# Especificaciones de la Interfaz HTTP para envío de SMS

**Altiria TIC, S.L.**

**Versión: 2.0**

**Historial de cambios**

| Versión | Cambios |
|---|---|
| 2.0 | Se añaden los códigos de error 018 y 019 (sección 2.8).<br>Se admite la autentificación sin el parámetro "domainId" en caso de que el login sea un email.<br>Se actualizan los ejemplos de programación (sección 2.9). |
| 1.16 | Se añade dos nuevos estados de confirmación de entrega: "ERROR_114" y "ERROR_115" (cuadro 2.9).<br>Se actualizan los ejemplos de programación (sección 2.9). |
| 1.15 | Se modifica la lista de caracteres válidos para el remitente de los mensajes así como la gestión de los caracteres inválidos en el parámetro "senderId" (sección 2.4.1). |
| 1.14 | Se reorganiza el documento para agrupar la información de forma más clara.<br>El parámetro "idAck" de los comandos de envío pasa a admitir hasta 20 caracteres en lugar de 10 (secciones 2.4.1 y 2.4.2).<br>Se añade el código de error 002 (sección 2.8).<br>Se precisa la información sobre el parámetro "senderId" para permitir el envío de remitentes numéricos (sección 2.4.1).<br>Se añade la posibilidad de enviar mensajes concatenados mediante el parámetro "concat" (sección 2.4.1).<br>Se añade la posibilidad de enviar mensajes codificados en UNICODE mediante el parámetro "encoding" (sección 2.4.1).<br>Se amplía la lista de caracteres permitidos para los mensajes SMS de texto en la codificación por defecto añadiendo los caracteres extendidos (sección 2.5.1).<br>Se actualizan las referencias bibliográficas del final del documento. |
| 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 2.9).<br>Se detalla el modo de codificar los caracteres si los parámetros de la petición se envían en la URL (sección 2.2).<br>Se precisa la información relativa a la especificación de una URL en mensajes WAP-PUSH (sección 2.6.2).<br>Se añaden dos parámetros opcionales al comando "sendsms" para permitir especificar el puerto origen y destino del SMS a enviar (sección 2.4.1).<br>Se añaden nuevos códigos de error correspondientes a los nuevos parámetros: 033 y 034 (sección 2.8). |

**Historial de cambios**

| Versión | Cambios |
|---|---|
| 1.11 | Se incluye el enlace a la página que detalla la lista de paises permitidos y las restricciones geográficas aplicables (sección 1).<br>Se añaden las vocales con acento grave (à) a la lista de caracteres permitidos para los mensajes SMS de texto (sección 2.5.1).<br>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 (secciónes 2.5.1) y 2.6.1).<br>Se actualiza la información relativa a la consulta de crédito disponible, previa el envío de mensajes (sección 2.4.3).<br>Se incluye un capítulo con enlaces a las tarifas y a la información de cobertura internacional. |
| 1.10 | Se actualiza la lista de caracteres permitidos para mensajes de texto, desaconsejando el de apertura de exclamación (cuadro 2.4).<br>Se detalla la respuesta a la peticion HTTP POST con datos de confirmacion de entrega (sección 2.7). |
| 1.9 | Añadido un nuevo ejemplo de programación (sección 2.9).<br>Modificados los posibles estados a recibir como información de confirmación de entrega (cuadro 2.9). |
| 1.8 | Se añade la sección 2.7 relativa a la nueva funcionalidad de la confirmación de entrega.<br>Se reestructura la documentación del comando "sendwappush" (sección 2.4.2) y se añade la sección 2.6.7, aclaratoria a ese respecto.<br>Se añaden nuevos parámetros a los comandos "sendsms" (sección 2.4.1) y "sendwappush" (sección 2.4.2) para gestionar la confirmación de entrega.<br>Se añade limitación de longitud y de caracteres permitidos en el parámetro senderId del comando "sendsms" (sección 2.4.1).<br>Añadido un nuevo código de error: 032 (sección 2.8). |
| 1.7 | Añadidos nuevos ejemplos de programación (sección 2.9). |
| 1.6 | Nuevo comando para el envío de mensajes multimedia WAP-PUSH (sección 2.4.2).<br>Nuevo apartado sobre los mensajes WAP-PUSH (sección 2.6).<br>Añadidos nuevos código de error: 030, 031 y eliminado el 021 (sección 2.8).<br>Apartado sobre las posibilidades de comprobar el credito disponible (sección 2.4.3).<br>El comando "getcredit" devuelve un valor numérico con dos decimales (sección 2.4.3).<br>Apartado sobre las reglas de composicion de una URL para los mensajes multimedia WAP-PUSH (sección 2.6.2). |

# Índice general

# 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 paises. Para conocer los paises 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) se puede enviar un correo electrónico a comercial@altiria.com.

El servicio opcional de confirmación de entrega requiere que el cliente configure un servidor de peticiones HTTP para recibir la información de confirmación (ver la sección 2.7).

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

La **URL** sobre la que envíar las peticiones HTTP es **http://www.altiria.net/api/http**

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.

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://www.altiria.net/api/http?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 2.9 se dan varios ejemplos.

En caso de enviar los parametros 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. Comandos de la API

A continuación se detallan los comandos disponibles en la API. 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.

Los comandos se distinguen entre sí mediante el parámetro “cmd”, obligatorio en todos los casos.

### 2.4.1. 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 sin sobrepasar el límite máximo permitido (consultar al soporte técnico de *Altiria* en soporte@altiria.com), asignándole cada vez el valor de un número de teléfono distinto (los teléfonos repetidos son descartados). Se recomienda en cualquier caso no exceder de 100 destinatarios por petición.

| **Nombre** | **Valor** | **Obligatorio** |
|---|---|---|
| cmd | ”sendsms” | sí |
| domainId | Identificador suministrado por *Altiria* al cliente. Se puede omitir si el login es un email. | no |
| login | Identificador de usuario suministrado por *Altiria* al cliente. | sí |
| passwd | Clave del usuario suministrada por *Altiria* al cliente. | sí |
| 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 |

