API PHP

LleidaNetworks Serveis Telemàtics, S.L.
devel@lleida.net

1 de febrero de 2012

Índice

Copyright

1. Descripción de la API PHP

Es una librería para la creación de páginas web con PHP 5 o superior con funciones de mensajería SMS/MMS a través de la Pasarela Avanzada de Mensajería (PAM) de Lleida Networks Serveis Telemàtics, SL.

Incluye la implementación del protocolo de comunicaciones de SMS MASIVO, para el envío masivo de mensajes de texto SMS y MMS, así como enlaces WAP y contenidos WAP-PUSH.

Esta API está disponible gratuitamente en la página web oficial del servicio:

http://soporte.lleida.net/

2. Requerimientos e instalación

La API funciona correctamente con la versión PHP 5 o superior.

Requiere el uso de SOCKETS y conexión a la pasarela de mensajería de Lleida.net (normalmente el host es **sms.lleida.net** y el puerto tcp 2048) por lo que deberemos consultar al administrador del servidor Web o proveedor de servicios de Hosting en caso de que la conexión esté limitada por algún tipo de cortafuegos.

Para el envío de contenidos multimedia (imágenes, sonidos, etc.), los ficheros deben tener los permisos necesarios para abrir los ficheros por el usuario que ejecuta el script, normalmente **apache** o **www-data**.

Es imprescindible disponer de una cuenta con saldo para el envío de mensajes.

Puede obtener una cuenta de usuario de forma totalmente gratuita a través de la página web de Lleida.net o llamando al teléfono de atención al cliente 902 999 272.

2.1. Recomendaciones

Si su dominio esta alojado en un servidor compartido y no tiene privilegios para modificar las opciones del Apache o PHP le recomendamos que se mire los ejemplos de implementación en PHP de la API XML HTTP que encontrara en soporte.

http://soporte.lleida.net/?p=211.

3. Descripción de las clases

La API requiere de los siguientes ficheros:

- lib/constants.php
- lib/socket.php
- lib/socket-php4.php
- lib/socket-php5.php
- lib/socket-const.php
- lib/logger.php
- lib/protocolproperties.php
- lib/userproperties.php
- lib/bulksend.php
- lib/savecommand.php

Todos ellos, excepto el lib/constants.php, están implementados como clases. Cada uno de estos ficheros se encarga de una tarea en concreto y no hace falta que se editen ni que se modifiquen.

3.1. Events

```
class Events {
    // Constructura estilo PHP4
    function Events(){}

    // Eventos relacionados con el Socket y la session del usuario
    function noCredit(){}
    function connected(){}
    function disconnected(){}
    function connectionError($errCode, $errMsg){}

    // Eventos de repuesta a los comandos de envio
    function reply($idEnvio, $status, $data=null){}

    // Eventos para los ACUSES
    function deliveryReceipt($timeStamp, $sendTimeStamp,
        $recipient, $text, $status, $id=''){}

    function deliveryReceiptMMS($timeStamp, $sendTimeStamp,
        $recipient, $status, $id=''){}

    // Eventos para los mensajes entrantes
    function incomingMo($idIncomingMo, $timeStamp, $sender,
        $recipient, $text){}

    function incomingMMS($idIncomingMMS, $timeStamp, $sender,
        $recipient){}

    // Eventos relacionados con los CHECKER
    function serviceUnavailable(){}
    function checkall($id, $timeStamp, $num, $ok, $rcode, $mcc,
        $mnc, $spid, $cc){}

    function checknetwork($id, $timeStamp, $num, $ok, $rcode, $mcc,
        $mnc, $spid, $cc){}

    function operatorInfo($phoneOrPrefix, $MCC, $MNC){}

    // Evento para obtener la tarifa segun el numero de destino
    function price($price){}
}
```

El primer paso que realizar es extender la clase **Events** con su propia implementación. Junto con la API se han añadido dos ejemplos el sample y el samplePDO. Estos ejemplos extienden la clase Events.

