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

ABARCANDO, S.L.

Versión: 1.5

Historial de cambios

Versión	Cambios
1.5	Se añade el parámetro opcional "multiSig" al recurso "certPdfFile" para definir más de un firmante por documento así como la posición de la firma (2.8.1). Se añade la sección 2.8.2 para detallar todos los aspectos relativos a la ubicación de las firmas en el documento. Se añaden los códigos de error 012, 013, 030, 031 y 032 (sección 2.8.5).
1.4	Se añaden los parámetros opcionales "title" y "forwardEmail" al recurso "certPdfFile" para la firma de documentos (2.8.1). Se actualiza la información devuelta en el recurso "checkPdfFile" (2.8.4).
1.3	Se añaden dos nuevas secciones sobre la firma de documentos (2.8) y la actualización de su estado (2.9). Se actualizan los ejemplos de programación (sección 2.10).
1.2	Se añade el parámetro opcional "certDelivery" a los recursos "sendSms" y "sendSmsMulti" para solicitar la certificación de la entrega del SMS enviado mediante un documento digital con sello certificado de tiempo (secciones 2.3.1 y 2.3.2). Se añade una nueva notificación de estado "CERT(url)" para recibir la URL de acceso al documento del certificado de entrega (cuadro 2.15).
1.1	Se añade una nueva sección sobre los enlaces a web móvil (2.6) y una subsección sobre la parametrización de webs móviles (2.6.1). Se actualiza la información del parámetro "msg" y se añaden los parámetros "landing", "params" y "webParams" en los cuadros 2.2 y 2.4, todo en relación a los enlaces a web móvil (sección 2.6). Se añaden los códigos de error 035, 036, 037, 038 y 039 (sección 2.3.5).
1.0	Primera versión del documento.

Índice general

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 recursos REST.

El servicio de envío de mensajes cortos está disponible en muchos paises. Para conocer los paises permitidos, las operadoras válidas en cada país y las posibles restricciones geográficas (salvedades al funcionamiento general detallado en este documento que pudieran aplicar en cada caso) se puede enviar un correo electrónico a comercial@abarcando.com.

El servicio opcional de confirmación de entrega requiere que el cliente exponga un recurso REST para recibir la información de confirmación (ver la sección 2.7).

Capítulo 2

Descripción de la API

2.1. Acceso a los recursos REST

La **URL base** de acceso a los recursos REST es **http://web.smsverificados.net/apirest/ws**

Cada petición REST enviada se corresponde con un recurso expuesto de la API, según se detalla en los siguientes apartados.

El cuerpo de cada petición REST está compuesto por un mensaje en formato JSON con el *"content-type:application/json;charset=UTF-8"*.

Se debe usar la **codificación UTF-8** en la comunicación con el servidor.

En el apartado 2.10 se dan varios ejemplos.

2.2. Respuesta a la petición REST

Cada petición de recurso REST lleva asociada una respuesta desde el servidor de Abarcando-SMSverificados en formato **JSON codificada en UTF-8**.

En los siguientes apartados se detalla la respuesta para cada petición siempre que resulte exitosa. En esos casos el código de estado HTTP será 200.

Si se produjese algún error en el servidor el código de estado HTTP denotará un fallo particular (será distinto de 200). Puede ocurrir si se ha producido un error de "binding", se intenta acceder a un recurso que no existe o se viola alguna restricción de alguno de los elementos del JSON de la petición. En esos casos el JSON de respuesta tendrá un único elemento "error" para informar sobre el problema.

Un ejemplo de respuesta debido a un error en la petición del cliente (falta el parámetro obligatorio "login") es el siguiente:

```
{"error":"LOGIN_NOT_NULL"}
```

2.3. Recursos de la API de envío de SMS

A continuación se detallan los recursos disponibles en la API REST y los elementos que componen el mensaje JSON de la petición. Cada elemento puede ser obligatorio u opcional.

Cabe citar que el nombre de todos los elementos del JSON podrá ser remitido en diferentes formatos:

- Según la notación Java ("domainId").

- Según la notación REST ("domain_id").
- Todo en minúsculas ("domainid").

Las **peticiones** a cada recurso se harán sobre la **URL base** suministrada, **concatenando el "path"** propio de cada recurso.

2.3.1. Envío de un mensaje de texto

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

Se trata del recurso con **"path" /sendSms**. El JSON está compuesto por los **elementos obligatorios** detallados en el cuadro 2.1.

Nombre	Valores	Composición
credentials	Datos de identificación del usuario suministrados por *Abarcando*	Elementos domainId, login y passwd
destination	Lista de números de teléfono móvil de los destinatarios del mensaje	Cada número 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
message	Datos propios del mensaje a enviar	Elementos msg, senderId, ack, idAck, dPort, sPort, encoding, concat y certDelivery

Cuadro 2.1: Parámetros de entrada del recurso sendSms

Para enviar el **mensaje a varios destinatarios** basta con añadir los diferentes números de teléfono al elemento "destination" 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.

En el cuadro 2.2 se detallan los elementos restantes constituyentes del JSON de la petición.

Nombre	Valor	Obligatorio
domainId	Identificador suministrado por *Abarcando* al cliente. Se puede omitir si el login es un email.	no
login	Identificador de usuario suministrado por *Abarcando* al cliente.	sí
passwd	Clave del usuario suministrada por *Abarcando* al cliente.	sí
msg	Mensaje a enviar. La lista de caracteres válidos y la longitud máxima permitida se detalla en la sección 2.4. 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 2.6 para conocer más detalles sobre las webs móviles.	sí

senderId	Remitente del mensaje a enviar, autorizado por *Abarcando* . 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 2.6 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 el **parámetro "webParams"**. Consultar la sección 2.6.1 para conocer más detalles sobre la parametrización de las webs móviles.	no
webParams	Lista de **pares id/params** para definir los parámetros de cada web móvil parametrizada. **Su uso es obligatorio** en caso de que "params" sea mayor que 0. Consultar la sección 2.6.1 para conocer más detalles sobre la parametrización de las webs móviles.	no
ack	Solicitud de confirmación de entrega de los mensajes enviados (ver sección 2.7). 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 2.7). Valor alfanumérico de hasta 20 caracteres (números y letras de la "a" a la "z" tanto mayúsculas como minúsculas sin incluir ni "Ñ" ni "ñ"). De rebasar la longitud máxima permitida será truncado. Los caracteres no permitidos serán eliminados. Solo será considerado si el parámetro ack se envía con valor "true". Si se incluye explícitamente este parámetro y toma como valor cadena vacía, anula la solicitud de confirmación de entrega.	no
dPort	Puerto destino del SMS a enviar. Valor numérico entre 1 y 65535 (solo dígitos decimales). Solo necesario si se desea cambiar el valor habitual de esta propiedad. Si se define, la longitud máxima del mensaje a enviar (parámetro "msg") se verá reducida (ver la sección 2.4.3) 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 2.4.3) 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 2.4.2). 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 2.4.3). 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 2.2: Lista de parámetros para sendsms

Un ejemplo de petición REST al recurso sendSms solicitando el envío de un mensaje concatenado a dos destinatarios y la certificación de la entrega sería este:

```
{
"credentials":{"domainId":"XXXXX",
   "login":"YYYYY",
   "passwd":"ZZZZZ"},
"destination":["346XXXXXXXX","346YYYYYYYY"],
"message":{"msg":"Ejemplo de mensaje concatenado enviado a más de un destinatario con
  la codificación UNICODE para admitir las vocales acentuadas y
  solicitud de confirmación de entrega.",
   "senderId":"remitente",
   "ack":"true",
   "idAck":"123456789",
   "concat":"true",
   "certDelivery":"true",
   "encoding":"unicode"}
}
```

La respuesta a la petición del recurso sendSms está compuesta por el elemento “status” con valor un código de estado general de los descritos en el apartado 2.3.5

Si la operación ha resultado exitosa (código de estado “000”) la respuesta contendrá adicionalmente un elemento “details” incluyendo para cada destinatario del envío (cada elemento de la lista “destination” de entrada) los siguientes datos:

- destination: 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 datos independientes para cada fragmento siendo cualificado el valor de “destination” con un sufijo que diferencie cada fragmento con un índice numérico comenzando por 0. Por ejemplo para un mensaje concatenado de tres fragmentos enviado al número “xxxxxxxxxxx” se recibirán datos para destination=xxxxxxxxxxx(0), destination=xxxxxxxxxxx(1) y destination=xxxxxxxxxxx(2).
- status: se corresponde con uno de los códigos de estado del apartado 2.3.5.
- idAck: se corresponde con el código de identificación asociado a la solicitud de confirmación de entrega (ver sección 2.7). Solo aparecerá si se solicita la confirmación de entrega y es aceptada.