| | | |
|---|---|---|
| msg | Mensaje a enviar. La lista de caracteres válidos y la longitud máxima permitida se detalla en la sección 2.5. No puede estar vacío (cadena vacía). | sí |
| senderId | Remitente del mensaje a enviar, autorizado por *Altiria* . La posibilidad de personalizar el remitente depende del país destinatario del mensaje. Puede tomar dos posibles valores: 1) valor alfanumérico de hasta 11 caracteres (números y letras de la "a" a la "z" tanto mayúsculas como minúsculas excluyendo la "Ñ" y la "ñ"); 2) valor numérico de hasta 15 dígitos decimales comenzando por el carácter "+". **Los caracteres inválidos serán suprimidos automáticamente**. Si se pretende que el receptor pueda responder al mensaje corto recibido se debería usar un remitente numérico (opción 2) incluyendo el prefijo de país. Si no se incluye, el mensaje se enviará con el remitente por defecto seleccionado por Altiria. | no |
| ack | Solicitud de confirmación de entrega de los mensajes enviados (ver sección 2.7). Si vale "true" solicita confirmación de entrega de los SMS enviados. En su ausencia o si tiene otro valor no se solicita confirmación de entrega. | no |
| idAck | Código identificativo para la confirmación de entrega (ver sección 2.7). Valor alfanumérico de hasta 20 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 |
| 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 máxima del mensaje a enviar (parámetro "msg") se verá reducida (ver la sección 2.5.3) y se invalidará la posibilidad de enviar mensajes concatenados (ver parámetro "concat"). Si solo se define "sPort", este tomará el valor 0. | 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 máxima del mensaje a enviar (parámetro "msg") se verá reducida (ver la sección 2.5.3) y se invalidará la posibilidad de enviar mensajes concatenados (ver parámetro "concat"). Si solo se define "dPort", este tomará el valor 0. | no |
| encoding | El único valor permitido es "unicode" para cambiar la codificación del SMS a Unicode (ver la sección 2.5.2). En su ausencia o si tiene otro valor el SMS tomará la codificación por defecto. | no |
| concat | Si vale "true" permite concatenar mensajes para enviar un mensaje corto de longitud mayor que la habitual (ver la sección 2.5.3). En su ausencia, si tiene otro valor o si se define el parámetro "sPort" o "dPort" se deshabilita la concatenación de mensajes. | no |

Cuadro 2.1: Lista de parámetros para sendsms

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 confirmacion de entrega (ver sección 2.7):

    `OK dest:xxxxxxxxxxx idAck:wwwwwwwwww`

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. Si se hubiese enviado un mensaje concatenado (ver el parámetro "concat") a un único destinatario le corresponderán varios mensajes, tantos como fragmentos compongan el mensaje concatenado. En ese caso aparecerán líneas independientes para cada fragmento siendo cualificado el valor de "dest" con un sufijo que diferencie cada fragmento con un índice numérico comenzando por 0. Por ejemplo para un mensaje concatenado de tres fragmentos enviado al número "xxxxxxxxxxx":

```
OK dest:xxxxxxxxxxx(0) idAck:wwwwwwwwww
OK dest:xxxxxxxxxxx(1) idAck:wwwwwwwwww
OK dest:xxxxxxxxxxx(2) idAck:wwwwwwwwww
```

El valor de "errNum" se corresponde con uno de los códigos de estado del apartado 2.8.

El valor de "idAck" se corresponde con el código de identificación asociado a la solicitud de confirmación de entrega (ver sección 2.7).

La infomación 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 2.4.3).

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.

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

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

Para conocer en qué consisten los mensajes de este tipo se aconseja leer la sección 2.6.

Para saber más sobre el proceso de envío de mensajes WAP-PUSH a través de la pasarela se aconseja leer la sección 2.6.7.

Este comando tiene la lista de parámetros del cuadro 2.2.

Para enviar el **mensaje a varios destinatarios** basta repetir el parámetro "dest" tantas veces como sea preciso sin sobrepasar el límite máximo permitido (consultar al soporte técnico de *Altiria* en soporte@altiria.com), asignándole cada vez el valor de un número de teléfono distinto (los teléfonos repetidos son descartados). Se recomienda en cualquier caso no exceder de 100 destinatarios por petición.

| Nombre | Valor | Obligatorio |
|---|---|---|
| cmd | "sendwappush" | sí |
| domainId | Identificador suministrado por *Altiria* al cliente. Se puede omitir si el login es un email. | no |
| login | Identificador de usuario suministrado por *Altiria* al cliente. | sí |
| passwd | Clave del usuario suministrada por *Altiria* al cliente. | sí |

| | | |
|---|---|---|
| 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 |
| 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 2.6.1. 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 parametro "url" (ver sección 2.6.7). No puede estar vacío. | sí |
| url | Dirección de Internet desde donde el teléfono móvil se descargará el contenido. Debe contener caracteres validos en una URL (ver sección 2.6.2). 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 parametro (ver sección 2.6.7). No puede estar vacío. No es posible seleccionar un puerto de conexión diferente al 80, el habitual en la navegación WEB. | sí |
| ack | Solicitud de confirmación de entrega de los mensajes enviados (ver sección 2.7). Si vale "true" solicita confirmación de entrega de los SMS enviados. En su ausencia o si tiene otro valor no se solicita confirmación de entrega. | no |
| idAck | Código identificativo para la confirmación de entrega (ver sección 2.7). Valor alfanumérico de hasta 20 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 |

Cuadro 2.2: Lista de parámetros para sendwappush

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 2.6.7) y de solicitar confirmacion de entrega (ver sección 2.7):

    ```
    OK dest:xxxxxxxxxxx key:xxxxxxxxxxx-zzzzzzzzzz idAck:wwwwwwwwww
    ```

    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: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 “key” se corresponde con la clave única asociada al destinatario (ver sección 2.6.7).

El valor de “idAck” se corresponde con el código de identificación asociado a la solicitud de confirmación de entrega (ver sección 2.7).

El valor de “errNum” se corresponde con uno de los códigos de estado del apartado 2.8.

La infomación 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 2.4.3).

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.

### 2.4.3. Consulta del crédito disponible

Permite conocer el crédito instantáneo disponible para enviar mensajes.

Este comando tiene la lista de parámetros del cuadro 2.3.

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 de este comando.

El 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.

| **Nombre** | **Valor** | **Obligatorio** |
|---|---|---|
| cmd | ”getcredit” | sí |
| domainId | Identificador suministrado por *Altiria* al cliente. Se puede omitir si el login es un email. | no |
| login | Identificador de usuario suministrado por *Altiria* al cliente. | sí |
| passwd | Clave del usuario suministrada por *Altiria* al cliente. | sí |

Cuadro 2.3: Lista de parámetros para getcredit

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 estado del apartado 2.8.

## 2.5. Mensajes de texto

Los **caracteres permitidos** para el texto del mensaje corto y la **longitud máxima** dependerán de la codificación de caracteres seleccionada: codificación por defecto (ver sección 2.5.1) o la codificación UNICODE (ver sección 2.5.2).

### 2.5.1. Codificación por defecto

La **codificación por defecto** permite los caracteres de la tabla 2.4.

La **longitud máxima permitida** se detalla en la sección 2.5.3.

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

Adicionalmente se admiten los **caracteres extendidos** de la tabla 2.5. Cada carácter extendido **ocupa el doble espacio que un carácter normal**, esto debe considerarse para el cómputo de la longitud máxima del mensaje.

