

Especificaciones de la Interfaz HTTP para envío de
SMS, landings web y firma de documentos

Especificaciones de la Interfaz HTTP para envío de SMS, landings web y firma de documentos

ABARCANDO, S.L.

Versión: 2.5

Copyright © ABARCANDO 2021

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
2.5	<p>Se añade el parámetro opcional “multiSig” al comando “certPdf-File” para definir más de un firmante por documento así como la posición de la firma (2.9.1).</p> <p>Se añade la sección 2.9.2 para detallar todos los aspectos relativos a la ubicación de las firmas en el documento.</p> <p>Se añaden los códigos de error 012, 013, 030, 031 y 032 (sección 2.9.5).</p>
2.4	<p>Se añaden los parámetros opcionales “title” y “forwardEmail” al comando “certpdffile” para la firma de documentos (2.9.1).</p> <p>Se actualiza la información devuelta en el comando “checkpdffile” (2.9.4).</p> <p>Se actualizan los ejemplos de programación (sección 2.11).</p>
2.3	<p>Se añaden dos nuevas secciones sobre la firma de documentos (2.9) y la actualización de su estado (2.10).</p> <p>Se actualizan los ejemplos de programación (sección 2.11).</p>
2.2	<p>Se añade el parámetro opcional “certDelivery” al comando “sendsms” para solicitar la certificación de la entrega del SMS enviado mediante un documento digital con sello certificado de tiempo (sección 2.4.1).</p> <p>Se añade una nueva notificación de estado “CERT(url)” para recibir la URL de acceso al documento del certificado de entrega (cuadro 2.11).</p>
2.1	<p>Se añade una nueva sección sobre los enlaces a web móvil (2.7) y una subsección sobre la parametrización de webs móviles (2.7.1).</p> <p>Se actualiza la información del parámetro “msg” y se añaden los parámetros “landing”, “params” y “webParams” en el cuadro 2.1, todo en relación a los enlaces a web móvil (sección 2.7).</p> <p>Se añaden los códigos de error 035, 036, 037, 038 y 039 (sección 2.4.4).</p>
2.0	<p>Se añaden los códigos de error 018 y 019 (sección 2.4.4).</p> <p>Se admite la autenticación sin el parámetro “domainId” en caso de que el login sea un email.</p> <p>Se actualizan los ejemplos de programación (sección 2.11).</p>
1.16	<p>Se añade dos nuevos estados de confirmación de entrega: “ERROR_114” y “ERROR_115” (cuadro 2.11).</p> <p>Se actualizan los ejemplos de programación (sección 2.11).</p>
1.15	<p>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).</p>

Historial de cambios

Versión	Cambios
1.14	<p>Se reorganiza el documento para agrupar la información de forma más clara.</p> <p>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).</p> <p>Se añade el código de error 002 (sección 2.4.4).</p> <p>Se precisa la información sobre el parámetro “senderId” para permitir el envío de remitentes numéricos (sección 2.4.1).</p> <p>Se añade la posibilidad de enviar mensajes concatenados mediante el parámetro “concat” (sección 2.4.1).</p> <p>Se añade la posibilidad de enviar mensajes codificados en UNI-CODE mediante el parámetro “encoding” (sección 2.4.1).</p> <p>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).</p> <p>Se actualizan las referencias bibliográficas del final del documento.</p>
1.13	<p>Se añade una referencia a la sección de preguntas frecuentes de nuestro portal web (sección 1).</p>
1.12	<p>Se incluye un nuevo ejemplo en ASP 3.0 y se corrige una errata en el ejemplo en PHP (sección 2.11).</p> <p>Se detalla el modo de codificar los caracteres si los parámetros de la petición se envían en la URL (sección 2.2).</p> <p>Se precisa la información relativa a la especificación de una URL en mensajes WAP-PUSH (sección 2.6.2).</p> <p>Se añaden dos parámetros opcionales al comando “sendsms” para permitir especificar el puerto origen y destino del SMS a enviar (sección 2.4.1).</p> <p>Se añaden nuevos códigos de error correspondientes a los nuevos parámetros: 033 y 034 (sección 2.4.4).</p>
1.11	<p>Se incluye el enlace a la página que detalla la lista de países permitidos y las restricciones geográficas aplicables (sección 1).</p> <p>Se añaden las vocales con acento grave (à) a la lista de caracteres permitidos para los mensajes SMS de texto (sección 2.5.1).</p> <p>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 2.5.1) y 2.6.1).</p> <p>Se actualiza la información relativa a la consulta de crédito disponible, previa el envío de mensajes (sección 2.4.3).</p> <p>Se incluye un capítulo con enlaces a las tarifas y a la información de cobertura internacional.</p>
1.10	<p>Se actualiza la lista de caracteres permitidos para mensajes de texto, desaconsejando el de apertura de exclamación (cuadro 2.5).</p> <p>Se detalla la respuesta a la petición HTTP POST con datos de confirmación de entrega (sección 2.8).</p>

Índice general

1. Introducción	4
2. Descripción de la API	5
2.1. Envío de la petición	5
2.2. Codificación de caracteres	5
2.3. Respuesta a la petición	6
2.4. Comandos de la API de envío de SMS	6
2.4.1. Envío de un mensaje de texto	6
2.4.2. Envío de un mensaje multimedia WAP-PUSH	9
2.4.3. Consulta del crédito disponible	10
2.4.4. Códigos de estado	12
2.5. Mensajes de texto	13
2.5.1. Codificación por defecto	13
2.5.2. Unicode	13
2.5.3. Longitud del mensaje	14
2.6. Mensajes WAP-PUSH	15
2.6.1. Caracteres permitidos	15
2.6.2. Especificación de la URL	15
2.6.3. Tipos de contenido	16
2.6.4. Formato del contenido	17
2.6.5. Dirección del contenido	17
2.6.6. Envío del contenido	17
2.6.7. Control de acceso al contenido	17
2.7. Enlaces a web móvil	19
2.7.1. Parametrización de la web móvil	20
2.8. Confirmación de entrega	22
2.9. Comandos de la API de firma de contrato online	25
2.9.1. Firma de un documento PDF	25
2.9.2. Posicionamiento de las firmas	29
2.9.3. Envío del documento PDF	30
2.9.4. Consulta del estado de firma de un documento PDF	30

2.9.5. Códigos de estado	31
2.10. Actualización del estado de firma de documentos	32
2.11. Ejemplos	35
2.11.1. Envío de un mensaje en PHP	35
2.11.2. Envío y firma de documento PDF en PHP	43
2.11.3. Envío de un mensaje en JAVA	50
2.11.4. Envío y firma de documento PDF en JAVA	51
2.11.5. Envío de un mensaje en VBA	54
2.11.6. Envío de un mensaje en .NET	54
2.11.7. Envío de un mensaje en Delphi	58
2.11.8. Envío de un mensaje en Perl	60
2.11.9. Envío y firma de documento PDF en Perl	61
2.11.10. Envío de un mensaje en Ruby	63
2.11.11. Envío y firma de documento PDF en Ruby	64
2.11.12. Envío de un mensaje en Python	66
2.11.13. Envío y firma de documento PDF en Python	67
2.11.14. Envío de un mensaje en node.js	69
2.11.15. Envío y firma de documento PDF en node.js	70

Capítulo 1

Introducción

En este documento se presenta la API disponible para el envío de mensajes cortos, landings web y firma de documentos sobre la interfaz de *Abarcando* 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. 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) se puede enviar un correo electrónico a comercial@abarcando.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 2.8).

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 enviar las peticiones HTTP es **http://web.smsverificados.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://web.smsverificados.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 2.11 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 Abarcando-SMSverificados, 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 de envío de SMS

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 *Abarcando* en soporte@abarcando.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 <i>Abarcando</i> al cliente. Se puede omitir si el login es un email.	no
login	Identificador de usuario suministrado por <i>Abarcando</i> al cliente.	sí
passwd	Clave del usuario suministrada por <i>Abarcando</i> 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). Se pueden añadir identificadores de web móvil para generar enlaces acortados únicos en el texto del mensaje. Consultar la sección 2.7 para conocer más detalles sobre las webs móviles.	sí
senderId	Remitente del mensaje a enviar, autorizado por <i>Abarcando</i> . 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 Abarcando-SMSverificados.	no
landing	Número con la suma de identificadores de web móvil (si se repiten también suman) que se utilizan en el texto del mensaje (parámetro “msg”) para citar una web móvil (su uso es obligatorio en este caso). Consultar la sección 2.7 para conocer más detalles sobre las webs móviles.	no
params	Número de webs móviles (de las citadas en el texto del mensaje mediante el parámetro “msg”) para las que se remiten valores parametrizados. Su uso es obligatorio en caso de definir el parámetro “landing” (se debe fijar a 0 si no hay webs parametrizadas). En caso de remitir valores parametrizados debe incluirse también parámetros tipo “webParams” . Consultar la sección 2.7.1 para conocer más detalles sobre la parametrización de las webs móviles.	no
webParams	Se debe añadir para cada web móvil parametrizada el parámetro “webParams” cualificado con el identificador de la web móvil. Define los parámetros particulares de cada web. Su uso es obligatorio en caso de que “params” sea mayor que 0. Consultar la sección 2.7.1 para conocer más detalles sobre la parametrización de las webs móviles.	no
ack	Solicitud de confirmación de entrega de los mensajes enviados (ver sección 2.8). Si vale “true” solicita confirmación de entrega de los SMS enviados. Si adicionalmente se solicita certificar la entrega de los SMS enviados (ver parámetro “certDelivery”), se remitirá la URL de acceso al documento del certificado. En su ausencia o si tiene otro valor no se solicita ni la confirmación de entrega ni la URL del documento del certificado.	no
idAck	Código identificativo para la confirmación de entrega (ver sección 2.8). 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
certDelivery	Si vale “true” solicita certificar la entrega del SMS enviado mediante un documento digital con sello certificado de tiempo. Este servicio tiene sobrecoste y solo está disponible en Europa. Si tiene dudas contacte con comercial@abarcando.com. En su ausencia o si tiene otro valor no se solicita el certificado. Si adicionalmente se solicita la confirmación de entrega del SMS enviado (ver parámetro “ack”), se podrá recibir la URL de acceso al documento del certificado.	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 2.8):

```
OK dest:xxxxxxxxxx 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:xxxxxxxxxx errNum:yy
```

- 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 “xxxxxxxxxx”:

```
OK dest:xxxxxxxxxx(0) idAck:wwwwwwwww
OK dest:xxxxxxxxxx(1) idAck:wwwwwwwww
OK dest:xxxxxxxxxx(2) idAck:wwwwwwwww
```

El valor de “errNum” se corresponde con uno de los códigos de estado del apartado 2.4.4.

