarios seleccionados. Como URL de descarga, los respectivos teléfonos móviles recibirán (la segunda parte de la clave es simplemente un ejemplo) :

- El móvil “346xxxxxxxx”: `www.miempresa.com/contenidos/descarga.php?k=346xxxxxxxx-12365`
- El móvil “346yyyyyyyy”: `www.miempresa.com/contenidos/descarga.php?k=346yyyyyyyy-12365`

Además la pasarela de *Altiria* responde a la petición HTTP POST del cliente las siguientes dos líneas:

```
OK dest:346xxxxxxxx key:346xxxxxxxx-12365
OK dest:346yyyyyyyy key:346yyyyyyyy-12365
```

3. Ambos destinatarios reciben en su teléfono la solicitud de aceptación de descarga del contenido multimedia. Suponemos que el destinatario con el número de teléfono “346xxxxxxxx” acepta la descarga.
4. El servidor WEB del cliente, habilitado para ofrecer los contenidos multimedia, recibe una petición de descarga. La petición estará dirigida a la URL

```
http://www.miempresa.com/contenidos/descarga.php?k=346xxxxxxxx-12365
```

por lo que podrá asociarla directamente al destinatario con número de teléfono “346xxxxxxxx”.

5. El servidor WEB entrega el contenido al teléfono del destinatario.
6. El teléfono del destinatario muestra el contenido.

## 2.11. Confirmación de entrega

El servicio de confirmación de entrega, solicitado a través del parámetro “ack” de los comandos de envío, permite recibir notificaciones con información sobre el estado de entrega de los mensajes cortos enviados mediante la pasarela.

Para tener acceso a este servicio es preciso que el cliente haya notificado a *Altiria* la dirección de Internet a donde se enviarán las informaciones de confirmación de entrega. En caso contrario, las solicitudes de confirmación de entrega serán ignoradas aunque los mensajes sí serán enviados.

Para que el cliente pueda asociar la información recibida sobre el estado de entrega de un mensaje con el propio mensaje enviado previamente a través de la pasarela, se usará el identificador devuelto en la respuesta a los comandos de envío en la parte “idAck”.

Este identificador puede albergar dos tipos de valores:

- Si en el propio comando de envío se incluye el parámetro “idAck” (es opcional), contendrá ese valor truncado a diez caracteres y formado solo por caracteres válidos. Es importante constatar que el identificador devuelto en este caso solo coincidirá con el suministrado en el correspondiente comando de envío si cumple los criterios de composición explicados en las tablas 2.1 y 2.2.
- Si en el propio comando de envío no se incluye el parámetro “idAck”, contendrá un valor numérico de diez dígitos como máximo generado por la pasarela automáticamente.

Las notificaciones del estado de entrega serán enviadas a través de peticiones HTTP POST dirigidas a la URL donde el cliente habrá configurado previamente un servidor HTTP a la escucha. La petición POST incluirá un solo parámetro, “notification=telefono,idAck,estado”, que podrá aparecer repetido

varias veces, una por cada notificación diferente que se envíe. Se observa entonces que es posible recibir en la misma petición HTTP varias notificaciones agrupadas. Los criterios de agrupación de notificaciones no responderán a ningún parámetro fijo, de modo que nada se podrá predecir a ese respecto.

El contenido detallado de la petición HTTP POST enviada será del tipo (content-type):

“application/x-www-form-urlencoded”, codificado con el juego de caracteres “UTF-8”.

Cada parametro “notification” incluye los datos del cuadro 2.8, separados por comas:

Dato	Valor
telefono	Número de teléfono móvil al que se refiere la información de estado de entrega.
idAck	Identificador que coincidirá con el suministrado por <i>Altiria</i> al cliente en la parte “idAck” de la respuesta al comando de envío. El contenido de este campo se ha explicado en el inicio de esta sección.
estado	Estado relativo a la entrega del mensaje. Podrá tomar los valores: “ENTREGADO”, “NO ENTREGADO”, “ERROR_100” o “ERROR_101”.