En caso de que el mensaje a enviar contenga **caracteres fuera de las listas** presentadas, estos serán **reemplazados por el carácter "?"** antes de enviar el mensaje.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| @ | ( | 4 | _ | L | W | h | s | Ú | ù |
| cr[1] | ) | 5 | A | M | X | i | t | á | |
| lf[2] | * | 6 | B | N | Y | j | u | é | |
| Ç | + | 7 | C | Ñ | Z | k | v | í | |
| sp[3] | , | 8 | D | O | ¿ | l | w | ó | |
| ! | - | 9 | E | P | a | m | x | ú | |
| ” | . | : | F | Q | b | n | y | Ü | |
| # | / | ; | 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 en la codificación por defecto

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| [ | ] | \ | ^ | { | } | \| | ~ | € |

Cuadro 2.5: Lista de caracteres extendidos permitidos para mensajes de texto

### 2.5.2. Unicode

La **codificación UNICODE**, forzada mediante el parámetro "unicode" (ver el cuadro 2.1), permite todo el juego de caracteres UNICODE de 16bits.

La **longitud máxima permitida** se detalla en la sección 2.5.3, siendo siempre menor que usando la codificación por defecto (ver la sección 2.5.1).

Con esta codificación sería posible por ejemplo el envío de vocales con tilde.

[1] Retorno de carro
[2] Nueva línea
[3] Espacio blanco

### 2.5.3. Longitud del mensaje

La longitud máxima de un mensaje de texto es un valor variable que depende de la codificación de caracteres usada y de la posibilidad de concatenación. Los mensajes que **excedan la longitud máxima aplicable serán rechazados** (no enviados).

- La longitud máxima de un mensaje corto con la **codificación por defecto es de 160 caracteres** (ver sección 2.5.1). Los caracteres extendidos (ver la tabla 2.5) ocupan el doble, por tanto la longitud máxima se reduce. Por ejemplo si el texto del SMS contuviera el símbolo del euro "€" y los corchetes "[ ]", la longitud máxima del mensaje corto se reduciría a 157 caracteres.
- La longitud máxima de un mensaje corto con la **codificación UNICODE es de 70 caracteres** (ver sección 2.5.2).

En caso de **definir puertos origen o destino** del SMS (ver los parámetros sPort y dPort en el cuadro 2.1) la **longitud máxima se reduce** de la siguiente forma:

- **152 caracteres para la codificación por defecto** (ver sección 2.5.1). Igualmente hay que considerar que los caracteres extendidos (ver la tabla 2.5) ocupan el doble.
- **66 caracteres para la codificación UNICODE** (ver sección 2.5.2).

Mediante el uso de **mensajes concatenados es posible ampliar esos límites**. Un mensaje concatenado consiste en varios mensajes en secuencia recibidos como un único mensaje en el teléfono del destinatario.

Los mensajes concatenados se posibilitan mediante el parámetro "concat" (ver el cuadro 2.1) siempre que no se esten definiendo ni el puerto origen ni el puerto destino del SMS (ver los parámetros sPort y dPort en el cuadro 2.1).

La plataforma de *Altiria* permite **concatenar hasta 10 mensajes**, aplicando en ese caso los límites siguientes a la longitud del mensaje:

- **1530 caracteres para la codificación por defecto** (ver sección 2.5.1). Igualmente hay que considerar que los caracteres extendidos (ver la tabla 2.5) ocupan el doble.
- **670 caracteres para la codificación UNICODE** (ver sección 2.5.2).

## 2.6. Mensajes 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...).

### 2.6.1. Caracteres permitidos

El cuadro 2.6 detalla los caracteres admisibles para los mensajes WAP-PUSH (parámetro “msg” del cuadro 2.2).

| cr[1] | lf[2] | sp[3] | ! | ” | # | & |
|---|---|---|---|---|---|---|
| ’ | ( | ) | * | + | , | - |
| . | / | 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 | Ú | á | é |
| í | ó | ú | | | | |

Cuadro 2.6: 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.

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

### 2.6.2. Especificación de la URL

La URL de descarga de los contenidos multimedia suministrados a través de mensajes WAP-PUSH (parámetro “url” del cuadro 2.2) debe seguir las siguientes normas de composición:

- Los caracteres del cuadro 2.7 son seguros y se pueden incluir sin codificar.

[1] Retorno de carro
[2] Nueva línea
[3] Espacio blanco

- Los caracteres del cuadro 2.8 son reservados y se pueden incluir sin codificar si se emplean dentro de la URL de acuerdo a su uso reservado. Por ejemplo el carácter "&" 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.

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 2.7: Lista de caracteres seguros

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

Cuadro 2.8: Lista de caracteres reservados

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 peticion 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.

### 2.6.3. 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 2.6, sin incluir las vocales acentuadas.

### 2.6.4. 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 240 x 240 "pixels".

### 2.6.5. 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.

### 2.6.6. 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.

### 2.6.7. Control de acceso al contenido

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:346xxxxxxx-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.

## 2.7. 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 envio 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 veinte caracteres y formado solo por caracteres válidos. Es importante constatar que el identificador devuelto en este caso solo coincidirá con el suministrado en el correpondiente 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=destination,idAck,status".

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"*.

El parametro "notification" incluye los datos del cuadro 2.9, separados por comas.

| Dato | Valor |
|---|---|
| destination | Número de teléfono móvil al que se refiere la información de estado de entrega. Si esa información es relativa a un mensaje concatenado, el teléfono figurará cualificado con un sufijo que haga referencia al fragmento particular del que se trate, por ejemplo xxxxxxxxxxx(2) (ver la respuesta al enviar un mesaje concatenado en la sección 2.4.1). |
| idAck | Identificador que coincidirá con el suministrado por *Altiria* 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. |
| status | Estado relativo a la entrega del mensaje. Podrá tomar los valores: "ENTREGADO", "NO ENTREGADO", "ERROR_100", "ERROR_101", "ERROR_114" o "ERROR_115". |

Cuadro 2.9: Lista de datos de la confirmación de entrega

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, recibiéndose 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 solvente los problemas, el mensaje será entregado, recibiéndose la correspondiente confirmación.

El estado "ERROR_114" indica que el mensaje ha sido enviado pero no ha podido ser entregado porque el número de teléfono destinatario no existe.

El estado "ERROR_115" indica que el mensaje ha sido enviado pero no ha podido ser entregado porque el destinatario no acepta mensajes.

Opcionalmente se podría configurar el envío de las notificaciones de entrega mediante la llamada a un recurso REST del cliente según la especificación relativa a la confirmación de entrega de la pasarela REST de *Altiria* para el envío de SMS.

Otra alternativa sería configurar el envío de las notificaciones de entrega mediante la llamada a un servicio web SOAP del cliente como también se detalla en la especificación relativa a la confirmación de entrega de la pasarela de *Altiria* de servicios web para el envío de SMS.

Consultar al soporte técnico de *Altiria* (soporte@altiria.com) para concer más detalles sobre estas posibilidades.