El valor de “idAck” se corresponde con el código de identificación asociado a la solicitud de confirmación de entrega (ver sección 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 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 través 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 *Abarcando* en soporte@abarcando.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 <i>Abarcando</i> al cliente. Se puede omitir si el login es un email.	no
login	Identificador de usuario suministrado por <i>Abarcando</i> al cliente.	sí
passwd	Clave del usuario suministrada por <i>Abarcando</i> 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 parámetro “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 parámetro (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 2.8). 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 2.8). 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 confirmación de entrega (ver sección 2.8):


```
OK dest:xxxxxxxxxx key:xxxxxxxxxx-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:xxxxxxxxxx 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 2.8).

El valor de “errNum” se corresponde con uno de los códigos de estado del apartado 2.4.4.

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 2.4.3).

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 *Abarcando* para reintentar los envíos pendientes.

Nombre	Valor	Obligatorio
cmd	"getcredit"	sí
domainId	Identificador suministrado por <i>Abarcando</i> al cliente. Se puede omitir si el login es un email.	no
login	Identificador de usuario suministrado por <i>Abarcando</i> al cliente.	sí
passwd	Clave del usuario suministrada por <i>Abarcando</i> 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 2.4.4.

2.4.4. Códigos de estado

El cuadro 2.4 presenta la lista de los posibles códigos de estado que podrán aparecer en la respuesta a cada comando del API de envío de SMS (ver 2.4).

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 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
035	La web móvil enlazada en el mensaje no pertenece al usuario
036	La web móvil enlazada en el mensaje no existe
037	Se ha excedido el máximo número de webs móviles enlazadas en el mensaje
038	Error de sintaxis en la definición del envío con enlace a web móvil
039	Error de sintaxis en la definición de los parámetros de la web móvil parametrizada

Cuadro 2.4: Lista de los códigos de estado

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 2.5.

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 2.6. 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	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 2.5: Lista de caracteres permitidos para mensajes de texto en la codificación por defecto

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

Cuadro 2.6: 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.

¹Retorno de carro

²Nueva línea

³Espacio 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 2.6) 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 2.6) 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 estén 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 *Abarcando* 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 2.6) ocupan el doble.
- **670 caracteres para la codificación UNICODE** (ver sección 2.5.2).

2.6. Mensajes WAP-PUSH

La interfaz HTTP de *Abarcando* 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 2.7 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 2.7: 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 2.8 son seguros y se pueden incluir sin codificar.

¹Retorno de carro

²Nueva línea

³Espacio blanco

- Los caracteres del cuadro 2.9 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 parámetros 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 2.8: Lista de caracteres seguros

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

Cuadro 2.9: 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 símbolo “%” 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.

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 2.7, 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 *Abarcando* 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=”. *Abarcando* agregará en ese caso a la URL enviada en cada mensaje la clave única con el formato: “xxxxxxxxxx-zzzzzzzzzz”; ‘xxxxxxxxxx’ 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 parámetros “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 *Abarcando* 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 *Abarcando* 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 *Abarcando* 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.

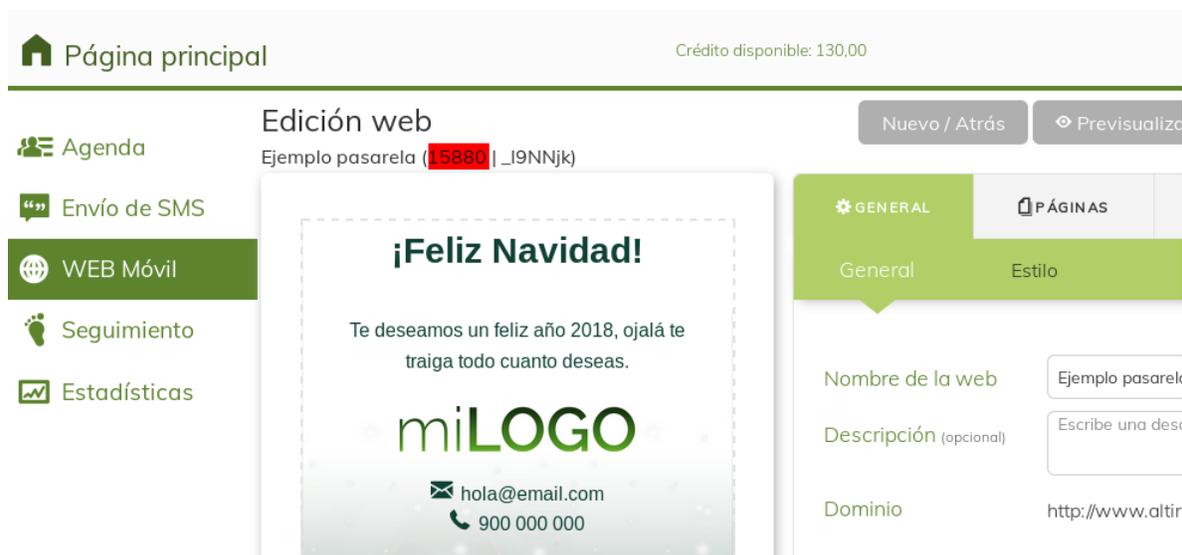
2.7. Enlaces a web móvil

El portal web de envío de SMS SMSVERIFICADOS de Abarcando ([SMSVERIFICADOS2021]) permite también componer páginas webs adaptadas para móviles ([LANDINGS]) de forma sencilla a partir de plantillas. Si desea más detalles sobre esta u otras funciones del portal web, envíe un correo electrónico solicitando información a comercial@abarcando.com.

Las webs creadas en el portal SMSVERIFICADOS de Abarcando se pueden enlazar fácilmente en el texto de los SMS enviados mediante la pasarela (API). Para ello se debe añadir el identificador de la web en el texto del SMS a enviar (parámetro “msg” del cuadro 2.1) con el siguiente formato:

{\$identificador\$}

Se puede obtener el valor del identificador de cada web en el portal SMSVERIFICADOS de Abarcando, en la columna de edición de la web, junto a su nombre.



Se ilustra a continuación los parámetros necesarios para la web móvil de la figura anterior con identificador 15880:

```
cmd="sendsms"
domainId="XXXXX"
login="YYYYY"
passwd="ZZZZZ"
dest="346XXXXXXXX"
msg="Estimado cliente, acceda a la promoción en el siguiente enlace: {$15880$}"
landing="1"
params="0"
```

En el momento del envío del SMS a cada destinatario, el identificador de la web será reemplazado por un enlace a un acortador de URLs.

Cada **enlace** consta de un **dominio acortado** y un **código hash alfanumérico único e irrepetible de 7 caracteres**.

El dominio del acortador por defecto es “<http://go2.re>” aunque se puede solicitar uno distinto (remitir la petición a comercial@abarcando.com).

Cuando el destinatario recibe el SMS, podría leer por ejemplo el siguiente mensaje:

Estimado cliente, acceda a la promoción en el siguiente enlace: <http://go2.re/wD34f1q>

Al tratarse de una URL única por destinatario, el portal SMSVERIFICADOS de Abarcando guarda estadísticas precisas sobre qué destinatarios han visitado la web o rellenado un formulario dentro de la misma, pudiendo trazar todo el proceso: **envío SMS => entrega SMS => web visitada => formulario completado**

Se deben satisfacer entonces los siguientes **requisitos** para añadir con éxito enlaces a webs móviles:

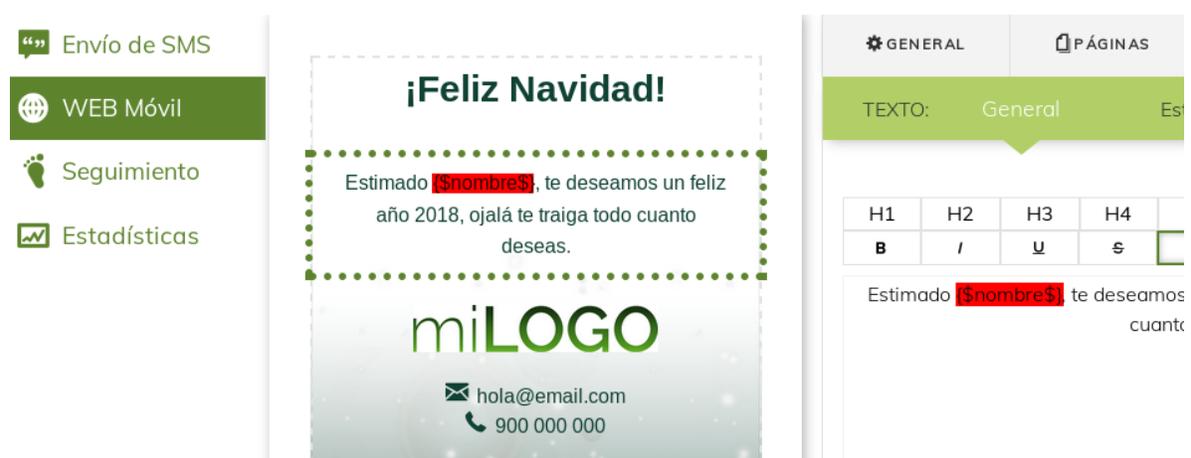
- Cada identificador de web contenido en el parámetro “msg” debe ir precedido y seguido por al menos un espacio, salto de línea, inicio de mensaje o fin de mensaje. De esta forma se garantiza que el terminal móvil que reciba el SMS reconocerá correctamente el enlace contenido y permitirá la navegación por la web.
- Se permite añadir hasta 10 identificadores distintos de web móvil en cada mensaje.
- Añadir el parámetro “landing” (ver el cuadro 2.1) que es la suma total de identificadores de web contenidos en el texto del SMS. Si alguna web se utiliza más de una vez, debe sumarse el identificador correspondiente.
- Añadir el parámetro “params” (ver el cuadro 2.1) indicando cuántas webs están parametrizadas (ver el apartado siguiente 2.7.1). Se debe fijar a 0 si no hay webs parametrizadas.

2.7.1. Parametrización de la web móvil

Las webs enlazadas en el texto de los SMS enviados se pueden parametrizar de forma que cada destinatario acceda a una versión propia, con unos datos particulares.

En primer lugar desde el editor de web móvil en el portal SMSVERIFICADOS de Abarcando se deben añadir los parámetros oportunos al contenido de la web.

Se ilustra a continuación el parámetro “nombre” añadido a la siguiente web móvil:



Se observa que el editor del portal SMSVERIFICADOS de Abarcando añade los parámetros al contenido de la web con el siguiente formato;

{ \$parámetro\$ }

El enlace a la web móvil debe añadirse al SMS tal y como se ha detallado en el apartado anterior (2.7). Además hay que satisfacer los siguientes **requisitos** adicionales:

- Definir el parámetro “params” (ver el cuadro 2.1) para indicar cuántas de las webs móviles enlazadas en el texto del SMS van a ser parametrizadas.
- Añadir para cada web móvil parametrizada el parámetro “webParams” cualificado con el identificador de la web móvil. Este parámetro define en **formato json** los pares **parámetro/valor** propios de la web. Si para algún parámetro se define un valor vacío, así será representado al visualizar la web móvil.

Se resalta que los caracteres del cuadro 2.10 deben ser representados de manera especial para evitar errores de procesado del json en los parámetros “webParams”:

Carácter	Representación
”	\\”
\\	\\\\

Cuadro 2.10: Caracteres especiales json

Para el ejemplo de la web móvil de la figura anterior con identificador 15880 serían estos parámetros:

```
cmd="sendsms"
domainId="XXXXX"
login="YYYYY"
passwd="ZZZZZ"
dest="346XXXXXXXX"
msg="Estimado cliente, acceda a la promoción en el siguiente enlace: {$15880$}"
landing="1"
params="1"
webParams15880="{\"nombre\": \"Pedro\"}"
```

Como resultado, el receptor del SMS con el enlace a la web móvil parametrizada verá el siguiente contenido:



Cada parámetro “webParams” puede contener tantos pares parámetro/valor como se hayan definido a través del editor web. Basta añadirlos correctamente formateados en json de esta forma:

```
{ "parámetro1": "valor1" , "parámetro2": "valor2", "parámetro3": "valor3" ... }
```

En caso de no remitir valores para todos los parámetros definidos a través del editor web, se usará el valor por defecto propio de cada parámetro, también definido a través del editor web.

2.8. 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.

Si adicionalmente se solicitó certificar la entrega del SMS a través del parámetro “certDelivery”, permite recibir la URL de acceso al documento del certificado.

Para tener acceso a este servicio es preciso que el cliente haya notificado a *Abarcando* 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 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 correspondiente comando de envío si cumple los criterios de composición explicados en las tablas 2.1y 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 parámetro “notification” incluye los datos del cuadro 2.11, 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 mensaje concatenado en la sección 2.4.1).
idAck	Identificador que coincidirá con el suministrado por <i>Abarcando</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.
status	Estado relativo a la entrega del mensaje. Podrá tomar los valores: “ENTREGADO”, “NO ENTREGADO”, “ERROR_100”, “ERROR_101”, “ERROR_114”, “ERROR_115” o “CERT(url)”.

Cuadro 2.11: 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, 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 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.

El estado “CERT(url)” referencia la url de acceso al documento del certificado de entrega del SMS (parámetro “certDelivery”). El certificado será global y único para todos los fragmentos que pudieran componer el SMS (en caso de ser concatenado).

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 *Abarcando* 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 *Abarcando* de servicios web para el envío de SMS.

Consultar al soporte técnico de *Abarcando* (soporte@abarcando.com) para conocer 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 *Abarcando* incluyendo entre otros los siguientes parámetros:
 - dest=346xxxxxxxx.
 - dest=346yyyyyyyy.
 - ack=true.
 - idAck=zxxx.

2. La pasarela de *Abarcando* 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:zxxx
OK dest:346yyyyyyyy idAck:zxxx
```

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

3. Tan solo uno de los destinatarios recibe 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 *Abarcando* , con el parámetro “notification” conteniendo:

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

```
‘‘notification=346yyyyyyy,zzzz,NO 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.
7. Si adicionalmente se solicitó certificar la entrega de los SMS a través del parámetro “certDelivery”, se recibirá también la URL de acceso al documento del certificado para el SMS entregado con éxito:

```
‘‘notification=346xxxxxxxx,zzzz,CERT(http://...)’’
```

2.9. Comandos de la API de firma de contrato online

A continuación se detallan los comandos disponibles en el API para firma de documentos. 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.

2.9.1. Firma de un documento PDF

Permite iniciar el proceso de firma online sobre un destinatario.

El **destinatario** se debe identificar con un **número de teléfono y/o con una dirección de correo electrónico**; al menos se debe suministrar alguno de los datos. La notificación de documento pendiente de firma se le remitirá por ambas vías si están disponibles.

Para **identificar a la empresa que emite la solicitud de firma de documento** es preciso configurar un nombre comercial y una dirección de correo electrónico. Contactar con comercial@abarcando.com para solicitar la configuración de esos datos.

Es también **obligatorio** seleccionar al menos un **mecanismo de firma** del documento entre los parámetros **smsOtpSig**, **emailOtpSig**, **webSig**, **ecertSig** y **manSig**. Se admite elegir más de un mecanismo de firma por documento.

El comando en sí sólo define los parámetros del servicio de firma sin adjuntar el documento PDF. A partir de la respuesta se obtendrá una URL sobre la que remitir el documento en un POST HTTP independiente.

Este comando tiene la lista de parámetros del cuadro 2.12.

Nombre	Valor	Obligatorio
cmd	certpdffile	sí
domainId	Identificador suministrado por <i>Abarcando</i> al cliente. Se puede omitir si el login es un email.	no*
login	Identificador de usuario suministrado por <i>Abarcando</i> al cliente.	sí
passwd	Clave del usuario suministrada por <i>Abarcando</i> al cliente.	sí
multiSig	Lista de firmantes del documento hasta un máximo de 15. Cada elemento de la lista se compondrá de los siguientes subelementos: destination : similar al parámetro “destination” descrito más adelante; email : similar al parámetro “email” descrito más adelante; sequence : valor opcional entero entre 1 y el máximo de firmantes (el cero o valores negativos serán omitidos). Define el número de orden de cada firmante. Los firmantes sin número de orden recibirán la petición de firma en primer lugar de forma simultánea. Los firmantes con número de orden recibirán la petición de firma cuando los firmantes ordenados previamente hayan firmado el documento. Se puede repetir valor entre distintos firmantes, se consideran firmantes con similar prioridad; sigLocX , sigLocY , sigLocWidth , sigLocHeight y sigLocPage : parámetros de posicionamiento de cada firma (consultar la sección 2.9.2).	sí*
destination	Número de teléfono móvil del destinatario del documento. 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. Su uso es obligatorio si no se incluye el parámetro “email” o si se escoge el método de firma por OTP SMS mediante el parámetro “smsOtpSig”.	sí*

email	Dirección de correo electrónico del destinatario del documento. Su uso es obligatorio si no se incluye el parámetro “destination” o si se escoge el método de firma por OTP email mediante el parámetro “emailOtpSig”.	sí*
type	Tipo de firma de contrato online solicitada. Solo puede tomar dos valores: “premium” o “simple”. Consultar las características y los costes de cada tipo de servicio en comercial@abarcando.com.	sí
smsOtpSig	SMS en el Web de firma. Sólo si vale “true” se escoge como mecanismo de firma el envío de un código OTP por SMS. Será entonces obligatorio definir el teléfono móvil de cada firmante mediante el parámetro “destination” . Es obligatorio escoger al menos un mecanismo de firma entre “smsOtpSig”, “emailOtpSig”, “webSig”, “ecertSig” y “manSig”.	no*
emailOtpSig	Correo-E en el Web de firma. Sólo si vale “true” se escoge como mecanismo de firma el envío de un código OTP por email. Será entonces obligatorio definir la dirección de correo electrónico de cada firmante mediante el parámetro “email” . Es obligatorio escoger al menos un mecanismo de firma entre las siguientes posibilidades: “smsOtpSig”, “emailOtpSig”, “webSig”, “ecertSig” y “manSig”.	no*
webSig	Click-Wrap en el Web de firma. Sólo si vale “true” se escoge como mecanismo de firma marcar varios “checkbox” en la página de firma del documento. Es obligatorio escoger al menos un mecanismo de firma entre las siguientes posibilidades: “smsOtpSig”, “emailOtpSig”, “webSig”, “ecertSig” y “manSig”.	no*
ecertSig	Firma-E en el Web de firma. Sólo si vale “true” se escoge como mecanismo de firma el uso del certificado electrónico digital del destinatario. Está opción no aparecerá disponible si el destinatario accede a la firma del documento desde un teléfono móvil. Será necesario además que la primera vez instale un componente de firma específico en su navegador. Es obligatorio escoger al menos un mecanismo de firma entre “smsOtpSig”, “emailOtpSig”, “webSig”, “ecertSig” y “manSig”.	no*
manSig	Firma-D en el Web de firma. Sólo si vale “true” se escoge la firma manuscrita como mecanismo. Si el destinatario accede a la firma del documento desde un teléfono móvil tendrá que usar el dedo para firmar. En otro caso se usará el puntero del ratón o el “touchpad”. Es obligatorio escoger al menos un mecanismo de firma entre “smsOtpSig”, “emailOtpSig”, “webSig”, “ecertSig” y “manSig”.	no*
senderId	Remitente a utilizar en los SMS enviados al destinatario del documento, autorizado por <i>Abarcando</i> . Solo tendrá uso si se define el teléfono móvil del destinatario mediante el parámetro “destination” . La posibilidad de personalizar el remitente depende del país destinatario del mensaje. Se admite un 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”). Los caracteres inválidos serán suprimidos automáticamente . Si no se incluye, el mensaje se enviará con el remitente por defecto seleccionado por Abarcando-SMSverificados.	no
smsText	Texto personalizado del SMS remitido al destinatario con la notificación de documento pendiente de firma. Como máximo 120 caracteres de la sección 2.5.1. El texto completo enviado en el SMS contendrá al final la URL de acceso al documento. Si no se define este parámetro el destinatario recibirá un texto por defecto.	no
emailText	Párrafo adicional añadido al texto del correo electrónico remitido al destinatario con la notificación de documento pendiente de firma. Como máximo 1023 caracteres. El salto de línea se debe marcar como “ ”.	no

title	Título del envío de documento para identificarlo en el portal web SMSVERIFICADOS de Abarcando ([SMSVERIFICADOS2021]). Como máximo 50 caracteres del alfabeto ISO8859-1. No se incluye en el documento que se firma.	no
forwardEmail	Dirección de correo electrónico a la que se enviará una copia del documento firmado.	no
refreshDays	Número entero entre 1 y 7 representando cada cuantos días se reenviará al destinatario, a modo de recordatorio, el correo electrónico con la notificación de documento pendiente de firma. Se usa en conjunción con el parámetro “refreshTimes” . Sólo se tendrá en cuenta si se solicita servicio “premium” (ver el parámetro “type”) y se define la dirección de correo del destinatario mediante el parámetro “email” ; en otro caso será omitido.	no
refreshTimes	Número entero entre 1 y 10 representando el número máximo de recordatorios que se reenviarán al destinatario. Sólo se tendrá en cuenta si se define el parámetro “refreshDays” , se solicita servicio “premium” (ver el parámetro “type”) y se define la dirección de correo del destinatario mediante el parámetro “email” ; en otro caso será omitido. Por defecto toma el valor 1.	no
requestDni	Sólo si vale “true” y se solicita servicio “premium” (ver el parámetro “type”) se requerirá la inclusión obligatoria de las dos caras del DNI del destinatario durante el proceso de firma, mediante archivos de imagen “jpg”. Si el destinatario accede a la firma del documento desde un teléfono móvil, además de poder elegir una imagen almacenada, podrá utilizar la cámara del dispositivo.	no
customClause	Claúsula personalizada que aparecerá como requisito previo en las condiciones de firma del documento. Cadena de texto formada por tres elementos separados por “;” con el siguiente formato: título;índice;texto . El título es un identificador textual de la cláusula (máximo 50 caracteres del alfabeto ISO8859-1). No debe contener el carácter “;”. El índice es un identificador numérico para ordenar la presentación de la cláusula respecto de las demás que puedan aparecer (entero positivo que no debe repetirse). El texto es el contenido en sí de la cláusula (máximo 10.000 caracteres del alfabeto ISO8859-1; el salto de línea se debe marcar como “ ”; se admiten etiquetas HTML para formatear el texto de la cláusula). Si se agrega el sufijo * al título de la cláusula, obliga a que el destinatario la acepte para poder firmar el documento. Se puede incluir más de una cláusula repitiendo este parámetro tantas veces como sea necesario.	no
callback	Si vale “true” solicita la recepción de las actualizaciones del estado de firma del documento enviado a través de un callback (ver sección 2.10).	no

Cuadro 2.12: Lista de parámetros para certpdffile

Los **firmantes del documento se pueden definir de dos formas**:

- Mediante los parámetros ”destination/email” sólo si hay un único firmante.
- Repitiendo el parámetro ”multiSig” para definir desde un firmante hasta 15. De esta forma se permite además determinar el orden de cada firmante y la posición de cada firma en el documento (consultar la sección 2.9.2).

A continuación se ilustra un ejemplo de cada caso:

Un ejemplo de parámetros del comando certPdfFile usando ”destination/email”:

- Envío de la notificación de firma del documento a un único destinatario a través de email y SMS.
- Servicio de firma “premium” por OTP desde SMS.
- Solicitud de recepción de la actualización de estado mediante callback (ver sección 2.10).

sería este:

```
domainId="XXXXX"  
login="YYYYY"  
passwd="ZZZZZ"  
destination="346XXXXXXXX"  
email="usuario@dominio.com"  
type="premium"  
smsOtpSig="true"  
callback="true"
```

Un ejemplo de parámetros del comando `certPdfFile` usando “multiSig”:

- Envío de la notificación de firma del documento a dos destinatarios a través de email y SMS.
- Definición del orden para cada destinatario.
- Servicio de firma “premium” por OTP desde SMS.
- Solicitud de recepción de la actualización de estado mediante callback (ver sección 2.10).

sería este:

```
domainId="XXXXX"  
login="YYYYY"  
passwd="ZZZZZ"  
multiSig="{\"destination\":\"346XXXXXXXX\",\"email\":\"usuario1@dominio.com\",\"sequence\":1}"  
multiSig="{\"destination\":\"346YYYYYYYYY\",\"email\":\"usuario2@dominio.com\",\"sequence\":2}"  
type="premium"  
smsOtpSig="true"  
callback="true"
```

Se observa que el **parámetro “multiSig” se repite por cada firmante**. El valor del parámetro se **formatea en json** de la forma:

```
{ "parámetro1":"valor1" , "parámetro2":"valor2", "parámetro3":"valor3" ... }
```

La **respuesta a este comando** será un mensaje json con el elemento “status” representando un código de estado general de los descritos en apartado 2.9.5

Si la operación ha resultado exitosa (código de estado “000”) la respuesta contendrá adicionalmente:

- **url**: URL sobre la que subir el documento PDF a remitir al destinatario.
- **id**: identificador alfanumérico de 33 caracteres de longitud correspondiente al documento PDF pendiente de remitir. Ese identificador deberá ser utilizado posteriormente en el resto de comandos para referenciar a ese documento en particular.

Un ejemplo de respuesta exitosa sería el mostrado a continuación:

```
{
"status":"000",
"url":"http://...",
"id":"XXXXXXXX_YYYYYYYYYYYYYYYY_ZZZZZZZZ"
}
```

Un ejemplo de respuesta notificando un error en la autenticación sería este:

```
{
"status":"020"
}
```

2.9.2. Posicionamiento de las firmas

El parámetro "multiSig" (ver el cuadro 2.12) permite definir la posición de las firmas en el documento.

Si no se suministra esa información (es opcional), **las firmas se posicionan de acuerdo a las siguientes reglas:**

- Si el documento no está firmado previamente de forma electrónica:
 - Hasta dos firmantes: las firmas se ubican al pie de la última página con fondo transparente para evitar tapar el texto en la medida de lo posible.
 - Más de dos firmantes: se incluye una página final adicional en el documento; las firmas se distribuyen en filas de tres en esa página con un fondo no transparente.
- Si el documento ya está firmado previamente de forma electrónica:
 - Hasta dos firmantes: las firmas se ubican al pie de la última página con fondo transparente para evitar tapar el texto en la medida de lo posible.
 - Más de dos firmantes: las firmas se distribuyen en la última página como aparecerían en la página adicional del caso anterior. Sería recomendable que la última página del documento estuviese vacía para que las firmas no tapen el texto.

Los **elementos de posicionamiento explícito de la firma** son: sigLocX, sigLocY, sigLocWidth, sigLocHeight y sigLocPage.

En el momento en que se defina la posición de la firma para algún destinatario se ha de especificar también para todos los demás. Por otra parte se debe **evitar solapar distintas firmas** porque se producirán errores durante el procesado posterior del documento, anulándose el envío.

A continuación se describen los 5 parámetros relativos a la posición de la firma y la **lógica aplicada a todos ellos en el momento en que alguno esté definido.**

En todos los casos los valores asignados están establecidos en "puntos"(pt). Las coordenadas 0,0 de la página se considera que empiezan en la esquina inferior izquierda de la hoja.

La correspondencia entre los puntos y la dimensión en mm de las hojas del documento se puede encontrar en esta URL [POINTS2MM].

- **sigLocX:** valor entero que se corresponde con el posicionamiento en el eje horizontal del cuadro de firma, tomando como referencia el borde izquierdo de la página. Si no es recibido, es menor que cero o la firma saldría fuera del documento (cálculo a partir del ancho de la página y del ancho de la firma)), la firma se situará en el borde derecho de la página. En caso contrario, se aplicará el valor especificado por el usuario.

- **sigLocY**: valor entero que se corresponde con el posicionamiento en el eje vertical del cuadro de firma, tomando como referencia el borde inferior de la página. Si no es recibido o es menor que cero, la firma se ajustará al borde inferior de la página. En caso de que la firma se saliese fuera del documento (cálculo a partir del alto de la página y de la altura de la firma), la firma se situará en el borde superior de la página. Para el resto de casos se aplicará el valor especificado por el usuario.
- **sigLocWidth**: valor entero que se corresponde con el tamaño para el ancho de la firma. Si no es recibido o es menor que el mínimo admisible (140pt) se aplicará el valor mínimo descrito. En caso de que el valor supere el máximo admisible (280pt) se aplicará el valor máximo descrito. Para el resto de casos se aplicará el valor especificado por el usuario. Es importante definirlo en consonancia con el tamaño de la altura de la firma para que esta guarde su relación de aspecto. Lo ideal es mantener una relación de aspecto de 2:1.
- **sigLocHeight**: valor entero que se corresponde con el tamaño para la altura de la firma. Si no es recibido o es menor que el mínimo admisible (70pt) se aplicará el valor mínimo descrito. En caso de que el valor supere el máximo admisible (140 pt) se aplicará el valor máximo descrito. Para el resto de casos se aplicará el valor especificado por el usuario. Es importante definirlo en consonancia con el tamaño de la anchura de la firma para que esta guarde su relación de aspecto. Lo ideal es mantener una relación de aspecto de 2:1.
- **sigLocPage**: valor entero que se corresponde con la página en que aparecerá la firma del destinatario. Si no es recibido o es menor que uno la firma se aplicará en la primera página. En caso de que el valor supere el número total de páginas del documento, la firma se aplicará en la última página.

2.9.3. Envío del documento PDF

Como se ha detallado en el comando de firma de documento (ver 2.9.1), el documento PDF se debe subir efectuando una petición HTTP POST a la URL suministrada en la respuesta.

El cuerpo de la petición HTTP contendrá el código binario del documento PDF, sin ninguna transformación. El PDF suministrado no debe estar protegido contra escritura ni por contraseña.

Se debe añadir a la petición la cabecera “content-type:application/pdf”.

La **respuesta a esta petición** será un mensaje json con el elemento “status” representando un código de estado general de los descritos en apartado 2.9.5

Por ejemplo en caso de éxito:

```
{
"status": "000"
}
```

En determinados casos excepcionales, ante errores genéricos producidos en el servidor, se recibirá como respuesta un mensaje json con el elemento “message” representado la causa del error y el elemento “status_code” asociado al código de error HTTP producido en el servidor. Un ejemplo sería el siguiente:

```
{"message": "Not Found", "status_code": 404}
```

2.9.4. Consulta del estado de firma de un documento PDF

Permite consultar el estado del proceso de firma certificada asociado a un documento remitido previamente.

Este comando tiene la lista de parámetros del cuadro 2.13.

Nombre	Valor	Obligatorio
cmd	checkpdffile	sí
domainId	Identificador suministrado por <i>Abarcando</i> al cliente. Se puede omitir si el login es un email.	no*
login	Identificador de usuario suministrado por <i>Abarcando</i> al cliente.	sí
passwd	Clave del usuario suministrada por <i>Abarcando</i> al cliente.	sí
id	Identificador alfanumérico del documento remitido con éxito previamente.	sí

Cuadro 2.13: Lista de parámetros para checkpdffile

La **respuesta a este comando** será un mensaje json con el **elemento “status”** representando un código de estado general de los descritos en apartado 2.9.5

Si la operación ha resultado exitosa (código de estado “000”) la respuesta contendrá adicionalmente:

- **fileStatus**: estado global del proceso de firma del documento enviado. Ver la sección 2.10 para consultar detalles sobre los posibles estados.
- **files**: lista de ficheros descargables asociados al proceso de firma. Ver la sección 2.10 para consultar detalles sobre los ficheros descargables. Además de los tipos detallados en esa sección figurará el tipo “all” para acceder a un archivo comprimido con todos los ficheros disponibles. Cada elemento de la lista se compone a su vez de los siguientes subelementos:
 - **fileType**: tipo del fichero descargable.
 - **fileUrl**: url de descarga del fichero

Un ejemplo de respuesta exitosa sería el mostrado a continuación:

```
{
  "status": "000",
  "fileStatus": "signed",
  "files":
  [{"fileType": "source", "fileUrl": "https://..."},
   {"fileType": "signed", "fileUrl": "https://..."},
   {"fileType": "all", "fileUrl": "https://..."}]
}
```

Un ejemplo de respuesta notificando un error en la autenticación sería este:

```
{
  "status": "020"
}
```

2.9.5. Códigos de estado

El cuadro 2.14 presenta la lista de los posibles códigos de estado que podrán aparecer en la respuesta a cada comando del API de firma de documentos (ver 2.9).

CÓDIGO	DETALLE
000	Éxito
001	Error interno. Contactar con el soporte técnico
002	Error de acceso al puerto seguro 443. Contactar con el soporte técnico
003	No hay saldo suficiente para completar la operación
004	Servicio no disponible. Consultar con comercial@abarcando.com
011	Error en el envío de los parámetros
012	Error en la definición del posicionamiento de alguna firma
013	Se ha superado el máximo número de firmantes del documento
020	Error en la autenticación
022	El remitente seleccionado para el envío no es válido
028	El identificador del documento no es válido o no existe
029	El documento no es un pdf válido
030	El documento está protegido con contraseña
031	No se admite un documento ya firmado electrónicamente en un contrato de tipo simple
032	El documento está protegido frente a escritura
040	Exceso de documentos pendientes de envío. Esperar unos minutos y reintentar

Cuadro 2.14: Lista de los códigos de estado

2.10. Actualización del estado de firma de documentos

El servicio de recepción de la actualización del estado de firma del documento enviado, solicitado a través del parámetro “callback” del comando de firma de documento PDF (ver 2.9.1), permite recibir notificaciones con cada nuevo estado por el que pasa el proceso de firma.

Para tener acceso a este servicio es preciso que el cliente haya notificado a *Abarcando* la dirección de Internet a donde se enviarán las notificaciones. En caso contrario, las solicitudes de actualización de estado serán ignoradas aunque los documentos sí serán enviados para iniciar el proceso de firma.

Para que el cliente pueda asociar la información recibida sobre el estado de firma con un documento enviado previamente a través de la pasarela, se usará el identificador devuelto en la respuesta del comando de firma de documento PDF (ver 2.9.1) en el elemento “id”.

Las **notificaciones del estado de firma** 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, “pdfNotification”, cuyo valor consta de un mensaje JSON con los elementos del cuadro 2.15.

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

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

A continuación se esquematiza un ejemplo del parámetro “pdfNotification” recibido:

```
{
  "id": "XXXXXXXX_YYYYYYYYYYYYYYY_ZZZZZZZ",
  "fileStatus": "processing",
  "fileType": "sentEmail",
  "fileUrl": "http://..."
}
```

En cada **notificación** se informa del **estado global del proceso de firma** y se remite la **url de acceso a un documento** PDF asociado a dicho proceso.

Los posibles estados para **fileStatus** son:

- **pending**: el proceso de firma del documento aún no se ha iniciado.

Dato	Valor
id	Identificador alfanumérico del documento PDF remitido previamente. Emitido como respuesta del comando de firma de documento PDF (ver 2.9.1).
fileStatus	Estado global del proceso de firma del documento enviado. Ver a continuación los posibles estados.
fileType	Tipo del fichero descargable. Ver a continuación los posibles tipos de ficheros.
fileUrl	Url de descarga del fichero.

Cuadro 2.15: Lista de datos de la notificación del estado de firma

- **error**: el proceso de firma del documento no ha podido ser iniciado y se ha cancelado.
- **processing**: el proceso de firma del documento se ha iniciado.
- **signed**: el destinatario ha firmado el documento remitido.

Los posibles estados para **fileType** son:

- **source**: documento pdf original remitido al destinatario para su firma.
- **sentEmail**: evidencia electrónica del envío al correo electrónico del destinatario de la notificación de documento pendiente de firma.
- **sentSms**: evidencia electrónica del envío al teléfono móvil del destinatario de un SMS con la notificación de documento pendiente de firma.
- **accessedFile**: evidencia electrónica de la visualización del documento pendiente de firma por parte del destinatario.
- **sentSmsOtp**: evidencia electrónica del envío al teléfono móvil del destinatario de un SMS con la clave para la firma OTP del documento.
- **sentEmailOtp**: evidencia electrónica del envío al correo electrónico del destinatario de la clave para la firma OTP del documento.
- **signedFile**: evidencia electrónica constatando la firma del documento por parte del destinatario. Contendrá las posibles imágenes del DNI del destinatario (ver el parámetro “requestDni” en la sección 2.9.1) y las posibles cláusulas adicionales remitidas (ver el parámetro “customClause” en la sección 2.9.1).
- **signed**: documento pdf ya firmado por el usuario.
- **record**: acta final que recoge las evidencias de todo el proceso.

En caso de solicitar la firma de tipo “simple” (ver el parámetro “type” en la sección 2.9.1), tan solo se podrán emitir los estados “source” y “signed”, no quedando constancia de ninguna otra evidencia ni fichero relacionado.

Opcionalmente se podría configurar el envío de las notificaciones mediante la llamada a un recurso REST del cliente según la especificación relativa a la actualización del estado de firma de la pasarela REST de *Abarcando* para la firma de documentos.

Otra alternativa sería configurar el envío de las notificaciones mediante la llamada a un servicio web SOAP del cliente como también se detalla en la especificación relativa a la actualización del estado de firma de la pasarela de servicios web de *Abarcando* para la firma de documentos.

Consultar al soporte técnico de *Abarcando* (soporte@abarcando.com) para conocer 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”.

2.11. Ejemplos

Se presentan extractos de programación en varios lenguajes.

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

Abarcando 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.

2.11.1. Envío de un mensaje en PHP

Ejemplo usando CURL

```
<?php
class AbarcandoSMS {

protected $domainId;
protected $login;
protected $password;
protected $senderId;
protected $encoding;
protected $concat;
protected $timeout=60;
protected $debug;

protected $url = 'http://web.smsverificados.net/api/http';

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 getSenderId() {
    return $this->senderId;
}

public function setSenderId($val) {
    $this->senderId = $val;
    return $this;
}

public function getEncoding() {
    return $this->encoding;
}

public function setEncoding($val) {
    $this->encoding = $val;
    return $this;
}

public function getConcat() {
    return $this->concat;
}

public function setConcat($val) {
    $this->concat = $val;
    return $this;
}

public function getTimeout() {
    return $this->timeout;
}

public function setTimeout($val) {
    $this->timeout = $val;
    return $this;
}

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

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

public function sendSMS($destination, $message) {

    $return=false;
    // Set the curl parameters.
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $this->url);
    curl_setopt($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, $this->getTimeout());
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_POST, 1);

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

//Como destinatarios se admite un array de teléfonos, una cadena de teléfonos
//separados por comas o un único teléfonos
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;
    }
}

//domainId solo es necesario si el login no es un email
$domainId=$this->getDomainId();
if (!isset($domainId) || empty($domainId)) {
    $this->logMsg("NO domainId");
}
else{
    $COMANDO.='&domainId='.$this->getDomainId();
    $this->logMsg("Add domainId ".$this->getDomainId());
}

//No es posible utilizar el remitente en América pero sí en España y Europa
$senderId=$this->getSenderId();
if (!isset($senderId) || empty($senderId)) {
    $this->logMsg("NO senderId ");
}
else{
    $COMANDO.='&senderId='.$senderId;
    $this->logMsg("Add senderId ".$senderId);
}

//Concat. Valores posibles true. Si no se especifica o es false, no se envía
$concat=$this->getConcat();
if (!isset($concat) || empty($concat) || $concat===false) {
    $this->logMsg("NO concat ");
}
else{
```

```
$COMANDO.='&concat=true';
$this->logMsg("Add concat true");
}

//No es posible utilizar el remitente en América pero sí en España y Europa
$encoding=$this->getEncoding();
if (!isset($encoding) || empty($encoding)) {
    $this->logMsg("NO encoding");
}
else{
    if ($encoding==='unicode'){
        $COMANDO.='&encoding='.$encoding;
        $this->logMsg("Add encoding ".$encoding);
    }
    else{
        $this->logMsg("Ignoring encoding. Encoding has no allowed value: ".$encoding);
    }
}

// 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){
    $this->logMsg("Server Abarcando-SMSverificados response: ".$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 function getCredit() {
    $return=false;
    // Set the curl parameters.
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $this->url);
    curl_setopt($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, $this->getTimeout());
```

```
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_POST, 1);

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

//domainId solo es necesario si el login no es un email
$domainId=$this->getDomainId();
$this->logMsg('Dominio: '.$domainId.'');
if (!isset($domainId) || empty($domainId)) {
    $this->logMsg("NO domainId");
}
else{
    $COMANDO.='&domainId='.$this->getDomainId();
    $this->logMsg("Add domainId ".$this->getDomainId());
}

// 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){
    $this->logMsg("Server Abarcando-SMSverificados response: ".$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){
        echo(date(DATE_RFC2822)." : ".$msg."<br>");
        error_log("\n".date(DATE_RFC2822)." : ".$msg."\r\n", 3, "app.log");
    }
}

}
?>

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

```
$abarcandoSMS = new AbarcandoSMS();
//domainId solo es necesario si el login no es un email
//$abarcandoSMS->setDomainId('XX');
$abarcandoSMS->setLogin('YY');
$abarcandoSMS->setPassword('ZZ');

$abarcandoSMS->setDebug(true);

//Use this ONLY with Sender allowed by abarcando sales team
//$abarcandoSMS->setSenderId('TestAbarcando');
//Concatenate messages. If message length is more than 160 characters.
//It will consume as many credits as the number of messages needed
//$abarcandoSMS->setConcat(true);
//Use unicode encoding (only value allowed).
//Can send áéíóú but message length reduced to 70 characters
//$abarcandoSMS->setEncoding('unicode');

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

$response = $abarcandoSMS->sendSMS($sDestination, "Mensaje de prueba");

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

Ejemplo abriendo 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 AbarcandoSMS($sDestination, $sMessage, $debug, $sSenderId){

    $sData ='cmd=sendsms&';
    //domainId solo es necesario si el login no es un email
    // $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('web.smsverificados.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 abarcando<br>";
    return $output;
} else {
    $buf = "POST /api/http HTTP/1.0\r\n";
    $buf .= "Host: web.smsverificados.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 Abarcando-SMSverificados<br>";
        return $output;
    }
    //Se comprueba la respuesta de Abarcando-SMSverificados
    if (strstr($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  
$resp= AbarcandoSMS("346xxxxxxxx,346yyyyyyy", "Mensaje de prueba", false, "");  
//Utilizar esta llamada solo si se cuenta con un remitente autorizado por Abarcando-SMSverificados  
//$resp= AbarcandoSMS("346xxxxxxxx,346yyyyyyy", "Mensaje de prueba", false, "remitente");  
echo $resp;  
  
?>
```

2.11.2. Envío y firma de documento PDF en PHP

Ejemplo usando CURL

```
<?php
class AACert{

    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 certPdf($destination, $type) {

    $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);

    $COMANDO='cmd=certpdf&login='.$this->getLogin().'&passwd='.$this->getPassword();
    $COMANDO.='&destination='.$destination;
    $COMANDO.='&type='.$type;
    $COMANDO.='&websig='.'true';
    if($this->getDomainId() != NULL)
        $COMANDO.= '&domainId='.$this->getDomainId();

    // 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){
        $this->logMsg("Server Abarcando-SMSverificados response: ".$httpResponse);
        $json_parsed = json_decode($httpResponse);
        $status = $json_parsed->status;
        echo 'Código de estado Abarcando-SMSverificados: '.$status.'  
';
        if ($status != '000'){
            $this->logMsg("Error cert pdf: ".$httpResponse);
            return false;
        }else
            $return = $httpResponse;
    }else{
        $this->logMsg("Error cert pdf: ".curl_error($ch).
        '('.curl_errno($ch).)'. $httpResponse);
        $return = false;
    }
    curl_close($ch);
    return $return;
}

public function uploadCertPdf($url, $path) {
```

```
$return=false;

// Set the curl parameters.
$ch = curl_init();

curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-Type: application/pdf'));
$filesize = filesize($path);
$fp = fopen($path, 'rb');
$binary = fread($fp, $filesize);
fclose($fp);
curl_setopt($ch, CURLOPT_POSTFIELDS, $binary);

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

if(curl_getinfo($ch, CURLINFO_HTTP_CODE) === 200){
    $this->logMsg("Server Abarcando-SMSverificados response: ".$httpResponse);

    $json_parsed = json_decode($httpResponse);
    $status = $json_parsed->status;
    echo 'Código de estado Abarcando-SMSverificados: '.$status.'  
';
    if ($status != '000'){
        $this->logMsg("Error upload cert pdf: ".$httpResponse);
        return false;
    }else
        $return = $httpResponse;
}else{
    $this->logMsg("Error upload cert pdf: ".curl_error($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
include('httpPHPAACert.php');

// XX, YY y ZZ se corresponden con los valores de identificacion
// del usuario en el sistema.
$aacert = new AACert();
$aacert->setUrl("http://web.smsverificados.net/api/http");
// domainId solo es necesario si el login no es un email
// $aacert->setDomainId('XX');
```

```
$aaCert->setLogin('YY');
$aaCert->setPassword('ZZ');
$aaCert->setDebug(true);

$response = $aaCert->certPdf('346xxxxxxx', 'simple');

if (!$response)
    echo "El certificado ha terminado en error";
else {
    echo "Response certpdffile: ".$response."\n";
    $decodedJson = json_decode($response, true);
    $responseUpload = $aaCert->uploadCertPdf($decodedJson['url'],'file.pdf');

    if (!$responseUpload)
        echo 'Error al subir el certificado';
    else
        echo 'Response upload certPdf: '.$responseUpload;
}
?>
```