Cuadro 2.8: Lista de datos del parámetro notification

El estado “NO ENTREGADO” aparece cuando el mensaje no puede ser entregado al teléfono móvil. Es un estado definitivo, por lo que ese mensaje particular nunca será entregado. Las causas pueden ser múltiples, desde que el teléfono haya estado apagado durante un tiempo superior al periodo de validez, hasta que el número no exista. En general no se puede conocer la causa.

El estado “ERROR\_100” indica que el mensaje por el momento no ha podido ser entregado al destinatario debido a algún problema en su teléfono móvil. Las causas más comunes son: mala cobertura, buzón de mensajes cortos lleno o teléfono apagado. El mensaje se intentará enviar varias veces con posterioridad durante un tiempo limitado. Si el problema en el teléfono se subsana a tiempo, el mensaje será finalmente entregado, recibándose la correspondiente confirmación.

El estado “ERROR\_101” indica que el mensaje por el momento no ha podido ser entregado al destinatario debido a algún problema en la red de telefonía móvil del operador. Habitualmente, cuando el operador solventa los problemas, el mensaje será entregado, recibándose la correspondiente confirmación.

El servidor HTTP del cliente deberá responder a la petición POST con un código de estatus 200 y un cuerpo tipo “text/plain” con un contenido simple, preferentemente una cadena de texto vacía o algo sencillo como “OK”.

Finalmente para clarificar todos los elementos de la funcionalidad de confirmación de entrega, se esquematizan los procesos involucrados en el envío de un mensaje a dos destinatarios:

1. El cliente efectúa una petición HTTP POST con el comando “sendsms” a la pasarela de *Altiria*, incluyendo entre otros los siguientes parámetros:
  - dest=346xxxxxxx.
  - dest=346yyyyyyy.
  - ack=true.
  - idAck=zzzz.
2. La pasarela de *Altiria* recibe la petición y remite el mensaje a los destinatarios seleccionados. Además la pasarela responde a la petición HTTP POST del cliente las siguientes dos líneas:

```
OK dest:346xxxxxxxx idAck:zzzz
OK dest:346yyyyyyyy idAck:zzzz
```

Si el cliente no hubiera incluido el parámetro “idAck” en el comando “sendsms” (punto 1 del ejemplo), la pasarela de *Altiria* habría autogenerado el identificador para añadirlo a la respuesta.

3. Ambos destinatarios reciben en su teléfono el mensaje corto enviado.
4. El servidor HTTP del cliente, habilitado para recibir las notificaciones de estado de entrega, recibe una petición HTTP POST desde la pasarela de *Altiria* , con el parámetro “notification” por duplicado, conteniendo:

```
‘notification=346xxxxxxxx,zzzz,ENTREGADO’
‘notification=346yyyyyyyy,zzzz,ENTREGADO’
```

5. El servidor HTTP del cliente responde a la petición POST el código de estatus 200 y una cadena de texto vacía en formato “text/plain”.
6. Empleando el identificador zzzz, el cliente podrá asociar la confirmación de entrega con el mensaje previamente enviado a cada uno de los destinatarios.

## 2.12. Códigos de error

El cuadro 2.9 presenta la lista de los posibles códigos de error que podrán aparecer en la respuesta a cada comando.

CÓDIGO	DETALLE
001	Error interno. Contactar con el soporte técnico
010	Error en el formato del número de teléfono
011	Error en el envío de los parámetros del comando o codificación incorrecta.
013	El mensaje excede la longitud máxima permitida
014	La petición HTTP usa una codificación de caracteres inválida
015	No hay destinatarios válidos para enviar el mensaje
016	Destinatario duplicado
017	Mensaje vacío
020	Error en la autenticación
022	El remitente seleccionado para el envío no es válido
030	La url y el mensaje superan la longitud máxima permitida
031	La longitud de la url es incorrecta
032	La url contiene caracteres no permitidos
033	El puerto destino del SMS es incorrecto
034	El puerto origen del SMS es incorrecto

Cuadro 2.9: Lista de los códigos de error

## 2.13. Ejemplos

Se presentan extractos de programación en varios lenguajes.