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 un envío de SMS a través de la pasarela de *Altiria* incluyendo entre otros los siguientes parámetros:
   - dest=346xxxxxxxx.
   - dest=346yyyyyyyy.
   - 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 sendas peticiones HTTP POST desde la pasarela de *Altiria* , con el parámetro "notification" conteniendo:

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

5. El servidor HTTP del cliente responde a cada 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.

## 2.8. Códigos de estado

El cuadro 2.10 presenta la lista de los posibles códigos de estado que podrán aparecer en la respuesta a cada comando.

| CÓDIGO | DETALLE |
|---|---|
| 001 | Error interno. Contactar con el soporte técnico |
| 002 | Error de acceso al puerto seguro 443. Contactar con el soporte técnico |
| 010 | Error en el formato del número de teléfono |
| 011 | Error en el envío de los parámetros de la petición 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 |
| 018 | Se ha excedido el máximo número de destinatarios autorizado |
| 019 | Se ha excedido el máximo número de mensajes autorizado |
| 020 | Error en la autentificación |
| 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 2.10: Lista de los códigos de estado

## 2.9. Ejemplos

Se presentan extractos de programación en varios lenguajes.

El **código fuente** todos los ejemplos se puede **descargar** de esta URL [CODIGOFUENTE]

*Altiria* no se responsabiliza del funcionamiento de los ejemplos presentados. Se deben considerar como fragmentos de código ilustrativos del acceso a algunas funcionalidades de la pasarela documentada.

Con objeto de **facilitar la lectura algunas líneas del código han sido partidas** con saltos de línea que podrían afectar al correcto funcionamiento del programa en su ejecución.

### 2.9.1. Envío de un mensaje en PHP

**Ejemplo usando CURL**

```
<?php
class AltiriaSMS {

public $url;
public $domainId;
public $login;
public $password;
public $debug;

public function getUrl() {
 return $this->url;
}

public function setUrl($val) {
 $this->url = $val;
 return $this;
}

public function getDomainId() {
 return $this->domain;
}

public function setDomainId($val) {
 $this->domain = $val;
 return $this;
}

public function getLogin() {
 return $this->login;
}

public function setLogin($val) {
 $this->login = $val;
 return $this;
}

public function getPassword() {
 return $this->password;
}
```

```
public function setPassword($val) {
 $this->password = $val;
 return $this;
}

public function getDebug() {
 return $this->debug;
}

public function setDebug($val) {
 $this->debug = $val;
 return $this;
}

public function sendSMS($destination, $message, $senderId=null) {

 $return=false;
 // Set the curl parameters.
 $ch = curl_init();
 curl_setopt($ch, CURLOPT_URL, $this->getUrl());
 curl_setopt($ch, CURLOPT_VERBOSE, 1);
 curl_setopt($ch, CURLOPT_HTTPHEADER,
             array('Content-type: application/x-www-form-urlencoded; charset=UTF-8'));
 curl_setopt($ch, CURLOPT_HEADER, false);
 // Max timeout in seconds to complete http request
 curl_setopt($ch, CURLOPT_TIMEOUT, 60);
 curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
 curl_setopt($ch, CURLOPT_POST, 1);

 $COMANDO='cmd=sendsms&domainId='.$this->getDomainId().'&login='.$this->getLogin();
 $COMANDO.='&passwd='.$this->getPassword().'&msg='.urlencode($message);

 //Como destinatarios se admite un array de teléfonos, una cadena de teléfonos
 //separados por comas o un único teléfono
 if (is_array($destination)){
    foreach ($destination as $telefono) {
        $this->logMsg("Add destination ".$telefono);
        $COMANDO.='&dest='.$telefono;
    }
 }
 else{
    if( strpos($destination, ',') !== false ){
        $destinationTmp= '&dest='.str_replace(',','&dest=',$destination).'&';
        $COMANDO .=$destinationTmp;
        $this->logMsg("Add destination ".$destinationTmp);
    }
    else{
        $COMANDO.='&dest='.$destination;
    }
 }

 //No es posible utilizar el remitente en América pero sí en España y Europa
 if (!isset($senderId) || empty($senderId)) {
      $this->logMsg("NO senderId ");
```

```
 }
 else{
     $COMANDO.='&senderId='.$senderId;
     $this->logMsg("Add senderId ".$senderId);
 }

 // Set the request as a POST FIELD for curl.
 curl_setopt($ch, CURLOPT_POSTFIELDS, $COMANDO);

 // Get response from the server.
 $httpResponse = curl_exec($ch);

 if(curl_getinfo($ch, CURLINFO_HTTP_CODE) === 200){
        $this->logMsg("Server Altiria response: ".$httpResponse);
        if (strstr($httpResponse,"ERROR errNum")){
                $this->logMsg("Error sending SMS: ".$httpResponse);
                return false;
        }
        else
                $return = $httpResponse;
 }
 else{
        $this->logMsg("Error sending SMS: ".curl_error($ch).'('.curl_errno($ch).')'.$httpResponse);
        $return = false;
 }

 curl_close($ch);
 return $return;
}


public static function getCredit() {
 $return=false;
 // Set the curl parameters.
 $ch = curl_init();
 curl_setopt($ch, CURLOPT_URL, $this->getUrl());
 curl_setopt($ch, CURLOPT_VERBOSE, 1);
 curl_setopt($ch, CURLOPT_HTTPHEADER,
             array('Content-type: application/x-www-form-urlencoded; charset=UTF-8'));
 curl_setopt($ch, CURLOPT_HEADER, false);
 // Max timeout in seconds to complete http request
 curl_setopt($ch, CURLOPT_TIMEOUT, 60);
 curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
 curl_setopt($ch, CURLOPT_POST, 1);

 $COMANDO='cmd=getcredit&domainId='.$this->getDomainId().'&login='.$this->getLogin();
 $COMANDO.='&passwd='.$this->getPassword();

 // Set the request as a POST FIELD for curl.
 curl_setopt($ch, CURLOPT_POSTFIELDS, $COMANDO);

 // Get response from the server.
 $httpResponse = curl_exec($ch);

 if(curl_getinfo($ch, CURLINFO_HTTP_CODE) === 200){
```

```
    $this->logMsg("Server Altiria response: ".$httpResponse);
    if (strstr($httpResponse,"ERROR errNum")){
       $this->logMsg("Error asking SMS credit: ".$httpResponse);
       $return = false;
    }
    else
       $return = $httpResponse;
 }
 else{
    $this->logMsg("Error getting credit: ".curl_error($ch).'('.curl_errno($ch).')'.$httpResponse);
    $return = false;
 }

 curl_close($ch);
 return $return;
}

public function logMsg($msg) {
 if ($this->getDebug()===true)
     error_log("\n".date(DATE_RFC2822)." : ".$msg."\r\n", 3, "app.log");
}

}
?>

<?php
// XX, YY y ZZ se corresponden con los valores de identificacion del
// usuario en el sistema.
include('httpPHPAltiria.php');

$altiriaSMS = new AltiriaSMS();
$altiriaSMS->setUrl("http://www.altiria.net/api/http");
$altiriaSMS->setDomainId('XX');
$altiriaSMS->setLogin('YY');
$altiriaSMS->setPassword('ZZ');

$altiriaSMS->setDebug(true);

//$sDestination = '346xxxxxxxx';
$sDestination = '346xxxxxxxx,346yyyyyyyy';
//$sDestination = array('346xxxxxxxx','346yyyyyyyy');

//No es posible utilizar el remitente en América pero sí en España y Europa
$response = $altiriaSMS->sendSMS($sDestination, "Mensaje de prueba");
//Utilizar esta llamada solo si se cuenta con un remitente autorizado por Altiria
//$response = $altiriaSMS->sendSMS($sDestination, "Mensaje de prueba", "remitente");

if (!$response)
  echo "El envío ha terminado en error";
else
  echo $response;
?>
```