La clase VirtualSMS es la responsable de lanzar los eventos al recibir las respuestas del servidor. Por ejemplo, al llamar al VirtualSMS->connect(), y si todo va bien, saltara el evento connected(). Si hay algún error saltara el evento connectionError().

```
function noCredit(){}
```

Se produce cuando se detecta que el saldo es inferior a 1 crédito.

```
function connected(){}
```

Se produce cuando se ha establecido la conexión con el servidor y el usuario se ha identificado correctamente en el sistema.

Nuestros SMSC's son de sesión única, es decir no permiten tener sesiones duplicadas.

```
function disconnected(){}
```

Se produce cuando se cierra la conexión con el servidor al llamar al método VirtualSMS->disconnect(). La API recibe el comando BYE y se cierra el Socket.

```
function connectionError($errCode, $errMsg){}
```

Se produce cuando se pierde la conexión con el servidor. El Socket esta cerrado o no se puede leer ni escribir en él. Los posibles valores del parámetro $errCode son:

```
SMSMASS_CONN_ERROR
SMSMASS_ERR_NO_HOST_FOUND
SMSMASS_ERR_CANT_SEND_DATA
SMSMASS_NOOK_NOT_CONNECTED_TO_HOST

SMSMASS_ERR_PING_TIMEOUT

SMSMASS_ERR_ALREADY_LOGGED
SMSMASS_ERR_INVALID_USER_OR_PASSWORD
SMSMASS_ERR_NO_USER_OR_PASSWORD_FOUND
```

```
function reply($idEnvio, $status, $data=null){}
```

Se produce cuando se recibe el estado preliminar de un mensaje tras ser enviado. Es decir, hemos recibido su solucitud de envio. Debe tener claro que esto no significa que el mensaje haya sido entregado en destino. Los posibles valores del parámetro $status son:

```
SMSMASS_OK
SMSMASS_OK_ANY_RECIPIENT_INVALID

SMSMASS_NOOK_GENERIC_ERROR
SMSMASS_NOOK_INSUFFICIENT_CREDIT
SMSMASS_NOOK_INVALID_DATE
SMSMASS_NOOK_ALL_RECIPIENT_INVALID
SMSMASS_NOOK_ANY_OR_ALL_TEXT_NOT_FOUND
SMSMASS_NOOK_ANY_OR_ALL_RECIPIENTS_NOT_FOUND
SMSMASS_NOOK_NUMBER_OF_TEXT_AND_RECIPIENTS_NOT_MATCH

SMSMASS_NOOK_ABNORMAL_ERROR
SMSMASS_NOOK_INVALID_TRANS
```

En estos otros casos además de lanzar el evento el método de envío devuelve el valor de la constante.

```
SMSMASS_NOOK_NOT_CONNECTED_TO_HOST
SMSMASS_NOOK_TEXT_NOT_FOUND
SMSMASS_NOOK_RECIPIENTS_NOT_FOUND
SMSMASS_NOOK_DATA_NOT_FOUND
SMSMASS_NOOK_SUBJECT_NOT_FOUND
SMSMASS_NOOK_ANY_OR_ALL_URL_NOT_FOUND
SMSMASS_NOOK_NUMBER_OF_URL_AND_RECIPIENTS_NOT_MATCH
SMSMASS_NOOK_MIMETYPE_OR_DATA_NOT_FOUND
```

Aunque siempre es preferible que traten el evento.

Si implementan el servicio Checker con identificador también se lanzara este evento al recibir el comando CHECKALLOK o CHECKNETWORKOK. Es decir, hemos recibido su solicitud y vamos a procesar el check. Cuando el servidor tenga la respuesta de la operadora se lanzara el evento checkall o checknetwork correspondiente con los valores de la consulta.