Un ejemplo de respuesta exitosa correspondiente a la petición al recurso **sendSms** del ejemplo anterior sería el mostrado a continuación. Al tratarse de un mensaje concatenado aparecen datos para cada uno de los tres fragmentos que lo componen para cada destinatario del envío:

```
{
"details":
[{"destination":"346XXXXXXXX(0)","idAck":"123456789","status":"000"},
 {"destination":"346XXXXXXXX(1)","idAck":"123456789","status":"000"},
 {"destination":"346XXXXXXXX(2)","idAck":"123456789","status":"000"},
 {"destination":"346YYYYYYYY(0)","idAck":"123456789","status":"000"},
 {"destination":"346YYYYYYYY(1)","idAck":"123456789","status":"000"},
 {"destination":"346YYYYYYYY(2)","idAck":"123456789","status":"000"}]
,"status":"000"
}
```

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

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

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 2.3.4).

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

2.3.2. Envío de múltiples mensajes de texto

Permite enviar una lista de mensajes cortos de texto, cada cual al destinatario indicado.

Se trata del recurso con **"path" /sendSmsMulti**. El JSON está compuesto por los **elementos obligatorios** detallados en el cuadro 2.3.

Nombre	Valores	Composición
credentials	Datos de identificación del usuario suministrados por *Abarcando*	Elementos domainId, login y passwd
messages	Lista de mensajes. Cada elemento de la lista define los datos propios de un mensaje a enviar	Elementos msg, senderId, ack, idAck, dPort, sPort, encoding, concat y certDelivery

Cuadro 2.3: Parámetros de entrada del recurso sendSmsMulti

Para enviar más de un mensaje basta con añadir los diferentes mensajes al elemento "messages", tantos como sea preciso sin sobrepasar el límite máximo permitido (consultar al soporte técnico de *Abarcando* en soporte@abarcando.com).

En el cuadro 2.4 se detallan los elementos restantes constituyentes del JSON de la petición.

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

destination	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 2.4. 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 2.6 para conocer más detalles sobre las webs móviles.	sí
senderId	Remitente del mensaje a enviar, autorizado por *Abarcando* . 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 2.6 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 el **parámetro "webParams"**. Consultar la sección 2.6.1 para conocer más detalles sobre la parametrización de las webs móviles.	no
webParams	Lista de **pares id/params** para definir los parámetros de cada web móvil parametrizada. **Su uso es obligatorio** en caso de que "params" sea mayor que 0. Consultar la sección 2.6.1 para conocer más detalles sobre la parametrización de las webs móviles.	no
idMsg	Referencia del mensaje para facilitar su identificación al procesar la respuesta a la petición de envío múltiple.	no
ack	Solicitud de confirmación de entrega de los mensajes enviados (ver sección 2.7). Si vale "true" solicita confirmación de entrega de los SMS enviados. 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 2.7). Valor alfanumérico de hasta 20 caracteres (números y letras de la "a" a la "z" tanto mayúsculas como minúsculas sin incluir ni "Ñ" ni "ñ"). De rebasar la longitud máxima permitida será truncado. Los caracteres no permitidos serán eliminados. Solo será considerado si el parámetro ack se envía con valor "true". Si se incluye explícitamente este parámetro y toma como valor cadena vacía, anula la solicitud de confirmación de entrega.	no

dPort	Puerto destino del SMS a enviar. Valor numérico entre 1 y 65535 (solo dígitos decimales). Solo necesario si se desea cambiar el valor habitual de esta propiedad. Si se define, la longitud máxima del mensaje a enviar (parámetro "msg") se verá reducida (ver la sección 2.4.3) 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 2.4.3) 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 2.4.2). 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 2.4.3). 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 2.4: Lista de parámetros para sendsmsmulti

Un ejemplo de petición REST al recurso sendSmsMulti solicitando el envío de tres mensajes sería este:

```
{"credentials":{"domain_id":"XXXXX",
                      "login":"YYYYY",
                      "passwd":"ZZZZZ"},
 "messages":[{"msg":"Mensaje de prueba 1",
                     "destination":"346XXXXXXXX"},
   {"msg":"Lorem Ipsum es simplemente el texto de relleno de las imprentas
               y archivos de texto. Lorem Ipsum ha sido el texto de relleno
               estandar de las industrias desde el año 1500",
     "destination":"346YYYYYYYY","concat":true,"cert_delivery":true,"id_msg":"id2"},
   {"msg":"Mensaje de prueba 3",
     "destination":"346ZZZZZZZZ",
     "sender_id":"remitente","ack":true,"id_ack":"123456789",
     "d_port":5000,"s_port":4000,"id_msg":"id3"}]
}
```

La respuesta a la petición del recurso sendSmsMulti está compuesta por el elemento "status" con valor un código de estado general de los descritos en el apartado 2.3.5

Si la operación ha resultado exitosa (código de estado "000") la respuesta contendrá adicionalmente un elemento "details" incluyendo para cada mensaje del envío (cada elemento de la lista "messages" de entrada) los siguientes datos:

- destination: se corresponde con el número de teléfono del destinatario. Si se hubiese enviado un

mensaje concatenado (ver el parámetro "concat") al destinatario le corresponderán varios mensajes, tantos como fragmentos compongan el mensaje concatenado. En ese caso aparecerán datos independientes para cada fragmento siendo cualificado el valor de "destination" con un sufijo que diferencie cada fragmento con un índice numérico comenzando por 0. Por ejemplo para un mensaje concatenado de tres fragmentos enviado al número "xxxxxxxxxxx" se recibirán datos para destination=xxxxxxxxxxx(0), destination=xxxxxxxxxxx(1) y destination=xxxxxxxxxxx(2).

- status: se corresponde con uno de los códigos de estado del apartado 2.3.5.
- idAck: se corresponde con el código de identificación asociado a la solicitud de confirmación de entrega (ver sección 2.7). Solo aparecerá si se solicita la confirmación de entrega y es aceptada.
- idMsg: se corresponde con la referencia asociada al mensaje remitida en la petición de envío. Solo aparecerá si se ha incluido en la petición al servidor.

Un ejemplo de respuesta exitosa correspondiente a la petición al recurso **sendSmsMulti** del ejemplo anterior sería el mostrado a continuación. Al ser el segundo mensaje concatenado aparecen datos para cada uno de los dos fragmentos que lo componen:

```
{"details":[
  {"destination":"346XXXXXXXX","status":"000"},
  {"destination":"346YYYYYYYY(0)","idMsg":"id2","status":"000"},
  {"destination":"346YYYYYYYY(1)","idMsg":"id2","status":"000"},
  {"destination":"346ZZZZZZZZ","idAdk":"123456789","idMsg":"id3","status":"000"}
  ],
"status":"000"}
```

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

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

La infomación de éxito para un mensaje 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 2.3.4).

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

2.3.3. Envío de un mensaje multimedia WAP-PUSH

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

Para conocer en qué consisten los mensajes de este tipo se aconseja leer la sección 2.5.

Para saber más sobre el proceso de envío de mensajes WAP-PUSH a través de la pasarela se aconseja leer la sección 2.5.7.

Se trata del recurso con **"path" /sendWapPush**. El JSON está compuesto por los **elementos obligatorios** detallados en el cuadro 2.5.

Nombre	Valores	Composición
credentials	Datos de identificación del usuario suministrados por *Abarcando*	Elementos domainId, login y passwd
destination	Lista de números de teléfono móvil de los destinatarios del mensaje	Cada número 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
message	Datos propios del mensaje a enviar	Elementos msg, url, ack y idAck

Cuadro 2.5: Parámetros de entrada del recurso sendWapPush

Para enviar el **mensaje a varios destinatarios** basta con añadir los diferentes números de teléfono al elemento "destination" 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.

En el cuadro 2.6 se detallan los elementos restantes constituyentes del JSON de la petición.