**Ejemplo abiendo directamente un socket**

```
<?php
// sDestination: lista de numeros de telefono separados por comas.
// Cada numero debe comenzar por el prefijo internacional de pais.
// sMessage: hasta 160 caracteres
// debug: Si es true muestra por pantalla la respuesta completa del servidor
// sSenderId: no es posible utilizar el remitente en América pero sí en España y Europa
// XX, YY y ZZ se corresponden con los valores de identificacion del
// usuario en el sistema.
function AltiriaSMS($sDestination, $sMessage, $debug, $sSenderId){

 $sData ='cmd=sendsms&';
 $sData .='domainId=XX&';
 $sData .='login=YY&';
 $sData .='passwd=ZZ&';
 //No es posible utilizar el remitente en América pero sí en España y Europa
 $sData .='senderId='.$sSenderId.'&';
 $sData .='dest='.str_replace(',','&dest=',$sDestination).'&';
 $sData .='msg='.urlencode(utf8_encode(substr($sMessage,0,160)));

 //Tiempo máximo de espera para conectar con el servidor = 5 seg
 $timeOut = 5;
 $fp = fsockopen('www.altiria.net', 80, $errno, $errstr, $timeOut);
 if (!$fp) {
  //Error de conexion o tiempo maximo de conexion rebasado
  $output = "ERROR de conexion: $errno - $errstr<br />\n";
  $output .= "Compruebe que ha configurado correctamente la direccion/url ";
  $output .= "suministrada por altiria<br>";
  return $output;
 } else {
  $buf = "POST /api/http 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 = "";

  //Tiempo máximo de espera de respuesta del servidor = 60 seg
  $responseTimeOut = 60;
  stream_set_timeout($fp,$responseTimeOut);
  stream_set_blocking ($fp, true);
  if (!feof($fp)){
   if (($buf=fgets($fp,128))===false){
    // TimeOut?
    $info = stream_get_meta_data($fp);
    if ($info['timed_out']){
     $output = 'ERROR Tiempo de respuesta agotado';
     return $output;
    } else {
     $output = 'ERROR de respuesta';
     return $output;
    }
```

```
   } else{
    while(!feof($fp)){
     $buf.=fgets($fp,128);
    }
   }
  } else {
   $output = 'ERROR de respuesta';
   return $output;
  }
  fclose($fp);
  //Si la llamada se hace con debug, se muestra la respuesta completa del servidor
  if ($debug){
   print "Respuesta del servidor: <br>".$buf."<br>";
  }
  //Se comprueba que se ha conectado realmente con el servidor
  //y que se obtenga un codigo HTTP OK 200
  if (strpos($buf,"HTTP/1.1 200 OK") === false){
   $output = "ERROR. Codigo error HTTP: ".substr($buf,9,3)."<br />\n";
   $output .= "Compruebe que ha configurado correctamente la direccion/url ";
   $output .= "suministrada por Altiria<br>";
   return $output;
  }
  //Se comprueba la respuesta de Altiria
  if (strstr($buf,"ERROR")){
   $output = $buf."<br />\n";
   $output .= " Ha ocurrido algun error. Compruebe la especificacion<br>";
   return $output;
  } else {
   $output = $buf."<br />\n";
   $output .= " Exito<br>";
   return $output;
  }
 }
}

//No es posible utilizar el remitente en América pero sí en España y Europa
$resp= AltiriaSMS("346xxxxxxxx,346yyyyyyyy", "Mensaje de prueba", false, "");
//Utilizar esta llamada solo si se cuenta con un remitente autorizado por Altiria
//$resp= AltiriaSMS("346xxxxxxxx,346yyyyyyyy", "Mensaje de prueba", false, "remitente");
echo $resp;

?>
```

### 2.9.2. Envío de un mensaje en JAVA

Ejemplo en Java utilizando Httpclient 4.5 como cliente HTTP (ver [HTTPCLIENT]).

Esta librería depende de los siguientes paquetes:

- commons-logging versión 1.2
- commons-codec versión 1.9
- httpcore versión 4.4.3

```
//Se fija el tiempo máximo de espera para conectar con el servidor (5000)
```

```
//Se fija el tiempo máximo de espera de la respuesta del servidor (60000)
RequestConfig config = RequestConfig.custom()
 .setConnectTimeout(5000)
 .setSocketTimeout(60000)
 .build();

//Se inicia el objeto HTTP
HttpClientBuilder builder = HttpClientBuilder.create();
builder.setDefaultRequestConfig(config);
CloseableHttpClient httpClient = builder.build();

//Se fija la URL sobre la que enviar la petición POST
HttpPost post = new HttpPost("http://www.altiria.net/api/http");

//Se crea la lista de parámetros a enviar en la petición POST
List<NameValuePair> parametersList = new ArrayList <NameValuePair>();
//XX, YY y ZZ se corresponden con los valores de identificación del
//usuario en el sistema.
parametersList.add(new BasicNameValuePair("cmd", "sendsms"));
parametersList.add(new BasicNameValuePair("domainId", "XX"));
parametersList.add(new BasicNameValuePair("login", "YY"));
parametersList.add(new BasicNameValuePair("passwd", "ZZ"));
parametersList.add(new BasicNameValuePair("dest", "346xxxxxxxx"));
parametersList.add(new BasicNameValuePair("dest", "346yyyyyyyy"));
parametersList.add(new BasicNameValuePair("msg", "Mesaje de prueba"));
//No es posible utilizar el remitente en América pero sí en España y Europa
//Descomentar la línea solo si se cuenta con un remitente autorizado por Altiria
//parametersList.add(new BasicNameValuePair("senderId", "remitente"));

try {
 //Se fija la codificacion de caracteres de la peticion POST
 post.setEntity(new UrlEncodedFormEntity(parametersList,"UTF-8"));
}
catch(UnsupportedEncodingException uex) {
 System.out.println("ERROR: codificación de caracteres no soportada");
}

CloseableHttpResponse response = null;

try {
 System.out.println("Enviando petición");
 //Se envía la petición
 response = httpClient.execute(post);
 //Se consigue la respuesta
 String resp = EntityUtils.toString(response.getEntity());

 //Error en la respuesta del servidor
 if (response.getStatusLine().getStatusCode()!=200){
   System.out.println("ERROR: Código de error HTTP:  " + response.getStatusLine().getStatusCode());
   System.out.println("Compruebe que ha configurado correctamente la direccion/url ");
   System.out.println("suministrada por Altiria");
   return;
 }else {
   //Se procesa la respuesta capturada en la cadena 'response'
   if (resp.startsWith("ERROR")){
```

```
    System.out.println(resp);
    System.out.println("Código de error de Altiria. Compruebe las especificaciones");
   } else
    System.out.println(resp);
   }
 }
 catch (Exception e) {
   System.out.println("Excepción");
   e.printStackTrace();
   return;
 }
 finally {
    //En cualquier caso se cierra la conexión
    post.releaseConnection();
    if(response!=null) {
     try {
       response.close();
     }
     catch(IOException ioe) {
       System.out.println("ERROR cerrando recursos");
     }
    }
 }
```