Ejemplo abriendo directamente un socket

```
<?php
// destination: destinatario.
// type: simple/premium
// debug: Si es true muestra por pantalla la respuesta completa del servidor
// XX, YY y ZZ se corresponden con los valores de identificacion del
// usuario en el sistema.
function aaCertPdfFile($destination, $type, $debug){

    $sData ='cmd=certpdffile&';
    // domainId solo es necesario si el login no es un email
    //$sData .= 'domainId=XX&';
    $sData .= 'login=YY&';
    $sData .= 'passwd=ZZ&';
    $sData .= 'destination='.$destination.'&';
    $sData .= 'type='.$type.'&';
    $sData .= 'websig='.'true'.'&';

    //Tiempo máximo de espera para conectar con el servidor = 5 seg
    $timeOut = 5;
    $fp = fsockopen('web.smsverificados.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 abarcando<br>";
        return $output;
    } else {
        $buf = "POST /api/http HTTP/1.0\r\n";
        $buf .= "Host: web.smsverificados.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){
    return $buf;
}

//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 Abarcando-SMSverificados<br>";
    return $output;
}
//Se comprueba la respuesta de Abarcando-SMSverificados
echo "buffer: ".$buf;

$json_parsed = json_decode($buf);
$status = $json_parsed->status;
echo 'Código de estado Abarcando-SMSverificados: '.$status.'<br/>';
if ($status != '000'){
    $output = $buf."<br />\n";
    $output .= " Ha ocurrido algun error. Compruebe la especificacion<br>";
    return $output;
} else
```

```
        return $output;
    }
}

function aaCertPdfUpload($url, $path, $debug){

    //Read cert file
    $filesize = filesize($path);
    $fp = fopen($path, 'rb');
    $sData = fread($fp, $filesize);
    fclose($fp);

    //Tiempo máximo de espera para conectar con el servidor = 5 seg
    $timeOut = 5;
    $fp = fsockopen('web.smsverificados.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 abarcando<br>";
        return $output;
    } else {
        $url = substr($url, strrpos($url, "80")+2);
        $buf = "POST ".$url." HTTP/1.0\r\n";
        $buf .= "Host: web.smsverificados.net\r\n";
        $buf .= "Content-type: application/pdf;\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){
    return $buf;
}

//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 Abarcando-SMSverificados<br>";
    return $output;
}

//Se comprueba la respuesta de Abarcando-SMSverificados
$json_parsed = json_decode($buf);
$status = $json_parsed->status;
echo 'Código de estado Abarcando-SMSverificados: '.$status.'<br/>';
if ($status != '000'){
    $output = $buf."<br />\n";
    $output .= " Ha ocurrido algun error. Compruebe la especificacion<br>";
    return $output;
}else
return $output;
}
}

$resp= aaCertPdfFile('346xxxxxxxx', 'simple', true);
echo "=>Response certpdffile:\n".$resp."\n\n";

//Get json object
$json = substr($resp, strrpos($resp, "{"));

//Parse json object
$decodedJson = json_decode($json, true);

if($decodedJson['status']=='000') {
    $resp= aaCertPdfUpload($decodedJson['url'],'file.pdf',true);
    echo "=>Response certPDF upload:\n".$resp."\n";
}
?>
```

2.11.3. 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://web.smsverificados.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"));
//domainId solo es necesario si el login no es un email
//parametersList.add(new BasicNameValuePair("domainId", "XX"));
parametersList.add(new BasicNameValuePair("login", "YY"));
parametersList.add(new BasicNameValuePair("passwd", "ZZ"));
parametersList.add(new BasicNameValuePair("dest", "346xxxxxxx"));
parametersList.add(new BasicNameValuePair("dest", "346yyyyyyy"));
parametersList.add(new BasicNameValuePair("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 Abarcando-SMSverificados
//parametersList.add(new BasicNameValuePair("senderId", "remitente"));

try {
    //Se fija la codificación de caracteres de la petición 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 Abarcando-SMSverificados");
    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 Abarcando-SMSverificados. 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");
        }
    }
}
}
```

2.11.4. Envío y firma de documento PDF 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

Adicionalmente se utilizan las siguientes librerías:

- gson versión 2.8.5
- commons-io versión 2.6

```
//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://web.smsverificados.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", "certpdffile"));
// domainId solo es necesario si el login no es un email
//parametersList.add(new BasicNameValuePair("domainId", "XX"));
parametersList.add(new BasicNameValuePair("login", "YY"));
parametersList.add(new BasicNameValuePair("passwd", "ZZ"));
parametersList.add(new BasicNameValuePair("destination", "346xxxxxxxx"));
parametersList.add(new BasicNameValuePair("type", "simple"));
parametersList.add(new BasicNameValuePair("websig", "true"));

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

CloseableHttpResponse response = null;
InputStream is = null;
try {
    System.out.println("Enviando petición al comando certPdfFile");
    //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 Abarcando-SMSverificados");
        return;
    }else {
        if(response != null) {
            response.close();
        }
    }
}
```



```

    is.close();
  }catch(IOException ioe) {
    System.out.println("ERROR cerrando recursos");
  }
}
}
}

```

2.11.5. 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://web.smsverificados.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.
'domainId solo es necesario si el login no es un email
objHTTP.Send ("cmd=sendsms&domainId=XX&login=YY&passwd=ZZ&dest=346xxxxxxxx&msg=Mensaje de prueba")
objHTTP.Send ("cmd=sendsms&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 Abarcando-SMSverificados
objHTTP.Send ("cmd=sendsms&login=YY&passwd=ZZ
'
'      &dest=346xxxxxxxx&msg=Mensaje de prueba&senderId=remitente")
MsgBox (objHTTP.Status)
MsgBox (objHTTP.ResponseText)

```

2.11.6. 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://web.smsverificados.net/api/http");

// Compone el mensaje a enviar
// XX, YY y ZZ se corresponden con los valores de identificación del usuario en el sistema.
//domainId solo es necesario si el login no es un email
string lcPostData =
    "cmd=sendsms&login=YY&passwd=ZZ&dest=346xxxxxxxx&dest=346yyyyyyyy" +
    //"&domainId=XX" +
    "&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 Abarcando-SMSverificados
//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 petición
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://web.smsverificados.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.
//domainId solo es necesario si el login no es un email
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", "346xxxxxxxxx"));
postData.Add(new KeyValuePair<string, string>("dest", "346yyyyyyyyy"));
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 Abarcando-SMSverificados
//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://web.smsverificados.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.
'domainId solo es necesario si el login no es un email
Dim lcPostData As String =
    "cmd=sendsms&login=YY&passwd=ZZ&dest=346xxxxxxx&dest=346yyyyyyy" +
    "&domainId=XX" +
    "&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 Abarcando-SMSverificados
' 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 petición
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://web.smsverificados.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.
'domainId solo es necesario si el login no es un email
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", "346xxxxxxx"))
```

```
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 Abarcando-SMSverificados
' 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
```

2.11.7. 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://web.smsverificados.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');
    //domainId solo es necesario si el login no es un email
    //Parametros.Add('domainId=XX');
    Parametros.Add('login=YY');
    Parametros.Add('passwd=ZZ');
    Parametros.Add('dest=346xxxxxxx');
    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 Abarcando-SMSverificados
```

```
//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://web.smsverificados.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');
//domainId solo es necesario si el login no es un email
//Parametros.Add('domainId=XX');
Parametros.Add('login=YY');
Parametros.Add('passwd=ZZ');
Parametros.Add('dest=346xxxxxxx');
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 Abarcando-SMSverificados
//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.
```

2.11.8. 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://web.smsverificados.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
#domainId solo es necesario si el login no es un email
my $data = ("cmd=sendsms&login=YY&passwd=ZZ&dest=346xxxxxxx&dest=346yyyyyyyy".
    "&domainId=XX".
    "&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 Abarcando-SMSverificados
    "#&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;
}
```

2.11.9. Envío y firma de documento PDF en Perl

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

Es preciso además instalar los módulos JSON y JSON::Parse.

```
#!/usr/bin/perl

use strict;
use warnings;
use LWP::UserAgent;
use utf8;
use Encode qw(decode encode);
use JSON;
use JSON::Parse 'parse_json';
use Try::Tiny;

binmode(STDOUT, ":utf8");

try {
    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://web.smsverificados.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
    # domainId solo es necesario si el login no es un email
    my $data =("cmd=certpdffile&login=XX&passwd=YY".
        "&destination=346xxxxxxxx&type=simple&websig=true".
        "#&domainId=ZZ".
        "");

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

    $req->content($data);

    my $resp = $ua->request($req);
    if ($resp->is_success) {
        # $resp->code = 200
        my $message = $resp->decoded_content;
        print "\nRespuesta comando certPdfFile: \n$message\n";

        my $parsedJson = parse_json ($message);
```


2.11.10. Envío de un mensaje en Ruby

Ejemplo en Ruby de cliente HTTP.