Nombre	Valor	Obligatorio
domainId	Identificador suministrado por *Abarcando* al cliente. Se puede omitir si el login es un email.	no
login	Identificador de usuario suministrado por *Abarcando* al cliente.	sí
passwd	Clave del usuario suministrada por *Abarcando* al cliente.	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 2.5.1. 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 2.5.7). 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 2.5.2). 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 2.5.7). No puede estar vacío. No es posible seleccionar un puerto de conexión diferente al 80, el habitual en la navegación WEB.	sí
ack	Solicitud de confirmación de entrega de los mensajes enviados (ver sección 2.7). Si vale "true" solicita confirmación de entrega de los SMS enviados. En su ausencia o si tiene otro valor no se solicita confirmación de entrega.	no
idAck	Código identificativo para la confirmación de entrega (ver sección 2.7). Valor alfanumérico de hasta 20 caracteres (números y letras de la "a" a la "z" tanto mayúsculas como minúsculas sin incluir ni "Ñ" ni "ñ"). De rebasar la longitud máxima permitida será truncado. Los caracteres no permitidos serán eliminados. Solo será considerado si el parámetro ack se envía con valor "true". Si se incluye explícitamente este parámetro y toma como valor cadena vacía, anula la solicitud de confirmación de entrega.	no

Cuadro 2.6: Lista de parámetros para sendwappush

Un ejemplo de petición REST al recurso sendWapPush sería este:

```
{"credentials":{"domainId":"XXXXX","login":"YYYYYY","passwd":"ZZZZZZ"},
 "message":{"msg":"Texto de alerta","url":"http://smsverificados.com?k=",
 "ack":"true","idAck":123456789},
 "destination":["346XXXXXXXX","346YYYYYYYY"]}
```

La respuesta a la petición del recurso sendWapPush está compuesta por el elemento "status" con valor un código de estado general de los descritos en el apartado 2.3.5

Si la operación ha resultado exitosa (código de estado "000") la respuesta contendrá adicionalmente un elemento "details" incluyendo para cada destinatario del envío (cada elemento de la lista "destination" de entrada) los siguientes datos:

- destination: se corresponde con el número de teléfono del destinatario.
- status: se corresponde con uno de los códigos de estado del apartado 2.3.5.
- idAck: se corresponde con el código de identificación asociado a la solicitud de confirmación de entrega (ver sección 2.7). Solo aparecerá si se solicita la confirmación de entrega y es aceptada.
- key: se corresponde con la clave única asociada al destinatario (ver sección 2.5.7). Solo aparecerá si se solicita la clave para cada destinatario

Un ejemplo de respuesta exitosa correspondiente al recurso **sendWapPush** del ejemplo anterior sería el mostrado a continuación:

```
{"details":[{"destination":"346XXXXXXXX","idAck":"123456789",
 "key":"346XXXXXXXX-22879196","status":"000"},
{"destination":"346YYYYYYYY","idAck":"123456789",
 "key":"346YYYYYYYY-22879196","status":"000"}],
"status":"000"}
```

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

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

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 2.3.4).

Para asegurar el adecuado funcionamiento de este recurso 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.

2.3.4. Consulta del crédito disponible

Permite conocer el crédito instantáneo disponible para enviar mensajes.

Se trata del recurso con **"path" /getCredit**. El JSON está compuesto por los **elementos obligatorios** detallados en el cuadro 2.7.

Nombre	Valores	Composición
credentials	Datos de identificación del usuario suministrados por *Abarcando*	Elementos domainId, login y passwd

Cuadro 2.7: Parámetros de entrada del recurso getCredit

En el cuadro 2.8 se detallan los elementos restantes constituyentes del JSON de la petición.

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 recurso.

El recurso 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 recurso 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 las peticiones 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**
domainId	Identificador suministrado por *Abarcando* al cliente. Se puede omitir si el login es un email.	no
login	Identificador de usuario suministrado por *Abarcando* al cliente.	sí
passwd	Clave del usuario suministrada por *Abarcando* al cliente.	sí

Cuadro 2.8: Lista de parámetros para getcredit

Un ejemplo de petición REST al recurso getCredit sería este:

```
{"credentials":{"domainId":"XXXXXX","login":"YYYYYY","passwd":"ZZZZZZ"}}
```

La respuesta a la petición del recurso getCredit está compuesta por el elemento "status" con valor un código de estado general de los descritos en el apartado 2.3.5

Si la operación ha resultado exitosa (código de estado "000") la respuesta contendrá adicionalmente el valor del crédito disponible como un número con dos decimales.

Un ejemplo de respuesta exitosa correspondiente al servicio **getCredit** del ejemplo anterior sería el mostrado a continuación:

```
{"credit":"1000000.70","status":"000"}
```

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

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

2.3.5. Códigos de estado

El cuadro 2.9 presenta la lista de los posibles códigos de estado que podrán aparecer en la respuesta a cada petición del API de envío de SMS (ver 2.3).

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
010	Error en el formato del número de teléfono
011	Error en el envío de los parámetros de la petición o codificación incorrecta.
013	El mensaje excede la longitud máxima permitida
014	La petición HTTP usa una codificación de caracteres inválida
015	No hay destinatarios válidos para enviar el mensaje
016	Destinatario duplicado
017	Mensaje vacío
018	Se ha excedido el máximo número de destinatarios autorizado
019	Se ha excedido el máximo número de mensajes autorizado
020	Error en la autentificación
022	El remitente seleccionado para el envío no es válido
030	La url y el mensaje superan la longitud máxima permitida
031	La longitud de la url es incorrecta
032	La url contiene caracteres no permitidos
033	El puerto destino del SMS es incorrecto
034	El puerto origen del SMS es incorrecto
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 2.9: Lista de los códigos de estado

2.4. 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 2.4.1) o la codificación UNICODE (ver sección 2.4.2).

2.4.1. Codificación por defecto

La **codificación por defecto** permite los caracteres de la tabla 2.10.

La **longitud máxima permitida** se detalla en la sección 2.4.3.

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 2.11. Cada carácter extendido **ocupa el doble espacio que un carácter normal**, esto debe considerarse para el cómputo de la longitud máxima del mensaje.

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

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

Cuadro 2.10: Lista de caracteres permitidos para mensajes de texto en la codificación por defecto

[	]	\	^	{	}	|	~	€

Cuadro 2.11: Lista de caracteres extendidos permitidos para mensajes de texto

2.4.2. Unicode

La **codificación UNICODE**, forzada mediante el parámetro "unicode" (ver el cuadro 2.2), permite todo el juego de caracteres UNICODE de 16bits.

La **longitud máxima permitida** se detalla en la sección 2.4.3, siendo siempre menor que usando la codificación por defecto (ver la sección 2.4.1).

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

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

2.4.3. 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 2.4.1). Los caracteres extendidos (ver la tabla 2.11) 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 2.4.2).

En caso de **definir puertos origen o destino** del SMS (ver los parámetros sPort y dPort en el cuadro 2.2) la **longitud máxima se reduce** de la siguiente forma:

- **152 caracteres para la codificación por defecto** (ver sección 2.4.1). Igualmente hay que considerar que los caracteres extendidos (ver la tabla 2.11) ocupan el doble.
- **66 caracteres para la codificación UNICODE** (ver sección 2.4.2).

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 2.2) siempre que no se esten definiendo ni el puerto origen ni el puerto destino del SMS (ver los parámetros sPort y dPort en el cuadro 2.2).

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 2.4.1). Igualmente hay que considerar que los caracteres extendidos (ver la tabla 2.11) ocupan el doble.
- **670 caracteres para la codificación UNICODE** (ver sección 2.4.2).

2.5. 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...).

2.5.1. Caracteres permitidos

El cuadro 2.12 detalla los caracteres admisibles para los mensajes WAP-PUSH (parámetro “msg” del cuadro 2.6).

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 2.12: 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.

2.5.2. Especificación de la URL

La URL de descarga de los contenidos multimedia suministrados a través de mensajes WAP-PUSH (parámetro “url” del cuadro 2.6) debe seguir las siguientes normas de composición:

- Los caracteres del cuadro 2.13 son seguros y se pueden incluir sin codificar.

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

- Los caracteres del cuadro 2.14 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 2.13: Lista de caracteres seguros

$	&	+	,	/	:	;	=	?	@

Cuadro 2.14: Lista de caracteres reservados

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

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

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

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

se debe enviar como (usando ISO8859-1):

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

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

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.

2.5.3. 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 2.12, sin incluir las vocales acentuadas.

2.5.4. 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".

2.5.5. 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.

2.5.6. 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.