Pude encontrar mas información en la web de soporte o en nuestro

webchecker.lleida.net.

```
function deliveryReceipt($timeStamp, $sendTimeStamp, $recipient,
    $text, $status, $id='){}

function deliveryReceiptMMS($timeStamp, $sendTimeStamp,
    $recipient, $status, $id='){}
```

Se produce cuando se recibe un acuse de recibo. Suele ser un estado final del SMS/MMS. Para que se lance este evento es necesario establecer la propiedad:

VirtualSMS->acuseOn('INTERNAL');

también se puede activar para que los envíos sean certificados:

VirtualSMS->acuseOnCertifiedSMS('INTERNAL');

Recuerde que en este caso deberán implementar la recogida de los PDF. Pueden encontrar un ejemplo en PHP en soporte

http://soporte.lleida.net/?p=644.

Los posibles valores del parámetro $status son:

```
// Entregado al destino
SMSMASS_DEL_DELIVERED

// Estados fallidos
SMSMASS_DEL_ACKED
SMSMASS_DEL_BUFFERED
SMSMASS_DEL_FAILED
SMSMASS_DEL_DELETED
SMSMASS_DEL_UNKNOWN
```

El parámetro $id se corresponde al parámetro $idEnvio de los métodos de envíos. También se corresponde con el parámetro $idEnvio del evento reply.

```
function incomingMo($idIncomingMo, $timeStamp, $sender,
    $recipient, $text)
```

Se produce cuando se recibe un mensaje de texto.

```
function incomingMMS($idIncomingMMS, $timeStamp, $sender,
    $recipient) {}
```

Se produce cuando se recibe un MMS. Puede usar la API XML MMS para obtener el contenido del MMS.
http://soporte.lleida.net/?p=568.

```
function serviceUnavailable(){}
```

Se produce cuando el servicio Checker no esta disponible. Solo se lanza si realiza un checker.

```
function checkall($id, $timeStamp, $num, $ok, $rcode, $mcc, $mnc,
    $spid, $cc){}
```

Se produce cuando se recibe la respuesta del método:
VirtualSMS->checkall($num);
El parámetro $id es un identificador único de Lleida.net y no lo debe confundir con el parámetro $id del método VirtualSMS->checkall(*num*,id).

```
function checknetwork($id, $timeStamp, $num, $ok, $rcode, $mcc,
    $mnc, $spid, $cc){}
```

Se produce cuando se recibe la respuesta del método:
VirtualSMS->checknetwork($num);
El parámetro $id es un identificador único de Lleida.net.

```
function operatorInfo($phoneOrPrefix, $mcc, $mnc){}
```

Se produce cuando se recibe la respuesta del método:
VirtualSMS->getOperatorInfo($num);

Nos indica el código MCC y MNC del número de teléfono o prefijo solicitado. En lugar de este sistema recomendamos realizar un checknetwork ya que este método no verifica la portabilidad del numero.

El significado de las abreviaciones son:

Abbr.	**Significado**
MNC	Mobile Network Code
MCC	Mobile Country Code
SPID	Código de LLeida.net
CC	País de la operadora

En los métodos checker debe centrarse en el parámetro $rCode cuyo significado es:

Código	**Significado**
-2	Número inválido
-1	No se puede resolver el estado del número
0	Número correcto
1	Formato inválido de la consulta
2	Se ha superado el tiempo máximo de espera sin obtener respuesta
3	El número no existe
4	El número no permite el servicio de SMS
5	Llamadas restringidas
6	El terminal del número no esta bien configurado
7	El número esta en roaming o no permite la entrada de SMS
8	Error de la operadora
9	El número esta apagado o fuera de cobertura
11	No tiene saldo para hacer la consulta
12	Error desconocido
13	Pendiente de validación

3.2. VirtualSMS

```
function VirtualSMS($user, $password) {}
```

La constructora de la clase genera los valores por defecto para las propiedades del protocolo y del usuario. Crea una instancia de la clase Events para que no haya errores de ejecución.

```
function setEvents($e) {}
function getEvents() {}
```

Si el parametro $e no extiende la clase Events se ignora.

```
function setProtocolProperties($prop) {}
function getProtocolProperties() {}
```

Las propiedades del protocolo se detallan mas adelante. Estos métodos permiten modificar el host, puerto y flags de ack's. Por ejemplo:

```
$this->sms->getProtocolProperties()->setHost('sms2.lleida.net');
$this->sms->connect();
```

```
function setUserProperties($prop) {}
function getUserProperties() {}
```