```
# encoding: UTF-8

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

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

  begin

    #Se fija la URL sobre la que enviar la petición POST
    url = "http://web.smsverificados.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
    #domainId solo es necesario si el login no es un email
    post_data = {'cmd' => 'sendsms', 'login' => 'YY', 'passwd' => 'ZZ',
    #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 Abarcando SMSverificados: #{response.body}")
        else
          puts("Cuerpo de la respuesta: \n#{response.body}")
        end
      end
    end
  end

  return response
end
```

```
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 abarcandoSms returns:"+
  "#{abarcandoSms('346xxxxxxxx,346yyyyyyy','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 Abarcando-SMSverificados
#puts "The function abarcandoSms returns:"+
  "###{abarcandoSms('346xxxxxxxx,346yyyyyyy','Mensaje de prueba', 'remitente', true).body}"
```

2.11.11. Envío y firma de documento PDF en Ruby

Ejemplo en Ruby de cliente HTTP.

```
# encoding: utf8

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

def aaCert(destination, type)
  puts "Enter aaCert: destination=#{destination}, type=#{type}"

  begin
    #Se fija la URL sobre la que enviar la petición POST
    url = "http://web.smsverificados.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.
    post_data = {'cmd' => 'certpdffile',
                 # domainId solo es necesario si el login no es un email
                 #'domainId' => 'XX',
                 'login' => 'YY', 'passwd' => 'ZZ',
                 'destination' => destination, 'type' => type, 'webSig' => 'true'}

    #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)
    unless response.code == "200" #Error en la respuesta del servidor
      puts("ERROR GENERAL cmd: #{response.code}")
      puts("#{response.body}")
    end
  end
end
```

```
else #Se procesa la respuesta
  puts("Código de estado HTTP cmd: #{response.code}")
  puts("Cuerpo respuesta cmd: #{response.body}")
  parsedJson = JSON.parse(response.body)
unless parsedJson['status'].include? "000"
  puts("Error de Abarcando SMSverificados cmd: #{response.body}")
else
  puts("Código de estado Alriria cmd: #{parsedJson['status']}")

  file = File.open("file.pdf", "rb")

  #Se fija la URL base de los recursos REST
  uri = URI.parse(parsedJson['url'])

  #Se inicia el objeto HTTP y se envía la petición
  #Se añade el JSON al cuerpo de la petición codificado en UTF-8
  request = Net::HTTP::Post.new(uri.request_uri, 'Content-Type' => 'application/pdf')
  request.body = file.read

  #Se consigue la respuesta
  response = http.request(request)
  unless response.code == "200" #Error en la respuesta del servidor
    puts("ERROR GENERAL subiendo fichero: #{response.code}")
    puts("#{response.body}")
  else #Se procesa la respuesta
    puts("Código de estado HTTP subiendo fichero: #{response.code}")
    puts("Cuerpo respuesta subiendo fichero: #{response.body}")
    parsedJson = JSON.parse(response.body)
    unless parsedJson['status'].include? "000"
      puts("Error de Abarcando SMSverificados subiendo fichero: #{response.body}")
    else
      puts("Proceso terminado con éxito")
    end
  end
end
end
end
rescue Net::OpenTimeout
  puts "Tiempo de conexión agotado"
rescue Net::ReadTimeout
  puts "Tiempo de respuesta agotado"
rescue Exception => e
  puts "Error interno: #{e}"
end
end

aaCert('346xxxxxxxx', 'simple')
```

2.11.12. Envío de un mensaje en Python

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

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

import requests

def abarcandoSms(destinations, message, senderId, debug):
    if debug:
        print 'Enter abarcandoSms: '+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.
        #domainId solo es necesario si el login no es un email
        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 codificación de caracteres de la petición 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://web.smsverificados.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 Abarcando SMSverificados: '+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 abarcandoSms returns: \n'
+abarcandoSms('346xxxxxxxx,346yyyyyyy','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 Abarcando-SMSverificados
#print 'The function abarcandoSms returns: \n'
# +abarcandoSms('346xxxxxxxx,346yyyyyyy','Mensaje de prueba', 'remitente', True)
```

2.11.13. Envío y firma de documento PDF en Python

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

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

import requests
import json

def aaCert(destination, fType):
    print 'Enter aaCert: destination= '+destination+', fType= '+fType

    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', 'certpdffile'),
            #domainId solo es necesario si el login no es un email
            ('domainId', 'XX'),
            ('login', 'YY'),
            ('passwd', 'ZZ'),
            ('destination', destination),
            ('type', fType),
            ('webSig', 'true')
        ]

        #Se fija la codificación de caracteres de la petición 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://web.smsverificados.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 str(r.status_code) != '200': #Error en la respuesta del servidor
    print 'Error general comando certPdfFile: '+str(r.status_code)
else: #Se procesa la respuesta
    print 'Codigo de estado HTTP cmd: '+str(r.status_code)
    print 'Respuesta certPdfFile: '+str(r.text)
    parsed_json = json.loads(r.text)
    status = parsed_json['status']
    if status == '000':
        print 'Codigo de estado Abarcando-SMSverificados cmd: '+status
        url = parsed_json['url']
        f = open("file.pdf", "rb")
        try:
            contentType = {'Content-Type': 'application/pdf'}
            r = requests.post(url,
                data=f,
                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)
        finally:
            f.close()
    if str(r.status_code) != '200': #Error en la respuesta del servidor
        print 'Error general subiendo fichero: '+str(r.status_code)
    else: #Se procesa la respuesta
        print 'Codigo de estado HTTP subiendo fichero: '+str(r.status_code)
        print 'Respuesta subiendo fichero: '+str(r.text)
        parsed_json = json.loads(r.text)
        status = parsed_json['status']
        if status == '000':
            print 'Proceso terminado con exito'
        else:
            print "Error Abarcando-SMSverificados. Codigo de estado: "+status
    else:
        print "Error Abarcando-SMSverificados. Codigo de estado: "+status

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)

aaCert('346xxxxxxxx', 'simple')
```

2.11.14. 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
  //domainId solo es necesario si el login no es un email
  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: 'web.smsverificados.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 Abarcando-SMSverificados
//sendSMS('XX', 'YY', 'ZZ', '346xxxxxxxx,346yyyyyyyy', 'remitente', 'Mensaje de prueba');
```

2.11.15. Envío y firma de documento PDF en node.js

Ejemplo en node.js.

```
const querystring = require('querystring');
const http = require('http');
const URL = require('url').URL;
const fs = require('fs');

function aaCert(domainId, login, passwd, destination, type) {
  // Se contruye la cadena del post desde un objeto
  // domainId solo es necesario si el login no es un email
  var post_data = querystring.stringify({
    'cmd' : 'certpdffile',
    //'domainId' : domainId,
    'login': login,
    'passwd': passwd,
    'destination' : destination,
    'type' : type,
    'webSig': 'true'
  });

  // Un objeto de opciones sobre donde se envia el post
  var post_options = {
    host: 'web.smsverificados.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 (data) {
      //Es necesario procesar la respuesta y los posibles errores
      console.log('Respuesta cmd: ' + data);
      var parsedJson = JSON.parse(data);
      if(parsedJson.status!='000') {
        console.log("Error de Abarcando SMSverificados cmd:"+data);
      }else{
        console.log("Codigo de estado Alriria cmd: "+parsedJson.status);
        var options = new URL(parsedJson.url);
        const imageBuffer = fs.readFileSync('file.pdf');

        // Un objeto de opciones sobre donde se envia el post
        var post_options = {
          host: options.hostname,
          port: options.port,
          path: options.pathname,
          method: 'POST',
          headers: {
```

```
        'Content-Type': 'application/pdf'
    }
};
var post_req = http.request(post_options, function(res) {
    res.setEncoding('utf8');
    res.on('data', function (data) {
        console.log('Respuesta subir fichero: ' + data);
        var parsedJson = JSON.parse(data);
        if(parsedJson.status!='000') {
            console.log("Error de Abarcando SMSverificados al subir fichero:"+data);
        }else{
            console.log("Proceso terminado con exito");
        }
    }, function (error) {
        console.log("Codigo de estado HTTP subiendo fichero: "+error)
    });
    post_req.write(imageBuffer);
    post_req.end();
}
}, function (error) {
console.log("Codigo de estado HTTP cmd: "+error)
});
});
// 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.
// domainId solo es necesario si el login no es un email
aaCert('XX','YY','ZZ','346xxxxxxxx','simple');
```

Referencias

[SMSVERIFICADOS2021] *Plataforma SMSVERIFICADOS de Abarcando:*

<https://web.smsverificados.net/web>

[LANDINGS] *Páginas webs adaptadas a móvil:*

<https://smsverificados.com/sms-landing>

[FAQ] *Preguntas frecuentes de la pasarela de envío de SMS de Abarcando-SMSverificados:*

<http://smsverificados.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/nussoap/>

[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://docs.abarcando.com/abarcando-push-ejemplos-codigo-fuente.zip>

[REQUESTS] *Página oficial de Requests:*

<http://docs.python-requests.org>

[SAVON] *Página oficial de Savon:*

<http://savourb.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>

[POINTS2MM] *Conversión de puntos a milímetros:*

<https://www.google.com/search?q=point+to+mm>