2.5.7. Control de acceso al contenido

La URL indicada en el campo "url" del recurso "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 recurso "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: "xxxxxxxxxxx-zzzzzzzzzz"; 'xxxxxxxxxxx" representa el número de teléfono del destinatario en formato internacional y "zzzzzzzzzz" representa un número asociado a cada petición al recurso "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 recurso "sendWapPush" se esquematizan los procesos involucrados en el envío de un mensaje a dos destinatarios solicitando claves independientes:

1. El cliente accede al recurso "sendWapPush" de la pasarela de *Abarcando* incluyendo los siguientes parámetros:
   - destination=346xxxxxxxx.
   - destination=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 al cliente un status general "000" y los siguientes datos para cada destinatario:

   ```
   status=000; destination=346xxxxxxxx; key=346xxxxxxxx-12365
   status=000; destination=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.

2.6. 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, envie 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" de los cuadros 2.2 y 2.4) 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:

```
{
"credentials":{"domainId":"XXXXX","login":"YYYYY","passwd":"ZZZZZ"},
"destination":["346XXXXXXXX"],
"message":{"msg":"Estimado cliente, acceda a la promoción
          en el siguiente enlace: {$15880$}",
"landing":1,
"params":0}
}
```

El 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 los cuadros 2.2 y 2.4) 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 los cuadros 2.2 y 2.4) indicando cuántas webs están parametrizadas (ver el apartado siguiente 2.6.1). Se debe fijar a 0 si no hay webs parametrizadas.

2.6.1. Parametrización de la web móvil

Las webs enlazadas en el texto de los SMS enviados se pueden parámetrizar 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 (2.6). Además hay que satisfacer los siguientes **requisitos** adicionales:

- Definir el parámetro "params" (ver los cuadros 2.2 y 2.4) para indicar cuántas de las webs móviles enlazadas en el texto del SMS van a ser parametrizadas.
- Añadir por cada web móvil parametrizada un elemento al parámetro "webParams". Cada elemento consta de los siguientes subelementos:
  - id: identificador de la web móvil.
  - params: elemento a su vez compuesto de 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.

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

```
{
"credentials":{"domainId":"XXXXX","login":"YYYYY","passwd":"ZZZZZ"},
"destination":["346XXXXXXXX"],
"message":{"msg":"Estimado cliente, acceda a la promoción
                 en el siguiente enlace: {$15880$}",
"landing":1,
"params":1,
"webParams":[{"id":15880,"params":{"nombre":"Pedro"}}]}
}
```

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

Cada elemento de "webParams" puede contener tantos parámetros como se hayan definido a través del editor web. Basta añadirlos correctamente formateados como pares parámetro/valor:

```
"webParams":[{"id":xxxx,"params":{"parámetro1":"valor1","parámetro2":"valor2", ... }},
 {"id":yyyy,"params":{"parámetro4":"valor4","parámetro5":"valor5", ... }}]
```

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.

2.7. Confirmación de entrega

El servicio de confirmación de entrega, solicitado a través del parámetro "ack" de los recursos 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 las peticiones de envio en la parte "idAck".

Este identificador puede albergar dos tipos de valores:

- Si en la propia petición 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 la correspondiente petición de envío si cumple los criterios de composición explicados en las tablas 2.2, 2.4 y 2.6
- Si en la propia petición 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 mediante **una petición al recurso REST** que debe exponer el cliente para desarrollar este servicio.

El cliente debe suministrar la URL donde se accederá al recurso.

La petición consta de un mensaje JSON codificado en UTF-8 (content-type: "application/json;charset=UTF-8") con un único elemento "notification".

En el cuadro 2.15 se detallan los elementos que a su vez lo componen.

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 xxxxxxxxxx(2) (ver la respuesta al enviar un mesaje concatenado en la sección 2.3.1).
idAck	Identificador que coincidirá con el suministrado por *Abarcando* al cliente en la parte "idAck" de la respuesta a la operación 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 2.15: Lista de datos de la confirmación de entrega

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

El estado "ERROR_100" indica que el mensaje por el momento no ha podido ser entregado al destinatario debido a algún problema en su teléfono móvil. Las causas más comunes son: mala cobertura, buzón de mensajes cortos lleno o teléfono apagado. El mensaje se intentará enviar varias veces con posterioridad durante un tiempo limitado. Si el problema en el teléfono se subsana a tiempo, el mensaje será finalmente entregado, recibiéndose la correspondiente confirmación.

El estado "ERROR_101" indica que el mensaje por el momento no ha podido ser entregado al destinatario debido a algún problema en la red de telefonía móvil del operador. Habitualmente, cuando el operador solvente los problemas, el mensaje será entregado, recibiéndose la correspondiente confirmación.

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

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

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).

Un ejemplo de notificación de entrega correspondiente a uno de los fragmentos del envío del SMS concatenado del ejemplo del recurso sendSms (ver sección 2.3.1) sería este:

```
{"notification": {"destination":"346YYYYYYYY(1)","idAck":"123456789","status":"ENTREGADO"}}
```

Opcionalmente se podría configurar el envío de las notificaciones de entrega mediante la llamada a un servicio web del cliente según la especificación relativa a la confirmación de entrega de la pasarela de servicios web de *Abarcando* para el envío de SMS.

Otra alternativa sería configurar el envío de las notificaciones de entrega mediante peticiones HTTP POST como también se detalla en la especificación relativa a la confirmación de entrega de la pasarela HTTP de *Abarcando* para el envío de SMS.

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

El **recurso del cliente deberá responder** un único contenido simple como por ejemplo la cadena "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:
   - destination=346xxxxxxxx, 346yyyyyyyy.
   - ack=true.
   - idAck=zzzz.
2. La pasarela de *Abarcando* recibe la petición y remite el mensaje a los destinatarios seleccionados. Además la pasarela responde al cliente un status general "000" y los siguientes datos para cada destinatario:

   ```
   status=000; destination=346xxxxxxxx; idAck=zzzz
   status=000; destination=346yyyyyyyy; idAck=zzzz
   ```

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

3. Solo uno de los destinatarios recibe en su teléfono el mensaje corto enviado.
4. El recurso REST del cliente, habilitado para recibir las notificaciones de estado de entrega, es accedido desde la pasarela de *Abarcando* en sendas peticiones con el parámetro "notification" conteniendo:

```
destination=346xxxxxxxx; idAck=zzzz; status=ENTREGADO
destination=346yyyyyyyy; idAck=zzzz; status=NO ENTREGADO
```

5. El recurso del cliente responde la cadena de texto "OK".
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:

```
destination=346xxxxxxxx; idAck=zzzz; status=CERT(http://...)
```

2.8. Recursos de la API de firma de contrato online

A continuación se detallan los recursos disponibles en la API REST para firma de documentos y los elementos que componen el mensaje JSON de la petición. Cada elemento puede ser obligatorio u opcional.

Cabe citar que el nombre de todos los elementos del JSON podrá ser remitido en diferentes formatos:

- Según la notación Java ("domainId").
- Según la notación REST ("domain_id").
- Todo en minúsculas ("domainid").

Las **peticiones** a cada recurso se harán sobre la **URL base** suministrada, **concatenando el "path"** propio de cada recurso.

2.8.1. 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.

La petición 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.

Se trata del recurso con **"path" /certPdfFile**. El JSON está compuesto por los **elementos obligatorios** detallados en el cuadro 2.16.

Nombre	Valores	Composición
credentials	Datos de identificación del usuario suministrados por *Abarcando*	Elementos domainId, login y passwd
document	Datos propios del documento a enviar	Elementos multiSig, destination, email, type, smsOtpSig, emailOtpSig, webSig, ecertSig, manSig, senderId, smsText, emailText, requestDni, customClause y callback

Cuadro 2.16: Parámetros de entrada del recurso certPdfFile

En el cuadro 2.17 se detallan los elementos restantes constituyentes del JSON de la petición.

Nombre	Valor	Obligatorio
domainId	Identificador suministrado por *Abarcando* al cliente. Se puede omitir si el login es un email.	no*
login	Identificador de usuario suministrado por *Abarcando* al cliente.	sí
passwd	Clave del usuario suministrada por *Abarcando* 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 simultanea. 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 2.8.2).	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 *Abarcando* . 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 2.4.1. 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	Lista de claúsulas personalizadas que aparecerán como requisito previo en las condiciones de firma del documento. Cada cláusula es una 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 claúsula respecto de las demás que puedan aparecer (entero positivo que no debe repetirse). El **texto** es el contenido en sí de la claúsula (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 claúsula). Si se agrega el **sufijo *** al título de la claúsula, obliga a que el destinatario la acepte para poder firmar el documento.	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 2.9).	no

Cuadro 2.17: Lista de parámetros para certpdffile

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

- Mediante los elementos "destination/email"sólo si hay un único firmante.
- Mediante el elemento "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 2.8.2).

A continuación se ilustra un ejemplo de cada caso:

Un ejemplo de petición REST al recurso 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 2.9).

sería este:

```
{
"credentials":{"domainId":"XXXXX",
               "login":"YYYYY",
               "passwd":"ZZZZZ"},
"document":{"destination":"346XXXXXXXX",
            "email":"usuario@dominio.com",
            "type":"premium",
            "smsOtpSig":"true",
            "callback":"true"}
}
```

Un ejemplo de petición REST al recurso 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 2.9).

sería este:

```
{
"credentials":{"domainId":"XXXXX",
               "login":"YYYYY",
               "passwd":"ZZZZZ"},
"document":{"multiSig":[
              {"destination":"346XXXXXXXX",
               "email":"usuario1@dominio.com",
               "sequence":1},
              {"destination":"346YYYYYYYY",
               "email":"usuario2@dominio.com",
               "sequence":2}],
            "type":"premium",
            "smsOtpSig":"true",
            "callback":"true"}
}
```

La **respuesta a este servicio** será un mensaje json con el elemento "status" representando un código de estado general de los descritos en apartado 2.8.5

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 recursos para referenciar a ese documento en particular.

Un ejemplo de respuesta exitosa sería el mostrado a continuación:

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

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

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

2.8.2. Posicionamiento de las firmas

El parámetro "multiSig"(ver el cuadro 2.17) 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 destinario 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.

Un ejemplo de definición de la posición de la firma en el elemento "multiSig"para dos destinatarios sería este:

```
"multiSig":[
   {"destination":"346XXXXXXXX","email":"usuario1@dominio.com","sequence":1,
    "sigLocX":0,"sigLocY":50,"sigLocWidth":200,"sigLocHeight":140,"sigLocPage":1},
   {"destination":"346YYYYYYYY","email":"usuario2@dominio.com","sequence":2,
    "sigLocX":0,"sigLocY":50,"sigLocWidth":200,"sigLocHeight":140,"sigLocPage":2}
]
```

2.8.3. Envío del documento PDF

Como se ha detallado en la petición de firma de documento (ver 2.8.1), 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 2.8.5

Por ejemplo en caso de éxito:

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

En deterinados 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}
```

2.8.4. Consulta del estado de firma de un documento PDF

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

Se trata del recurso con **"path" /checkPdfFile**. El JSON está compuesto por los **elementos obligatorios** detallados en el cuadro 2.18.

Nombre	Valores	Composición
credentials	Datos de identificación del usuario suministrados por *Abarcando*	Elementos domainId, login y passwd
query	Datos de la consulta	Elemento id

Cuadro 2.18: Parámetros de entrada del recurso checkPdfFile

En el cuadro 2.19 se detallan los elementos restantes constituyentes del JSON de la petición.

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

Cuadro 2.19: Lista de parámetros para checkpdffile

Un ejemplo de petición REST al recurso checkPdfFile sería:

```
{
"credentials":{"domainId":"XXXXX",
   "login":"YYYYY",
   "passwd":"ZZZZZ"},
"query":{"id":"XXXXXXXX_YYYYYYYYYYYYYYY_ZZZZZZZZ"}
}
```

La **respuesta a este servicio** será un mensaje json con el **elemento "status"** representando un código de estado general de los descritos en apartado 2.8.5

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 2.9 para consultar detalles sobre los posibles estados.
- **files**: lista de ficheros descargables asociados al proceso de firma. Ver la sección 2.9 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 autentificación sería este:

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

2.8.5. Códigos de estado

El cuadro 2.20 presenta la lista de los posibles códigos de estado que podrán aparecer en la respuesta a cada petición del API de firma de documentos (ver 2.8).

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 autentificación
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 2.20: Lista de los códigos de estado

2.9. 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 recurso de firma de documento PDF (ver 2.8.1), 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 recurso de firma de documento PDF (ver 2.8.1) en el elemento "id").

Las **notificaciones del estado de firma** serán enviadas mediante **una petición al recurso REST** que debe exponer el cliente para desarrollar este servicio.

El cliente debe suministrar la URL donde se accederá al recurso.

La petición consta de un mensaje JSON codificado en UTF-8 (content-type: "application/json;charset=UTF-8") con un único elemento "pdfNotification".

En el cuadro 2.21 se detallan los elementos que a su vez lo componen.

A continuación se esquematiza un ejemplo:

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

Dato	Valor
id	Identificador alfanumérico del documento PDF remitido previamente. Emitido como respuesta del recurso de firma de documento PDF (ver 2.8.1).
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 2.21: Lista de datos de la notificación del estado de firma

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.
- **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 del visualización del documento pendiente de firma por parte del destinatario.
- **sentSmsOtp**: evidencia electónica del envío al teléfono móvil del destinatario de un SMS con la clave para la firma OTP del documento.
- **sentEmailOtp**: evidencia electónica 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 2.8.1) y las posibles claúsulas adicionales remitidas (ver el parámetro "customClause" en la sección 2.8.1).
- **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 2.8.1), 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 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.

Otra alternativa sería configurar el envío de las notificaciones mediante peticiones HTTP POST como también se detalla en la especificación relativa a la actualización del estado de firma de la pasarela HTTP de *Abarcando* para la firma de documentos.

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

El **recurso del cliente deberá responder** un único contenido simple como por ejemplo la cadena "OK".

2.10. 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.

2.10.1. Envío de un mensaje en PHP

Ejemplo desarrollado para PHP7. En versiones anteriores es posible que se requiera instalar el soporte para la librería curl.

```
<?php

function AbarcandoSMS($sDestination, $sMessage, $sSenderId, $debug){
if($debug)
  echo 'Enter AbarcandoSMS <br/>';

//URL base de los recursos REST
$baseUrl = 'http://web.smsverificados.net/apirest/ws';

//Se inicia el objeto CUrl
$ch = curl_init($baseUrl.'/sendSms');

//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
$credentials = array(
  //'domainId' => 'XX',
  'login'    => 'YY',
  'passwd'   => 'ZZ'
);

$destinations = explode(',', $sDestination);

$jsonMessage = array(
  'msg' => substr($sMessage,0,160),
  'senderId' => $sSenderId
);

$jsonData = array(
  'credentials' => $credentials,
  'destination' => $destinations,
  'message'     => $jsonMessage
);

//Se construye el mensaje JSON
$jsonDataEncoded = json_encode($jsonData);

//Indicamos que nuestra petición sera Post
```

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

//Se fija el tiempo máximo de espera para conectar con el servidor (5 segundos)
curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, 5);

//Se fija el tiempo máximo de espera de la respuesta del servidor (60 segundos)
curl_setopt($ch, CURLOPT_TIMEOUT, 60);

//Para que la peticion no imprima el resultado como un 'echo' comun
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

//Se añade el JSON al cuerpo de la petición codificado en UTF-8
curl_setopt($ch, CURLOPT_POSTFIELDS, $jsonDataEncoded);

//Se fija el tipo de contenido de la peticion POST
curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-Type: application/json;charset=UTF-8'));

//Se envía la petición y se consigue la respuesta
$response = curl_exec($ch);

$statusCode = curl_getinfo($ch, CURLINFO_HTTP_CODE);

if($debug) {
  //Error en la respuesta del servidor
  if($statusCode != 200){
    echo 'ERROR GENERAL: '.$statusCode;
    echo $response;
  }else{
    //Se procesa la respuesta capturada
    echo 'Código de estado HTTP: '.$statusCode.'<br/>';
    $json_parsed = json_decode($response);
    $status = $json_parsed->status;
    echo 'Código de estado Abarcando-SMSverificados: '.$status.'<br/>';
    if ($status != '000')
      echo 'Error: '.$response.'<br/>';
    else{
      echo 'Cuerpo de la respuesta: <br/>';
      echo 'destails[0][destination]: '.$json_parsed->details[0]->destination.'<br/>';
      echo 'destails[0][status]: '.$json_parsed->details[0]->status.'<br/>';
      echo 'destails[1][destination]: '.$json_parsed->details[1]->destination.'<br/>';
      echo 'destails[1][status]: '.$json_parsed->details[1]->status.'<br/>';
    }
  }
}

//Si ha ocurrido algún error se lanza una excepción
if(curl_errno($ch))
  throw new Exception(curl_error($ch));

return $response;
}


try{
  echo "The function AbarcandoSMS returns: "
```

```
   .AbarcandoSMS('346xxxxxxx,346yyyyyyyy','Mensaje de prueba', '', true);
  //No es posible utilizar el remitente en América pero sí en España y Europa
  //Utilizar esta llamada solo si se cuenta con un remitente autorizado por Abarcando-SMSverificados
  //echo "The function AbarcandoSMS returns: "
  // .AbarcandoSMS('346xxxxxxx,346yyyyyyyy','Mensaje de prueba', 'remitente', true);
}catch(Exception $e){
  echo 'Error: '.$e->getMessage();
}

?>
```

2.10.2. Envío y firma de documento PDF en PHP

Ejemplo desarrollado para PHP7. En versiones anteriores es posible que se requiera instalar el soporte para la librería curl.