Las propiedades del usuario se detallan mas adelante. Estos métodos permiten modificar el usuario, contraseña, email y opcionales del certificado. Por ejemplo:

```
$this->sms->getUserProperties()->setCustomizedSender('Lleidanet');
$this->sms->connect();
```

```
function connect() {}
```

El método VirtualSMS->connect() puede devolver estos valores:

```
// Si se ha establecido la conexión con el servidor
SMSMASS_CONN_CONNECTED

// Posibles errores que puede devolver
SMSMASS_ERR_NO_USER_OR_PASSWORD_FOUND
SMSMASS_ERR_NO_HOST_FOUND
SMSMASS_CONN_ERROR
```

```
function disconnect() {}
```

El método VirtualSMS->disconnect() activa el flag VirtualSMS->isQuit que desactiva la posibilidad de enviar mas comandos por el Socket. El ultimo comando que envía es el QUIT y espera a recibir el BYE. Cuando se recibe el BYE se desconecta y se destruye el Socket.

```
function isLogged() {}
```

El método VirtualSMS->isLogged() verifica que el exista el Socket, este conectado y no se haya enviado el QUIT.

```
function acuseOff() {}
function acuseOn($mail = '') {}
function acuseOnCertifiedSMS($mail = '', $lang = 'X', $type = 'X',
    $name = 'X', $nid = 'X') {}
```

Los métodos de acuse requieren estar conectado ya que envian por Socket los comandos de configuración.

La API dispone de un flag para activar/desactivar la gestión de los acuses de forma global. Es decir, si desactivan los acuses la API no enviara ningún comando de ACUSEACK. Para mas detalles pueden consultar la documentación del protocolo.

```
// Desactiva los ACK's para los ACUSE
$this->sms->getProtocolProperties()->setAcusesAck('false');
```

En los metodos VirtualSMS->acuseOn y VirtualSMS->acuseOnCertifiedSMS todos los parametros son opcionales ya que se usan las propiedades del usuario.

Debe tener en cuenta que al poner valores en los parametros automaticamente se actualizaran los valores de las propiedades de usuario. Por ejemplo:

```
$email = $this->sms->getUserProperties()->getMailDeliveryReceipt();
// $email = 'INTERNALID'

$lang = $this->sms->getUserProperties()->getLang();
// $lang = 'ES'

$this->sms->acuseOnCertifiedSMS('noreply@lleida.net', 'CA');

$email = $this->sms->getUserProperties()->getMailDeliveryReceipt();
// $email = 'noreply@lleida.net'

$lang = $this->sms->getUserProperties()->getLang();
// $lang = 'CA'
```

```
function checkall($num, $id='') {}
function checknetwork($num, $id='') {}
function getOperatorInfo($num) {}
```

La API dispone de un flag para activar/desactivar la gestión de los checker de forma global. Es decir, si desactivan los checker la API no enviara ningún comando de RCHECKXXX. Para mas detalles pueden consultar la documentación del protocolo.

```
// Desactiva los ACK's para los CHECKER
$this->sms->getProtocolProperties()->setCheckerAck('false');
```

Si se añade el parámetro opcional $id se lanzara el evento reply al recibir el comando CHECKALLOK o CHECKNETWORKOK. En este caso $status puede devolver:

```
SMSMASS_OK

SMSMASS_NOOK_GENERIC_ERROR
SMSMASS_NOOK_INSUFFICIENT_CREDIT
```

Si se realizan checkers sin $id se lanza el evento checkall/checknetwork con el rcode correspondiente. Sin $id no se lanza el evento reply. Dicho de otro modo, sin un identificador no puede saber si hemos recibido su solicitud de validación.

```
function setAllowAnswer($estat){}
```

Para activar la respuesta al numero largo $estat debe ser TRUE (es el valor por defecto), para desactivar esta funcionalidad debe ser FALSE.