### 2.13.1. Envío de un mensaje en PHP

```
<?
// sDestination: lista de números, comenzando por 34 y separados por comas
// sMessage: hasta 160 caracteres
// XX, YY y ZZ se corresponden con los valores de identificación del
// usuario en el sistema.
// Como ejemplo la petición se envía a www.altiria.net/sustituirPOSTsms
// Se debe reemplazar la cadena '/sustituirPOSTsms' por la parte correspondiente
// de la URL suministrada por Altiria al dar de alta el servicio

function AltiriaSMS($sDestination,$sMessage) {
    $sData = "cmd=sendsms&domainId=XX&login=YY&passwd=
        ZZ&dest=".str_replace(",","&dest=",$sDestination)."&msg="
        .urlencode(utf8_encode(substr($sMessage,0,160)));
    $fp = fsockopen("www.altiria.net", 80);

    // Reemplazar la cadena '/sustituirPOSTsms' por la parte correspondiente
    // de la URL suministrada por Altiria al dar de alta el servicio
    $buf = "POST /sustituirPOSTsms HTTP/1.0\r\n";
    $buf .= "Host: www.altiria.net\r\n";
    $buf .= "Content-type: application/x-www-form-urlencoded; charset=UTF-8\r\n";
    $buf .= "Content-length: ".strlen($sData)."\r\n";
    $buf .= "\r\n";
    $buf .= $sData;
    fputs($fp, $buf);
    $buf = "";
    while (!feof($fp))
        $buf .= fgets($fp,128);
    fclose($fp);
    if (strstr($buf,"ERROR"))
        return $buf;
    else
        return "";
}

/*
$resp= AltiriaSMS("34600111222,34600111333", "Texto de prueba");
if (!$resp)
    print "Mensaje enviado correctamente!\n";
else
    echo strstr($resp,"ERROR");
*/
?>
```

### 2.13.2. Envío de un mensaje en JAVA

Ejemplo en Java utilizando HttpClient como cliente HTTP (ver [HTTPCLIENT]):

```
//Se inicia el objeto HTTP
HttpClient client = new HttpClient();
client.setStrictMode(true);

//Se fija el tiempo máximo de espera de la respuesta del servidor
client.setTimeout(60000);

//Se fija el tiempo máximo de espera para conectar con el servidor
client.setConnectionTimeout(5000);

PostMethod post = null;

//Se fija la URL sobre la que enviar la petición POST
//Como ejemplo la petición se envía a www.altiria.net/sustituirPOSTsms
//Se debe reemplazar la cadena '/sustituirPOSTsms' por la parte correspondiente
//de la URL suministrada por Altiria al dar de alta el servicio
post = new PostMethod("http://www.altiria.net/sustituirPOSTsms");

//Se fija la codificación de caracteres en la cabecera de la petición
post.setRequestHeader("Content-type",
    "application/x-www-form-urlencoded; charset=UTF-8");

//Se crea la lista de parámetros a enviar en la petición POST
NameValuePair[] parametersList = new NameValuePair[7];

//XX, YY y ZZ se corresponden con los valores de identificación del
//usuario en el sistema.
parametersList[0] = new NameValuePair("cmd", "sendsms");
parametersList[1] = new NameValuePair("domainId", "XX");
parametersList[2] = new NameValuePair("login", "YY");
parametersList[3] = new NameValuePair("passwd", "ZZ");
parametersList[4] = new NameValuePair("dest", "34600111222");
parametersList[5] = new NameValuePair("dest", "34600111333");
parametersList[6] = new NameValuePair("msg", "Texto de prueba");

//Se rellena el cuerpo de la petición POST con los parámetros
post.setRequestBody(parametersList);

int httpstatus = 0;
String response = null;

try {
    //Se envía la petición
    httpstatus = client.executeMethod(post);

    //Se consigue la respuesta
    response = post.getResponseBodyAsString();
}
catch (Exception e) {
    //Habrà que prever la captura de excepciones
}
```

```
    return;
}
finally {
    //En cualquier caso se cierra la conexión
    post.releaseConnection();
}
//Habrá que prever posibles errores en la respuesta del servidor
if (httpStatus!=200){
    return;
}
else {
    //Se procesa la respuesta capturada en la cadena ‘‘response’’
}
```