```
<?php
function aaCertPdf($destination,$path, $debug){
  if($debug)
  echo 'Enter aaCertPdf <br/>';

  //URL base de los recursos REST
  $baseUrl = 'http://web.smsverificados.net/apirest/ws';

  //Se inicia el objeto CUrl
  $ch = curl_init($baseUrl.'/certPdfFile');

  //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
  $credentials = array(
  //'domainId' => 'XX',
  'login'    => 'YY',
  'passwd'   => 'ZZ'
  );

  $document = array(
  'destination' => $destination,
  'type'    => 'simple',
  'webSig'   => true
  );

  $jsonData = array(
  'credentials' => $credentials,
  'document' => $document,
  );

  //Se construye el mensaje JSON
  $jsonDataEncoded = json_encode($jsonData);

  //Indicamos que nuestra petición sera Post
  curl_setopt($ch, CURLOPT_POST, 1);

  //Se fija el tiempo máximo de espera para conectar con el servidor (5 segundos)
```

```
curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, 5);

//Se fija el tiempo máximo de espera de la respuesta del servidor (60 segundos)
curl_setopt($ch, CURLOPT_TIMEOUT, 60);

//Para que la peticion no imprima el resultado como un 'echo' comun
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

//Se añade el JSON al cuerpo de la petición codificado en UTF-8
curl_setopt($ch, CURLOPT_POSTFIELDS, $jsonDataEncoded);

//Se fija el tipo de contenido de la peticion POST
curl_setopt($ch, CURLOPT_HTTPHEADER, array('Content-Type: application/json;charset=UTF-8'));

//Se envía la petición y se consigue la respuesta
$response = curl_exec($ch);

$statusCode = curl_getinfo($ch, CURLINFO_HTTP_CODE);

if($debug) {
  //Error en la respuesta del servidor
  if($statusCode != 200){
    echo 'ERROR GENERAL: '.$statusCode;
    echo $response;
  }else{
    //Se procesa la respuesta capturada
    echo 'Código de estado HTTP: '.$statusCode.'<br/>';
    $json_parsed = json_decode($response);
    $status = $json_parsed->status;
    echo 'Código de estado Abarcando-SMSverificados: '.$status.'<br/>';
    if ($status != '000')
      echo 'Error: '.$response.'<br/>';
    else{
     echo "Respuesta certPdfFile: ".$response."\n";

     //Si ha ocurrido algún error se lanza una excepción
     if(curl_errno($ch))
       throw new Exception(curl_error($ch));

     curl_close($ch);

     $url = $json_parsed->url;

     //Upload file request
     $ch = curl_init();
     curl_setopt($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($ch, CURLINFO_HTTP_CODE) === 200){
        if (!strstr($httpResponse,"000"))
          echo "\nError upload cert pdf: ".$httpResponse;
        else
          echo "\nResponse upload file: ".$httpResponse;
      }else
        echo "Error upload cert pdf: ".curl_error($ch).'('.curl_errno($ch).')'.$httpResponse;

        //Si ha ocurrido algún error se lanza una excepción
        if(curl_errno($ch))
          throw new Exception(curl_error($ch));

        curl_close($ch);
      }
    }
  }
}

try{
   aaCertPdf('346xxxxxxxx','file.pdf', true);
}catch(Exception $e){
   echo 'Error: '.$e->getMessage();
}

?>
```

2.10.3. 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 construye el mensaje JSON
JsonObject textMessageFilter = new JsonObject();
JsonObject credentials = new JsonObject();
//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
//credentials.addProperty("domainId","XX");
credentials.addProperty("login","YY");
credentials.addProperty("passwd","ZZ");

JsonArray destinations = new JsonArray();
destinations.add(new JsonPrimitive(new String("346xxxxxxx")));
destinations.add(new JsonPrimitive(new String("346yyyyyyy")));

JsonObject textMessage = new JsonObject();
```

```
textMessage.addProperty("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
//textMessage.addProperty("senderId", "remitente");

textMessageFilter.add("credentials", credentials);
textMessageFilter.add("destination", destinations);
textMessageFilter.add("message", textMessage);

//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 base de los recursos REST
String baseUrl = "http://web.smsverificados.net/apirest/ws";
HttpPost request = new HttpPost(baseUrl+"/sendSms");

//Se añade el JSON al cuerpo de la petición codificado en UTF-8
request.setEntity(new StringEntity(textMessageFilter.toString(),"UTF-8"));

//Se fija el tipo de contenido de la peticion POST
request.addHeader("content-type", "application/json;charset=UTF-8");

CloseableHttpResponse response = null;

try {
  System.out.println("Enviando petición");
  //Se envía la petición
  response = httpClient.execute(request);
  //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 y el content-type");
   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
  request.releaseConnection();
  if(response!=null) {
   try {
     response.close();
   }
   catch(IOException ioe) {
     System.out.println("ERROR cerrando recursos");
   }
  }
}
```

2.10.4. 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 construye el mensaje JSON
JsonObject certPdfFilter = new JsonObject();

// 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
JsonObject credentials = new JsonObject();

// credentials.addProperty("domainId","XX");
credentials.addProperty("login", "YY");
credentials.addProperty("passwd", "ZZ");

JsonObject document = new JsonObject();
document.addProperty("destination", "346xxxxxxxx");
document.addProperty("type", "simple");
document.addProperty("webSig", true);
```

```
certPdfFilter.add("credentials", credentials);
certPdfFilter.add("document", document);

// 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
String baseUrl = "http://web.smsverificados.net/apirest/ws";
HttpPost request = new HttpPost(baseUrl + "/certPdfFile");

// Se añade el JSON al cuerpo de la petición codificado en UTF-8
request.setEntity(new StringEntity(certPdfFilter.toString(), "UTF-8"));

// Se fija el tipo de contenido de la peticion POST
request.addHeader("content-type", "application/json;charset=UTF-8");

CloseableHttpResponse response = null;
InputStream is = null;

try {
  System.out.println("Enviando petición");
  // Se envía la petición
  response = httpClient.execute(request);
  // 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 y el content-type");
return;
  } else {
     if (response != null) {
      response.close();
      response = null;
     }
     // Se procesa la respuesta capturada en la cadena 'response'
    Gson gson = new GsonBuilder().create();
    TreeMap<String, String> map = gson.fromJson(resp, TreeMap.class);

    String status = map.get("status");
    if (!status.equals("000"))
       System.out.println("ERROR: código de estado: " + status);
    else {
```

```
        String url = map.get("url");

        File file = new File("file.pdf");
        is = new FileInputStream(file);

        request = new HttpPost(url);
        request.setEntity(new ByteArrayEntity(IOUtils.toByteArray(is)));
        request.setHeader("Content-type", "application/pdf");

        System.out.println("Subiendo fichero");
        // Se envía la petición
        response = httpClient.execute(request);
        // Se consigue la respuesta
        resp = EntityUtils.toString(response.getEntity());