3.2.1. Métodos de envío

```
// SMS
function sendTextSMS($idEnvio, $text, $recipients, $dateTime =
    '') {}
function sendBinarySMS($idEnvio, $data, $recipients, $dateTime
    = '') {}
function sendUnicodeTextSMS($idEnvio, $text, $recipients,
    $dateTime = '') {}

// MMS
function sendMMS($idEnvio, $subject, $text, $recipients,
    $mimeType, $fileData) {}
function sendSMIL($idEnvio, $recipients, $mms) {}
function sendWapPush($idEnvio, $subject, $text, $recipients,
    $mimeType, $fileData) {}
function sendWapLink($idEnvio, $subject, $URL, $recipients) {}
```

Los metodos de envio admiten tipos de datos array para los parametros $recipients, $text y $data.

El parámetro $dateTime es opcional y sirve para programar el envío. El formato debe ser YYYYMMDDHHMM. La API se encarga de añadir la zona horaria del servidor web donde corre.

El parámetro $idEnvio suele ser el ID del registro de una BBDD y no puede contener los caracteres:

```
"\n", "\r", ":", " ", "(" y ")"
```

En los métodos de envío binarios y unicode se debe codificar el parámetro $data en Base64.

```
if (!$this->sms->isLatin1($text)) {
    // Si el texto no esta en latin1 lo envio como unicode
    if (!mb_check_encoding($text, 'UTF-8')){
        $text = utf8_encode($text);
    }
    $this->sms->sendUnicodeTextSMS($idEnvio,
        base64_encode(iconv("UTF-8", "UTF-16", $text)),
        $recipients);
}
```

El parámetro $idEnvio se usa para identificar el envío. Se usara este valor en los eventos que genere la API.

Aunque el parámetro $recipients sea un array se lanzara un único evento reply. Si hay algún destino erróneo del tipo **+3444pf34** el evento **reply** se lanzara con el parámetro opcional $data con un array de los destinos inválidos.

3.3. Properties

Estas clases te permite establecer las propiedades del protocolo y del usuario.

```
class UserProperties {
    var $m_User = '';
    var $m_Pass = '';
    var $m_Credit = 0;
    var $m_MailDeliveryReceipt = 'INTERNALID';
    var $m_CustomizedSender = '';
    var $m_AllowAnswer = true;
    var $m_Lang = 'ES';
    var $m_Type = 'D';
    var $m_CertName = '';
    var $m_CertNameID = '';

    // Y sus correspondientes métodos Setter y Getter.
}
```

La API tiene activas las respuestas a los comandos INCOMING, CHECKER y ACUSE. Para activar o desactivar los comandos que debe de gestionar vuestra aplicación PHP tenéis la clase ProtocolProperties.

```
class ProtocolProperties {
    var $bDebug = true;
    var $m_Host = 'sms.lleida.net';
    var $m_Port = 2048;
    var $m_ackAcuses = true;
    var $m_ackChecker = true;
    var $m_ackIncoming = true;

    // Y sus correspondientes métodos Setter y Getter.
}
```

3.4. Sample

```
class SendMessages extends Events {
       var $sms;
       var $connected = false;

       function __construct() {
           // Configuro los eventos
           parent::Events();

           // Creo el objeto para enviar SMS
           $this->sms = new VirtualSMS(SMS_USER, SMS_PASS);
           $this->sms->setEvents($this);

           // Configuro las propiedades del protocolo
           $this->sms->getProtocolProperties()->setHost('sms2.lleida.net');

           // Configuro las propiedades del usuario
           $this->sms->getUserProperties()->setCustomizedSender('Lleidanet');

           // Conecto
           $this->sms->connect();

           // Configuro los acuses
           $this->sms->acuseOn('INTERNAL');
       }
```

```
        // Funcion para consultar si esta conectado
        function isConnected() {
            // Controlo que la session este conectada y que no haya
                saltado
            // ningun evento de desconexion
            return $this->connected && $this->sms->isLogged();
        }

        function reply($idEnvio, $status, $data=null) {
            parent::reply($idEnvio, $status, $data);
            // Tu codigo va aqui:

        }
        ...
```