### 2.9.3. Envío de un mensaje en VBA

Ejemplo en Visual Basic para aplicaciones usando ServerXMLHTTP como cliente HTTP

```
'Se fija la URL sobre la que enviar la petición POST
Set objHTTP = CreateObject("MSXML2.ServerXMLHTTP")
objHTTP.Open "POST", "http://www.altiria.net/api/http", False
objHTTP.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.
objHTTP.Send ("cmd=sendsms&domainId=XX&login=YY&passwd=ZZ&dest=346xxxxxxxx&msg=Mensaje de prueba")
'No es posible utilizar el remitente en América pero sí en España y Europa
'Utilizar esta llamada solo si se cuenta con un remitente autorizado por Altiria
'objHTTP.Send ("cmd=sendsms&domainId=XX&login=YY&passwd=ZZ
'               &dest=346xxxxxxxx&msg=Mensaje de prueba&senderId=remitente")
MsgBox (objHTTP.Status)
MsgBox (objHTTP.ResponseText)
```

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

Todos los ejemplos de .NET se han desarrollado con Visual Studio 2015

**Ejemplo en Visual C# usando HttpWebRequest 4.0 como cliente HTTP**

```
//Se fija la URL sobre la que enviar la petición POST
HttpWebRequest loHttp =
  (HttpWebRequest)WebRequest.Create("http://www.altiria.net/api/http");

// 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=346xxxxxxxx&dest=346yyyyyyyy" +
  "&msg=Mensaje de prueba";
  //No es posible utilizar el remitente en América pero sí en España y Europa
  //Descomentar la línea solo si se cuenta con un remitente autorizado por Altiria
  //cmd=cmd + "&senderId=remitente";

// Se 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;

//Fijamos tiempo de espera de respuesta = 60 seg
loHttp.Timeout = 60000;
String error = "";
String response = "";
// Envía la peticion
try {
  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 vendría 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
  response = loResponseStream.ReadToEnd();
  loWebResponse.Close();
  loResponseStream.Close();
}
catch (WebException e) {
  if (e.Status == WebExceptionStatus.ConnectFailure)
     error = "Error en la conexión";
  else if (e.Status == WebExceptionStatus.Timeout)
     error = "Error TimeOut";
  else
     error = e.Message;
}
finally {
  if (error != "")
     Console.WriteLine(error);
  else
     Console.WriteLine(response);
}
```

**Ejemplo en Visual C# usando HttpClient 4.5 como cliente HTTP**

```
HttpClient client = new HttpClient();
client.BaseAddress = new Uri("http://www.altiria.net");
//Establecemos el TimeOut para obtener la respuesta del servidor
client.Timeout = TimeSpan.FromSeconds(60);
```

```
//Se compone el mensaje a enviar
// XX, YY y ZZ se corresponden con los valores de identificación del usuario en el sistema.
var postData = new List<KeyValuePair<string, string>>();
postData.Add(new KeyValuePair<string, string>("cmd", "sendsms"));
postData.Add(new KeyValuePair<string, string>("domainId", "XX"));
postData.Add(new KeyValuePair<string, string>("login", "YY"));
postData.Add(new KeyValuePair<string, string>("passwd", "ZZ"));
postData.Add(new KeyValuePair<string, string>("dest", "346xxxxxxxx"));
postData.Add(new KeyValuePair<string, string>("dest", "346yyyyyyyy"));
postData.Add(new KeyValuePair<string, string>("msg", "Mensaje de prueba"));
//No es posible utilizar el remitente en América pero sí en España y Europa
//Descomentar la línea solo si se cuenta con un remitente autorizado por Altiria
//postData.Add(new KeyValuePair<string, string>("senderId", "remitente"));

HttpContent content = new FormUrlEncodedContent(postData);
String err = "";
String resp="";
try {
   //Se fija la URL sobre la que enviar la petición POST
   HttpRequestMessage request = new HttpRequestMessage(HttpMethod.Post, "/api/http");
   request.Content = content;
   content.Headers.ContentType.CharSet = "UTF-8";
   request.Content.Headers.ContentType =
     new MediaTypeHeaderValue("application/x-www-form-urlencoded");
   HttpResponseMessage response = client.SendAsync(request).Result;
   var responseString = response.Content.ReadAsStringAsync();
   resp = responseString.Result;
}
catch (Exception e) {
   err = e.Message;
}
finally {
   if (err != "")
     Console.WriteLine(err);
   else
     Console.WriteLine(resp);
}
```

**Ejemplo en Visual Basic usando HttpWebRequest 4.0 como cliente HTTP**

```
'Se fija la URL sobre la que enviar la petición POST
Dim loHttp As HttpWebRequest
loHttp =
  CType(WebRequest.Create("http://www.altiria.net/api/http"), 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=346xxxxxxxx&dest=346yyyyyyyy" +
  "&msg=Texto de prueba"
' No es posible utilizar el remitente en América pero sí en España y Europa
' Descomentar la línea solo si se cuenta con un remitente autorizado por Altiria
' lcPostData += "&senderId=remitente"
```

```
'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

'Fijamos TimeOut de espera de respuesta del servidor = 60 seg
loHttp.Timeout = 60000
Dim err As String = ""
Dim response As String = ""

'Envía la peticion
Try
  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 vendría 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
  response = loResponseStream.ReadToEnd()
  loWebResponse.Close()
  loResponseStream.Close()

Catch e As WebException
  If (e.Status = WebExceptionStatus.ConnectFailure) Then
     err = "Error en la conexión"
  ElseIf (e.Status = WebExceptionStatus.Timeout) Then
     err = "Error Time Out"
  Else
     err = e.Message
  End If

Finally
  If (err <> "") Then
     Console.WriteLine(err)
  Else
     Console.WriteLine(response)
  End If
End Try
```