        if (response.getStatusLine().getStatusCode() != 200)
           System.out.println("ERROR AL SUBIR EL FICHERO: Código de error HTTP:  "
            + response.getStatusLine().getStatusCode());
        else {
           // Se procesa la respuesta capturada en la cadena 'response'
          gson = new GsonBuilder().create();
          map = gson.fromJson(resp, TreeMap.class);
          status = map.get("status");
          if (status.equals("000"))
           System.out.println("Fichero subido correctamente: " + resp);
          else
           System.out.println("ERROR: Error al subir el fichero: " + resp);
       }
    }
   }
} catch (Exception e) {
   System.out.println("Excepción");
   e.printStackTrace();
   return;
} finally {
   // En cualquier caso se cierra la conexión
   request.releaseConnection();
   if (response != null) {
    try {
      response.close();
    } catch (IOException ioe) {
      System.out.println("ERROR cerrando recursos");
    }
   }
   if (is != null) {
    try {
      is.close();
    } catch (IOException ioe) {
      System.out.println("ERROR cerrando recursos");
    }
  }
}
```

2.10.5. Envío de un mensaje en Python

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

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

import requests
import json as JSON

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

try:

  #Se fija la URL base de los recursos REST
  baseUrl = 'http://web.smsverificados.net/apirest/ws'

  #Se construye el mensaje JSON
  #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
  credentials = {'login': 'YY', 'passwd': 'ZZ'}
  #credentials = {'domainId': 'XX', 'login': 'YY', 'passwd': 'ZZ'}
  destination = destinations.split(",")

  messageData = {'msg': message, 'senderId': senderId}
  jsonData = {'credentials': credentials, 'destination': destination, 'message': messageData}

  #Se fija el tipo de contenido de la peticion POST
  contentType = {'Content-Type':'application/json;charset=UTF-8'}
  #Se añade el JSON al cuerpo de la petición
  #Se envía la petición y se recupera la respuesta
  r = requests.post(baseUrl+'/sendSms',
   data=JSON.dumps(jsonData),
   headers=contentType,
   #Se fija el tiempo máximo de espera para conectar con el servidor (5 seg.)
   #Se fija el tiempo máximo de espera de la respuesta del servidor (60 seg)
   timeout=(5, 60)) #timeout(timeout_connect, timeout_read)

  if debug:
   #Error en la respuesta del servidor
   if str(r.status_code) != '200':
    print 'ERROR GENERAL: '+str(r.status_code)
    print r.text
   else:
    #Se procesa la respuesta capturada
    print 'Código de estado HTTP: '+str(r.status_code)
    jsonParsed = JSON.loads(r.text)
    status = str(jsonParsed['status'])
    print 'Código de estado Abarcando-SMSverificados: '+status
    if status != '000':
     print 'Error: '+r.text
    else:
     print 'Cuerpo de la respuesta:'
     print "details[0]['destination']: "+str(jsonParsed['details'][0]['destination'])
```

```
    print "details[0]['status']: "+str(jsonParsed['details'][0]['status'])
    print "details[1]['destination']: "+str(jsonParsed['details'][1]['destination'])
    print "details[1]['status']: "+str(jsonParsed['details'][1]['status'])

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

2.10.6. Envío y firma de documento PDF en Python

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

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

import requests
import json as JSON

def aaCert(destination, fType):
print 'Enter aaCert: destination='+destination+', type: '+fType
try:
#Se fija la URL base de los recursos REST
baseUrl = 'http://web.smsverificados.net/apirest/ws'

#Se construye el mensaje JSON
#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
#credentials = {'domainId': 'XX', 'login': 'YY', 'passwd': 'ZZ'}
credentials = {'login': 'YY', 'passwd': 'ZZ'}

document = {'destination': destination, 'type': fType, 'webSig': True}

jsonData = {'credentials': credentials, 'document': document}

#Se fija el tipo de contenido de la peticion POST
contentType = {'Content-Type':'application/json;charset=UTF-8'}

#Se añade el JSON al cuerpo de la petición
#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(timeout_connect, timeout_read)
```

```
#Se envía la petición y se recupera la respuesta
r = requests.post(baseUrl+'/certPdfFile',
  data=JSON.dumps(jsonData),
  headers=contentType,
  timeout=(5, 60))

#Error en la respuesta del servidor
if str(r.status_code) != '200':
  print 'ERROR GENERAL: '+str(r.status_code)
  print r.text
else:
  #Se procesa la respuesta capturada
  print 'Codigo de estado HTTP cmd: '+str(r.status_code)
  jsonParsed = JSON.loads(r.text)
  status = str(jsonParsed['status'])
  print 'Codigo de estado Abarcando-SMSverificados: '+status
  if status != '000':
   print 'Error: '+r.text
  else:
   print 'Respuesta cmd: '+str(r.text)
   f = open("file.pdf", "rb")
   try:
    contentType = {'Content-Type':'application/pdf'}
    r = requests.post(str(jsonParsed['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
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('346xxxxxxx','simple')
```

2.10.7. Envío de un mensaje en Ruby

Ejemplo en Ruby de cliente REST.

```
# 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
#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
credentials = {:login => "YY", :passwd => "ZZ"}
#credentials = {:domainId => "XX", :login => "YY", :passwd => "ZZ"}
destination = destinations.split(",")

messageData = {:msg => message, :senderId => senderId}
#Se construye el mensaje JSON
jsonData = {:credentials => credentials, :destination => destination, :message => messageData}

#Se fija la URL base de los recursos REST
baseUrl = 'http://web.smsverificados.net/apirest/ws'
uri = URI.parse(baseUrl+"/sendSms")
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 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/json;charset=UTF-8')
request.body = jsonData.to_json

#Se consigue la respuesta
response = http.request(request)

if debug
  #Error en la respuesta del servidor
  unless response.code == "200"
  puts("ERROR GENERAL: #{response.code}")
  puts("#{response.body}")
else
#Se procesa la respuesta capturada
  puts("Código de estado HTTP: #{response.code}")
  jsonResponse = JSON.parse(response.body)
  puts("Código de estado de Abarcando-SMSverificados: #{jsonResponse['status']}")
  unless jsonResponse['status'].include? "000"
```

```
   puts("Error de Abarcando SMSverificados: #{response.body}")
  else
   puts("Cuerpo de la respuesta:")
   puts("details[0]destination: #{jsonResponse['details'][0]['destination']}")
   puts("details[0]status: #{jsonResponse['details'][0]['status']}")
   puts("details[1]destination: #{jsonResponse['details'][1]['destination']}")
   puts("details[1]status: #{jsonResponse['details'][1]['status']}")
  end
end
end

return response

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

end

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

2.10.8. Envío y firma de documento PDF en Ruby

Ejemplo en Ruby de cliente REST.

```
# encoding: utf8

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

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

begin
#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
credentials = {
#:domainId => "XX",
 :login => "YY", :passwd => "ZZ"}
document = {:destination => destination, :type => type, :webSig => true}

#Se construye el mensaje JSON
jsonData = {:credentials => credentials, :document => document}
```

```
#Se fija la URL base de los recursos REST
baseUrl = 'http://web.smsverificados.net/apirest/ws'
uri = URI.parse(baseUrl+"/certPdfFile")
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 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/json;charset=UTF-8')
request.body = jsonData.to_json

#Se consigue la respuesta
response = http.request(request)

#Error en la respuesta del servidor
unless response.code == "200"
  puts("ERROR GENERAL: #{response.code}")
  puts("#{response.body}")
else
  #Se procesa la respuesta capturada
  puts("Código de estado HTTP: #{response.code}")
  parsedJson = JSON.parse(response.body)
  puts("Código de estado de Abarcando-SMSverificados cmd: #{parsedJson['status']}")
  unless parsedJson['status'].include? "000"
    puts("Error de Abarcando SMSverificados: #{response.body}")
  else
    file = File.open("file.pdf", "rb")

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

2.10.9. Envío de un mensaje en Perl

Ejemplo en Perl (ver [PERL]) de cliente REST usando LWP::UserAgent.

```
#!/usr/bin/perl

require HTTP::Request;
require LWP::UserAgent;
use JSON;
use strict;
use warnings;

binmode(STDOUT, ":utf8");

sub abarcandoSms{

if ($_[3] eq 'true'){
    print 'Enter abarcandoSms: destinations='.$_[0].', message='.$_[1].', senderId='.$_[2]."\n";
}

try {
#Se fija la URL base de los recursos REST
my $base_url = 'http://web.smsverificados.net/apirest/ws';

my @array = split(',', $_[0]);

my $json = JSON->new->utf8;

#Se construye el mensaje JSON
#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 $data_to_json = {credentials=>{login=>'YY',passwd=>'ZZ'},
#credentials=>{domainId=>'XX',login=>'YY',passwd=>'ZZ'},
destination=> \@array,
message=>{msg=>$_[1],senderId=>$_[2]}
```

```
};

#Se inicia el objeto HTTP
my $request = HTTP::Request->new( 'POST', $base_url.'/sendSms' );
#Se añade el JSON al cuerpo de la petición codificado en UTF-8
$request->header( 'Content-Type' => 'application/json;charset=UTF-8' );
#Se añade el JSON a la petición
$request->content( $json->encode($data_to_json) );

my $lwp = LWP::UserAgent->new;
#Se fija el tiempo máximo de espera de la respuesta del servidor (60 segundos)
$lwp->timeout(60);
#Se consigue la respuesta
my $response = $lwp->request( $request );

my $message = $response->decoded_content;

#El cliente HTTP en caso de timeout devuelve un estado 500 con su correspondiente mensaje
if(index($message, 'timeout') != -1){
  die 'Timeout error'; #Se lanza una excepción
}

if ($_[3] eq 'true'){
  if ( $response->is_success ) {
   print "Código de estado HTTP: ".$response->code."\n";
   my $decoded = decode_json($message);
   my $abarcando_status = $decoded->{'status'};
   print "Código de estado de Abarcando-SMSverificados: " .$abarcando_status. "\n";

   if($abarcando_status eq '000'){#Se procesa la respuesta capturada
     print "Cuerpo de la respuesta: \n";
     print "details[0]destination: ".$decoded->{'details'}[0]{'destination'}."\n";
     print "details[0]status: ".$decoded->{'details'}[0]{'status'}."\n";
     print "details[1]destination: ".$decoded->{'details'}[1]{'destination'}."\n";
     print "details[1]status: ".$decoded->{'details'}[1]{'status'}."\n";
   }else{#Error en la respuesta del servidor
     print "Error de Abarcando SMSverificados: ".$message."\n";
   }
 } else {
print "ERROR GENERAL: ".$response->code."\n";
print $message."\n";
}
}
return $message;
};
catch {
  print "Error: ".$@->what."\n";
};
}

print "The function abarcandoSms returns: "
.abarcandoSms('346xxxxxxxx,346yyyyyyyy','Mensaje de prueba', '', 'true')."\n";
#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: "
```

```
# .abarcandoSms('346xxxxxxx,346yyyyyyyy','Mensaje de prueba', 'remitente', 'true')."\n";
```

2.10.10. Envío y firma de documento PDF en Perl

Ejemplo en Perl (ver [PERL]) de cliente REST usando LWP::UserAgent.