### 2.13.3. Envío de un mensaje en Visual Basic

Ejemplo en Visual Basic 6.0 usando ServerXMLHTTP como cliente HTTP (ver [SERVERXMLHTTPFAQ] y [SERVERXMLHTTAPI]):

```
Dim objXSH As New MSXML2.ServerXMLHTTP50
Dim strPostText As String
Dim sResponseText As String

'Se fija la URL sobre la que enviar la petición POST
'Como ejemplo la petición se envía a www.altiria.net/sustituirPOSTsms
'Se debe reemplazar la cadena /sustituirPOSTsms por la parte correspondiente
'de la URL suministrada por Altiria al dar de alta el servicio
objXSH.open "POST", "http://www.altiria.net/sustituirPOSTsms", False
objXSH.setRequestHeader "Content-Type", "application/x-www-form-urlencoded;charset=UTF-8"

'XX, YY y ZZ se corresponden con los valores de identificación del usuario en el sistema.
strPostText=
    "cmd=sendsms&domainId=XX&login=YY&passwd=ZZ&dest=34600111222&msg=Texto de prueba"

objXSH.send strPostText

If objXSH.Status = 200 Then
    'La respuesta se debe usar para filtrar el resultado devuelto por la pasarela
    sResponseText = objXSH.responseText
Else
    Debug.Print "Error: (" & objXSH.Status & ") " & objXSH.statusText
End If
```

#### 2.13.4. Envío de un mensaje en .NET

##### Ejemplo en C#

Ejemplo en C# usando `HttpRequest` como cliente HTTP (ver [HTTPWEBREQUESTAPI]):

```
//Se fija la URL sobre la que enviar la petición POST
//Como ejemplo la petición se envía a www.altiria.net/sustituirPOSTsms
//Se debe reemplazar la cadena '/sustituirPOSTsms' por la parte correspondiente
//de la URL suministrada por Altiria al dar de alta el servicio
HttpRequest loHttp =
    (HttpRequest) WebRequest.Create("http://www.altiria.net/sustituirPOSTsms");

// Compone el mensaje a enviar
// XX, YY y ZZ se corresponden con los valores de identificación del usuario en el sistema.
string lcPostData =
    "cmd=sendsms&domainId=XX&login=YY&passwd=ZZ&dest=34600111222&msg=Texto de prueba";

// Lo codifica en utf-8
byte [] lbPostBuffer = System.Text.Encoding.GetEncoding("utf-8").GetBytes(lcPostData);

loHttp.Method="POST";
loHttp.ContentType="application/x-www-form-urlencoded";
loHttp.ContentLength = lbPostBuffer.Length;

// Envía la petición
Stream loPostData = loHttp.GetRequestStream();
loPostData.Write(lbPostBuffer,0,lbPostBuffer.Length);
loPostData.Close();

// Prepara el objeto para obtener la respuesta
HttpWebResponse loWebResponse = (HttpWebResponse) loHttp.GetResponse();

// La respuesta vendrá codificada en utf-8
Encoding enc = System.Text.Encoding.GetEncoding("utf-8");

StreamReader loResponseStream =
    new StreamReader(loWebResponse.GetResponseStream(),enc);

// Conseguimos la respuesta en una cadena de texto
string lcHtml = loResponseStream.ReadToEnd();

loWebResponse.Close();
loResponseStream.Close();
```