**Ejemplo en Visual Basic usando HttpClient 4.5 como cliente HTTP**

```
Dim client As HttpClient = New HttpClient
client.BaseAddress = New Uri("Http://www.altiria.net")
'Fijamos TimeOut de espera de respuesta del servidor = 60 seg
client.Timeout = TimeSpan.FromSeconds(60)
```

```
'Se compone el mensaje a enviar
'XX, YY y ZZ se corresponden con los valores de identificación del usuario en el sistema.
Dim postData As List(Of KeyValuePair(Of String, String))
postData = New List(Of KeyValuePair(Of String, String))
postData.Add(New KeyValuePair(Of String, String)("cmd", "sendsms"))
postData.Add(New KeyValuePair(Of String, String)("domainId", "XX"))
postData.Add(New KeyValuePair(Of String, String)("login", "YY"))
postData.Add(New KeyValuePair(Of String, String)("passwd", "ZZ"))
postData.Add(New KeyValuePair(Of String, String)("dest", "346xxxxxxxx"))
postData.Add(New KeyValuePair(Of String, String)("dest", "346yyyyyyyy"))
postData.Add(New KeyValuePair(Of String, String)("msg", "Mensaje de prueba"))
' No es posible utilizar el remitente en América pero sí en España y Europa
' Descomentar la línea solo si se cuenta con un remitente autorizado por Altiria
' postData.Add(New KeyValuePair(Of String, String)("senderId", "remitente"))

Dim content As HttpContent = New FormUrlEncodedContent(postData)
Dim err = ""
Dim resp = ""
Try
  'Se fija la URL sobre la que enviar la petición POST
  Dim request As HttpRequestMessage
  request = New HttpRequestMessage(HttpMethod.Post, "/api/http")
  request.Content = content
  content.Headers.ContentType.CharSet = "UTF-8"
  Dim contentType = "application/x-www-form-urlencoded"
  request.Content.Headers.ContentType = New Headers.MediaTypeHeaderValue(contentType)
  Dim response As HttpResponseMessage = client.SendAsync(request).Result

  Dim responseString = response.Content.ReadAsStringAsync
  resp = responseString.Result

Catch e1 As Exception
   err = e1.Message
Finally
   If (err <> "") Then
      Console.WriteLine(err)
   Else
      Console.WriteLine(resp)
   End If
End Try
```

### 2.9.5. Envío de un mensaje en Delphi

Se ha programado Delphi en RAD Studio 10.

#### Ejemplo en usando IdHTTP como cliente HTTP

```
 try
   //Se fija la URL sobre la que enviar la petición POST
   SUrl:='http://www.altiria.net/api/http';
   //Compone 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=346xxxxxxxx');
   Parametros.Add('dest=346yyyyyyyy');
   Parametros.Add(UTF8Encode('msg=Mensaje de prueba'));
   //No es posible utilizar el remitente en América pero sí en España y Europa
   //Descomentar la línea solo si se cuenta con un remitente autorizado por Altiria
   //Parametros.Add('senderId=remitente');

   IdHTTP1:= TIdHTTP.Create(nil);

   //Se fija el tiempo máximo de espera para conectar con el servidor (5000)
   //Se fija el tiempo máximo de espera de la respuesta del servidor (60000)
   IdHTTP1.ConnectTimeout := 5000;
   IdHTTP1.ReadTimeout:=60000;

   IdHTTP1.Request.ContentType :='application/x-www-form-urlencoded';
   IdHTTP1.Request.Charset := 'UTF-8';
   IdHTTP1.Request.ContentEncoding := 'UTF-8';

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

 except
   on E: Exception do
     if E.ClassName='EIdConnectTimeout' then
       WriteLn ('ERROR Connect Timeout')
     else if E.ClassName='EIdReadTimeout' then
       WriteLn ('ERROR Response Timeout')
     else
       Writeln(E.ClassName, ': ', E.Message);
 end;
 Parametros.Free;
 IdHTTP1.Free;
end.
```

**Ejemplo en usando TnetHTTPClient como cliente HTTP**

```
begin
 try
   client := TNetHTTPClient.Create(nil);
   client.ContentType :='application/x-www-form-urlencoded';
   //Se fija la URL sobre la que enviar la petición POST
   SUrl:='http://www.altiria.net/api/http';
   //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=346xxxxxxxx');
   Parametros.Add('dest=346yyyyyyyy');
   Parametros.Add(UTF8Encode('msg=Mensaje de prueba'));
```

```
   //No es posible utilizar el remitente en América pero sí en España y Europa
   //Descomentar la línea solo si se cuenta con un remitente autorizado por Altiria
   //Parametros.Add('senderId=remitente');

   Response:=client.Post(SUrl,Parametros);

   if Response.StatusCode = 200 then
     WriteLn(Response.ContentAsString())
   else
     begin
       WriteLn('Código de error: ' + IntToStr(Response.StatusCode));
       WriteLn(Response.StatusText);
     end;
 except
   on E: ENetHTTPClientException do
     WriteLn ('ERROR Conexión.  ' + E.Message);

   on E: Exception do
     Writeln(E.ClassName, ': ', E.Message);
 end;
 Parametros.Free;
 Client.Free;
end.
```

### 2.9.6. Envío de un mensaje en Perl

Ejemplo en strawberryPerl 5.22.0.1 para Windows (ver [PERL]) usando LWP::UserAgent como cliente HTTP.

```
#!/usr/bin/perl
use strict;
use warnings;
use LWP::UserAgent;
use utf8;
use Encode qw(decode encode);

my $ua = new LWP::UserAgent();
# Timeout en segundos
$ua->timeout(60);
# Se fija la URL sobre la que enviar la petición POST
my $req = new HTTP::Request POST => "http://www.altiria.net/api/http";
$req->header('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
my $data =("cmd=sendsms&domainId=XX&login=YY&passwd=ZZ&dest=346xxxxxxx&dest=346yyyyyyyy".
   "&msg=Mensaje de prueba".
   #No es posible utilizar el remitente en América pero sí en España y Europa
   #Descomentar la línea solo si se cuenta con un remitente autorizado por Altiria
   #"&senderId=remitente".
           "");

$data = encode('UTF8',$data);

$req->content($data);
```

```
my $resp = $ua->request($req);
if ($resp->is_success) {
# $resp->code = 200
  my $message = $resp->decoded_content;
  print "\nRespuesta: \n$message\n";
}else {
  print "HTTP POST error code: ", $resp->code, "\n";
  print $resp->decoded_content;
}
```