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

```
#!/usr/bin/perl

use strict;
use warnings;
use HTTP::Request;
use LWP::UserAgent;
use JSON;
use JSON::Parse 'parse_json';
use Try::Tiny;

binmode(STDOUT, ":utf8");

try {
  #Se fija la URL base de los recursos REST
  my $base_url = 'http://web.smsverificados.net/apirest/ws';

  my $json = JSON->new->utf8;

  #Se construye el mensaje JSON
  #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_to_json = {credentials=>{
  #domainId=>'XX',
  login=>'YY',passwd=>'ZZ'},
  document=>{destination=>'346xxxxxxx',type=>'simple',webSig=>'true'}};

  #Se inicia el objeto HTTP
  my $req = HTTP::Request->new( 'POST', $base_url.'/certPdfFile' );
  #Se añade el JSON al cuerpo de la petición codificado en UTF-8
  $req->header( 'Content-Type' => 'application/json;charset=UTF-8' );
  #Se añade el JSON a la petición
  $req->content( $json->encode($data_to_json) );

  my $ua = LWP::UserAgent->new;
  #Se fija el tiempo máximo de espera de la respuesta del servidor (60 segundos)
  $ua->timeout(60);
  #Se consigue la respuesta
  my $resp = $ua->request($req);

  my $message = $resp->decoded_content;

  my $parsedJson = parse_json ($message);

  my $status = $parsedJson->{status};

  if ($status != '000') {
   print "\nERROR COMANDO CERTPDFFILE: Código Abarcando-SMSverificados: \n$status\n";
```

```
  }else {
   print "Comando procesado correctamente: \n".$message."\n";
   print "\nSubiendo fichero\n";

   # Se fija la URL sobre la que enviar la petición POST

   $req = HTTP::Request->new( 'POST',$parsedJson->{url});
   $req->header('content-type'=>'application/pdf');

   # Se lee el fichero y se almacena en un buffer
   my $fname = 'file.pdf';
   my $BLOCK_SIZE=1024;
   open(F,"<$fname") or die("Fichero no encontrado: $fname, $!");
   binmode(F);
   my $buf;
   my $ct=0;
   while(read(F,$buf,$BLOCK_SIZE,$ct*$BLOCK_SIZE)){
    $ct++;
  }
close(F);

$req->content($buf);
$resp = $ua->request($req);
if ($resp->is_success) {
 $message = $resp->decoded_content;
 print "\nRespuesta al subir certificado: \n$message\n";
 $parsedJson = parse_json ($message);
 $status = $parsedJson->{status};
 if ($status != '000') {
  print "\nERROR AL SUBIR FICHERO: Código Abarcando-SMSverificados: \n$status\n";
 }else {
  print "\nProceso terminado con exito.\n";
 }
}else {
 print "HTTP POST error al subir el fichero. Code: ", $resp->code, "\n";
 print $resp->decoded_content;
}
}
}catch { #Se capturan posibles errores en el lado cliente
  print "Error inesperado: $_\n";
};
```

2.10.11. 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 REST

```
using System;
using System.Text;
using System.Net;
using System.IO;

namespace RestVCAbarcando {
```

```
class Program {
static void Main(string[] args) {
  var response = "";
  var error = "";
  try {
   //Se fija la URL sobre la que enviar la petición POST
   var httpWebRequest = (HttpWebRequest)WebRequest.
    Create("http://web.smsverificados.net/apirest/ws/sendSms");
   httpWebRequest.ContentType = "application/json";
   httpWebRequest.Method = "POST";
   //Establecemos el TimeOut para obtener la respuesta del servidor
   httpWebRequest.Timeout = 60000;

   //domainId solo es necesario si el login no es un email
   using (var streamWriter = new StreamWriter(httpWebRequest.GetRequestStream())) {
    string json = "{\"credentials\":{"
                  //+ \"domainId\":\"XX\","
                    + "\"login\":\"YY\",\"passwd\":\"ZZ\"},";
    json += " \"destination\":[\"346xxxxxxx\",\"346yyyyyyyy\"],";
    //No es posible utilizar el remitente en América pero si en España y Europa
    //Descomentar la linea solo si se cuenta con un remitente autorizado por Abarcando-SMSverificados
    //json += " \"message\": {\"msg\":\"Mensaje de prueba\","
    //        + "\"senderId\":\"remitente\"}}";
    json += " \"message\": {\"msg\":\"Mensaje de prueba\"}}";
    streamWriter.Write(json);
    streamWriter.Flush();
    streamWriter.Close();
}

var httpResponse = (HttpWebResponse)httpWebRequest.GetResponse();
using (var streamReader = new StreamReader(httpResponse.GetResponseStream())) {
  response = streamReader.ReadToEnd();
}

  }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 Basic usando HttpWebRequest 4.0 como cliente REST

```
Imports System.IO
Imports System.Net
```

```
Imports System.Text

Module RestVBAbarcando

  Sub Main()

Dim err = ""
Dim resp = ""
Try
  '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 json = "{""credentials"":{""login"":""YY"",""passwd"":""ZZ""},"
  //Dim json = "{""credentials"":{""domainId"":""XX"",""login"":""YY"",""passwd"":""ZZ""},"
  json += " ""destination"":[""346xxxxxxx"",""346yyyyyyyy""],"
  '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
  'json += " ""message"": {""msg"":""Mensaje de prueba"",""senderId"":""remitente""}}"
  json += " ""message"": {""msg"":""Mensaje de prueba""}}"

  Dim jsonDataBytes = Encoding.UTF8.GetBytes(json)

  Dim req As WebRequest = WebRequest.Create("http://web.smsverificados.net/apirest/ws/sendSms")
  req.ContentType = "application/json"
  req.Method = "POST"
  req.ContentLength = jsonDataBytes.Length
  'Fijamos TimeOut de espera de respuesta del servidor = 60 seg
  req.Timeout = 60000

  Dim stream = req.GetRequestStream()
  stream.Write(jsonDataBytes, 0, jsonDataBytes.Length)
  stream.Close()

  Dim response = req.GetResponse().GetResponseStream()

  Dim reader As New StreamReader(response)
  'Conseguimos la respuesta en una cadena de texto
  resp = reader.ReadToEnd()
  reader.Close()
  response.Close()
Catch e1 As Exception
  err = e1.Message
Finally
  If (err <> "") Then
Console.WriteLine(err)
  Else
Console.WriteLine(resp)
  End If
End Try
  End Sub
End Module
```

2.10.12. Envío y firma de documento PDF en curl

Se adjuntan ejemplos de uso de curl a través de scripts de shell

Solicitud de firma

```
#!/bin/sh

#Url del servicio certpdf
url="http://web.smsverificados.net/apirest/ws/certPdfFile"


#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

generate_post_data() {
  cat <<EOF
{
  "credentials": {
  "domainId": "XX",
  "login": "YY",
  "passwd": "ZZ"
  },
  "document": {
   "destination": "346xxxxxxxx",
   "type": "premium",
   "webSig": true
  }
}
EOF
}


response=$(curl -i \
-H "Accept: application/json" \
-H "Content-Type:application/json" \
-X POST --data "$(generate_post_data)" "$url")

echo "Respuesta al servicio certpdf: $response"

exit 0
```

Subida del fichero

```
#!/bin/sh

#$1 Parametro de entrada: url obtenida como respuesta al servicio "certpdf" en el elemento "url"

#Ruta local al fichero a subir para su firma
path="/tmp/fichero.pdf"

response=$(curl -i \
-H "Content-Type:application/pdf" \
```

```
-X POST --data-binary @"$path" "$1")

echo "Respuesta al servicio uploadfile: $response"

exit 0
```

Consulta de estado

```
#!/bin/sh

#Url del servicio checkpdf
url="http://web.smsverificados.net/apirest/ws/checkPdfFile"

#$1 Parametro de entrada: identificador obtenido como respuesta
#al servicio "certpdf" en el elemento "id"

#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

generate_post_data="{\"credentials\":{\"domainId\":\"XX\",\"login\":\"YY\",\"passwd\":\"ZZ\"},
 \"query\":{\"id\":\"$1\"}}"

#echo $generate_post_data

response=$(curl -i \
-H "Accept: application/json" \
-H "Content-Type:application/json" \
-X POST --data `echo $generate_post_data` "$url")

echo "Respuesta al servicio checkpdf: $response"

exit 0
```

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/nusoap/

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

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

[CODIGOFUENTE] *Enlace a la descarga del código fuente de los ejemplos:*
https://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://savonrb.com

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

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

[POINTS2MM] *Conversión de puntos a milímetros:*
https://www.google.com/search?q=point+to+mm