## Ejemplo en Visual Basic

Ejemplo en Visual Basic usando `HttpWebRequest` como cliente HTTP (ver [HTTPWEBREQUESTAPI]):

```
'Se fija la URL sobre la que enviar la petición POST
'Como ejemplo la petición se envía a www.altiria.net/sustituirPOSTsms
'Se debe reemplazar la cadena /sustituirPOSTsms por la parte correspondiente
'de la URL suministrada por Altiria al dar de alta el servicio
Dim loHttp As HttpWebRequest
loHttp =
    CType(WebRequest.Create("http://www.altiria.net/sustituirPOSTsms"), HttpWebRequest)

'Compone el mensaje a enviar
'XX, YY y ZZ se corresponden con los valores de identificación del usuario en el sistema.
Dim lcPostData As String =
    "cmd=sendsms&domainId=XX&login=YY&passwd=ZZ&dest=34600111222&msg=Texto de prueba"

'Lo codifica en utf-8
Dim lbPostBuffer As Byte() =
    System.Text.Encoding.GetEncoding("utf-8").GetBytes(lcPostData)

loHttp.Method = "POST"
loHttp.ContentType = "application/x-www-form-urlencoded"
loHttp.ContentLength = lbPostBuffer.Length

'Envía la petición
Dim loPostData As System.IO.Stream = loHttp.GetRequestStream()
loPostData.Write(lbPostBuffer, 0, lbPostBuffer.Length)
loPostData.Close()

'Prepara el objeto para obtener la respuesta
Dim loWebResponse As HttpWebResponse = CType(loHttp.GetResponse(), HttpWebResponse)

'La respuesta vendrá codificada en utf-8
Dim enc As System.Text.Encoding = System.Text.Encoding.GetEncoding("utf-8")

Dim loResponseStream As System.IO.StreamReader =
    New System.IO.StreamReader(loWebResponse.GetResponseStream(), enc)

'Conseguimos la respuesta en una cadena de texto
Dim lcHtml As String = loResponseStream.ReadToEnd()

loWebResponse.Close()
loResponseStream.Close()
```

### Ejemplo en ASP 3.0

Ejemplo en ASP 3.0 usando ServerXMLHTTP como cliente HTTP (ver [SERVERXMLHTTPFAQ] y [SERVERXMLHTTPAPI]):

```
Set objXSH = Server.CreateObject("Msxml2.ServerXMLHTTP")

Dim strPostText, sResponseText

'Se fija la URL sobre la que enviar la petición POST
'Como ejemplo la petición se envía a www.altiria.net/sustituirPOSTsms
'Se debe reemplazar la cadena /sustituirPOSTsms por la parte correspondiente
'de la URL suministrada por Altiria al dar de alta el servicio
objXSH.open "POST", "http://www.altiria.net/sustituirPOSTsms", False
objXSH.setRequestHeader
    "Content-Type", "application/x-www-form-urlencoded; charset=UTF-8"

'XX, YY y ZZ se corresponden con los valores de identificación del usuario en el sistema.
strPostText =
    "cmd=sendsms&domainId=XX&login=YY&passwd=ZZ&dest=34600111222&msg=Texto de prueba"

objXSH.send strPostText

If objXSH.Status = 200 Then
    'La respuesta se debe usar para filtrar el resultado devuelto por la pasarela
    sResponseText = objXSH.responseText
Else
    Debug.Print "Error: (" & objXSH.Status & ") " & objXSH.statusText
End If
Set objXSH = Nothing
```

### 2.13.5. Envío de un mensaje en Borland C++ Builder

Los siguientes pasos se han seguido en la versión 6 del paquete de desarrollo:

1. Incluir el componente TIdHTTP
2. Configurar las cabeceras asignando a la propiedad

```
"Request->ContentType"
```

el valor

```
"application/x-www-form-urlencoded; charset=UTF-8"
```

3. Usar el componente de la siguiente forma:

```
//XX, YY y ZZ se corresponden con los valores de identificación del usuario en el sistema.
AnsiString SParametros= String("cmd=sendsms") + "&" +
    String("domainId=XX") + "&" +
    String("login=YY") + "&" +
    String("passwd=ZZ") + "&" +
    String("dest=34600111222") + "&" +
```

```
String("msg=") + UTF8Encode("Texto de prueba");
```

```
//Se fija la URL sobre la que enviar la petición POST
//Como ejemplo la petición se envía a www.altiria.net/sustituirPOSTsms
//Se debe reemplazar la cadena '/sustituirPOSTsms' por la parte correspondiente
//de la URL suministrada por Altiria al dar de alta el servicio
AnsiString SUrl= "http://www.altiria.net/sustituirPOSTsms";

// Enviamos un mensaje
TStringStream *sr=new TStringStream(SParametros);
TStringStream *ss=new TStringStream("");

IdHTTP->Post(SUrl,sr,ss); // En "ss" se recibe la respuesta del servidor
```

### 2.13.6. Envío de un mensaje en Borland Delphi

Los siguientes pasos se han seguido en la versión 7 del paquete de desarrollo:

1. Incluir el componente TIdHTTP ubicado en la paleta de componentes Indy Clients.
2. Configurar las cabeceras asignando a la propiedad

```
"Request->ContentType"
```

el valor

```
"application/x-www-form-urlencoded; charset=UTF-8"
```

3. Usar el componente de la siguiente forma:

```
procedure TFormEnviar.btnEnviarClick(Sender: TObject);
var
  DResultado,SUrl : String;
  Parametros      : TStringList;
begin

  //Se fija la URL sobre la que enviar la petición POST
  //Como ejemplo la petición se envía a www.altiria.net/sustituirPOSTsms
  //Se debe reemplazar la cadena '/sustituirPOSTsms' por la parte correspondiente
  //de la URL suministrada por Altiria al dar de alta el servicio
  SUrl:='http://www.altiria.net/sustituirPOSTsms';

  //Compono el mensaje a enviar
  //XX, YY y ZZ se corresponden con los valores de identificación del usuario en el sistema
  Parametros := TStringList.Create;
  Parametros.Add('cmd=sendsms');
  Parametros.Add('domainId=XX');
  Parametros.Add('login=YY');
  Parametros.Add('passwd=ZZ');
  Parametros.Add('dest=34600111222');
  Parametros.Add('msg='+UTF8Encode('Texto de prueba'));
```

```
// Enviamos un mensaje, recibiendo en "DResultado" la respuesta del servidor
DResultado:=IdHTTP1.Post(SUrl,Parametros);

Parametros.free;
end;
```

## Capítulo 3

# Tarifas y cobertura internacional

Las tarifas actualizadas de envío se pueden consultar en la pagina:

**[http://www.altiria.com/portal/web/sms\\_mms/tarifas\\_envio\\_sms](http://www.altiria.com/portal/web/sms_mms/tarifas_envio_sms)**

Respecto a la lista de países en que el servicio está disponible, detallando las operadoras y las restricciones geográficas por país, se puede consultar en:

**[http://www.altiria.com/portal/web/sms\\_mms/enviar\\_SMS\\_operadores\\_moviles\\_internacional](http://www.altiria.com/portal/web/sms_mms/enviar_SMS_operadores_moviles_internacional)**

# Referencias

[FAQ] *Preguntas frecuentes de la pasarela de envío de SMS de Altiria:*

[http://www.altiria.com/web/sms\\_mms/faq\\_api\\_sdk\\_pasarela\\_envio\\_sms](http://www.altiria.com/web/sms_mms/faq_api_sdk_pasarela_envio_sms)

[HTTPCLIENT] *El proyecto HTTPCLIENT de Jakarta:*

<http://jakarta.apache.org/commons/httpclient>

[SERVERXMLHTTPFAQ] *FAQ sobre el componente ServerXMLHTTP:*

<http://support.microsoft.com/default.aspx?kbid=290761&product=msxml>

[SERVERXMLHTTPAPI] *API del componente ServerXMLHTTP:*

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/xmlsdk/html/xmobjpmeserverxmlhttp.asp>

[HTTPWEBREQUESTAPI] *API del componente HttpWebRequest:*

<http://msdn.microsoft.com/es-es/library/system.net.httpwebrequest.aspx>