### 2.9.7. Envío de un mensaje en Ruby

Ejemplo en Ruby de cliente HTTP.

```
# encoding: UTF-8

require 'net/http'
require 'json'
require 'uri'

def altiriaSms(destinations, message, senderId, debug)
  if debug
    puts "Enter altiriaSms: destinations=#{destinations}, message=#{message}, senderId=#{senderId}"
  end

  begin

    #Se fija la URL sobre la que enviar la petición POST
    url = "http://www.altiria.net/api/http"
    uri = URI.parse(url)
    http = Net::HTTP.new(uri.host, uri.port)
    #Se fija el tiempo máximo de espera para conectar con el servidor (5 segundos)
    #Se fija el tiempo máximo de espera de la respuesta del servidor (60 segundos)
    http.open_timeout = 5
    http.read_timeout = 60

    #Se crea la lista de parámetros a enviar en la petición POST
    #XX, YY y ZZ se corresponden con los valores de identificación del usuario en el sistema.
    #No es posible utilizar el remitente en América pero sí en España y Europa
    post_data = {'cmd' => 'sendsms', 'domainId' => 'XX', 'login' => 'YY', 'passwd' => 'ZZ',
    'dest' => destinations.split(","), 'msg' => message, 'senderId' => senderId}

    #Se envía la petición y se consigue la respuesta
    #La codificación es de tipo "application/x-www-form-urlencoded;charset=utf-8"
    #fijado por el método "post_form"
    response = Net::HTTP.post_form( uri, post_data)

    if debug
      unless response.code == "200" #Error en la respuesta del servidor
        puts("ERROR GENERAL: #{response.code}")
        puts("#{response.body}")
      else #Se procesa la respuesta
        puts("Código de estado HTTP: #{response.code}")
```

```
        if "#{response.body}".include? "ERROR errNum:"
          puts("Error de Altiria: #{response.body}")
        else
           puts("Cuerpo de la respuesta: \n#{response.body}")
        end
      end
    end

    return response

  rescue Net::OpenTimeout
    puts "Tiempo de conexión agotado"
  rescue Net::ReadTimeout
    puts "Tiempo de respuesta agotado"
  rescue Exception => e
    puts "Error interno: #{e.class}"
  end

end

puts "The function altiriaSms returns:"+
     "#{altiriaSms('346xxxxxxxx,346yyyyyyyy','Mensaje de prueba', '', true).body}"
#No es posible utilizar el remitente en América pero sí en España y Europa
#Utilizar esta llamada solo si se cuenta con un remitente autorizado por Altiria
#puts "The function altiriaSms returns:"+
#"#{altiriaSms('346xxxxxxxx,346yyyyyyyy','Mensaje de prueba', 'remitente', true).body}"
```

### 2.9.8. Envío de un mensaje en Python

Ejemplo en Python utilizando la librería Requests como cliente HTTP (ver [REQUESTS]).

```
# -*- coding: utf-8 -*-

import requests

def altiriaSms(destinations, message, senderId, debug):
  if debug:
    print 'Enter altiriaSms: '+destinations+', message: '+message+', senderId: '+senderId

  try:
    #Se crea la lista de parámetros a enviar en la petición POST
    #XX, YY y ZZ se corresponden con los valores de identificación del usuario en el sistema.
    payload = [
      ('cmd', 'sendsms'),
      ('domainId', 'XX'),
      ('login', 'YY'),
      ('passwd', 'ZZ'),
      #No es posible utilizar el remitente en América pero sí en España y Europa
      ('senderId', senderId),
      ('msg', message)
    ]

    #add destinations
    for destination in destinations.split(","):
```

```
      payload.append(('dest', destination))

    #Se fija la codificacion de caracteres de la peticion POST
    contentType = {'Content-Type':'application/x-www-form-urlencoded;charset=utf-8'}

    #Se fija la URL sobre la que enviar la petición POST
    url = 'http://www.altiria.net/api/http'

    #Se envía la petición y se recupera la respuesta
    r = requests.post(url,
    data=payload,
    headers=contentType,
    #Se fija el tiempo máximo de espera para conectar con el servidor (5 segundos)
    #Se fija el tiempo máximo de espera de la respuesta del servidor (60 segundos)
    timeout=(5, 60)) #timeout(timeout_connect, timeout_read)

    if debug:
      if str(r.status_code) != '200': #Error en la respuesta del servidor
        print 'ERROR GENERAL: '+str(r.status_code)
      else: #Se procesa la respuesta
        print 'Código de estado HTTP: '+str(r.status_code)
        if (r.text).find("ERROR errNum:"):
          print 'Error de Altiria: '+r.text
        else:
          print 'Cuerpo de la respuesta: \n'+r.text

    return r.text

  except  requests.ConnectTimeout:
    print "Tiempo de conexión agotado"

  except  requests.ReadTimeout:
    print "Tiempo de respuesta agotado"

  except Exception as ex:
    print "Error interno: "+str(ex)

print 'The function altiriaSms returns: \n'
     +altiriaSms('346xxxxxxxx,346yyyyyyyy','Mensaje de prueba', '', True)
#No es posible utilizar el remitente en América pero sí en España y Europa
#Utilizar esta llamada solo si se cuenta con un remitente autorizado por Altiria
#print 'The function altiriaSms returns: \n'
#     +altiriaSms('346xxxxxxxx,346yyyyyyyy','Mensaje de prueba', 'remitente', True)
```

### 2.9.9. Envío de un mensaje en node.js

Ejemplo en node.js.

```
var querystring = require('querystring');
var http = require('http');

function sendSMS(domainId, login, passwd, tel, sender, text) {
  // Se contruye la cadena del post desde un objeto
  var post_data = querystring.stringify({
```

```
      'cmd' : 'sendsms',
      'domainId' : domainId,
      'login': login,
      'passwd': passwd,
      'dest' : tel,
      //No es posible utilizar el remitente en América pero sí en España y Europa
      'senderId' : sender,
      'msg' : text
  });

  // Un objeto de opciones sobre donde se envia el post
  var post_options = {
      host: 'www.altiria.net',
      port: '80',
      path: '/api/http',
      method: 'POST',
      headers: {
          'Content-Type': 'application/x-www-form-urlencoded',
          'Content-Length': Buffer.byteLength(post_data)
      }
  };

  // Se efectua la petición
  var post_req = http.request(post_options, function(res) {
      res.setEncoding('utf8');
      res.on('data', function (chunk) {
          //Es necesario procesar la respuesta y los posibles errores
          console.log('Response: ' + chunk);
      });
  });

  // post the data
  post_req.write(post_data);
  post_req.end();

}

// XX, YY y ZZ se corresponden con los valores de identificacion del
// usuario en el sistema.
sendSMS('XX','YY','ZZ','346xxxxxxxx,346yyyyyyyy','','Mensaje de prueba');
//No es posible utilizar el remitente en América pero sí en España y Europa
//Utilizar esta llamada solo si se cuenta con un remitente autorizado por Altiria
//sendSMS('XX','YY','ZZ','346xxxxxxxx,346yyyyyyyy','remitente','Mensaje de prueba');
```

# Referencias

[FAQ] *Preguntas frecuentes de la pasarela de envío de SMS de Altiria:*
http://www.altiria.com/preguntas-frecuentes-faq-pasarela-sms-api

[HTTPCLIENT] *El proyecto HTTPCLIENT de Apache:*
http://hc.apache.org/httpcomponents-client-ga/

[NUSOAP] *Librería NuSOAP en Sourceforge:*
http://sourceforge.net/projects/nusoap/

[WSIMPORT] *Herramienta para generación de WS en Java:*
http://docs.oracle.com/javase/8/docs/technotes/tools/unix/wsimport.html

[PERL] *Página oficial de descargas de perl:*
https://www.perl.org/get.html

[CODIGOFUENTE] *Enlace a la descarga del código fuente de los ejemplos:*
https://static.altiria.com/especificaciones/altiria-push-ejemplos-codigo-fuente.zip

[REQUESTS] *Página oficial de Requests:*
http://docs.python-requests.org

[SAVON] *Página oficial de Savon:*
http://savonrb.com

[SUDS] *API de Suds:*
https://pypi.python.org/pypi/suds

[SOAPLite] *API de SOAP::Lite:*
http://search.cpan.org/ phred/SOAP-Lite-1.20/lib/SOAP/Lite.pm