API PHP 2.0

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

6 de abril de 2010

Índice

Copyright

1. Descripción de la API PHP 2.0

Es una librería para la creación de páginas web con PHP 4 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 4 o superior.
Requiere el uso de SOCKETS y conexión a la pasarela de mensajería de Lleida.net (normalmente el host es "sms.lleida.netz el puerto "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 (imágenes, sonidos, etc.) multimedia, los ficheros deben tener los permisos necesarios para abrir los ficheros por el usuario que ejecuta el script, normalmente APACHE.

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

Si no dispone de cuenta de usuario puede obtener una cuenta de demostración de forma totalmente gratuita a través de la página web de Lleida.net o llamando al teléfono de atención al cliente: 902 999 272.

http://www.lleida.net/es/alta.html

Existen dos versiones de la API, una completa que captura eventos y permite tratar los IncomingMO, Checkers y Acuses. Y otra mas simple, que esta

limita en funcionalidades.

Ambas API pueden convivir en el mismo árbol de desarrollo, por lo que se pueden tener juntas en el mismo directorio.

2.1. Diferencias básicas

La API Simple, simplesms.php, tan solo requiere de los ficheros:

- lib/constants.php
- lib/socket.php
- lib/socket-php4.php
- lib/socket-php5.php
- lib/socket-const.php
- lib/logger.php

Mientras que la completa , virtualsms.php, requiere, los ficheros anteriores, y además:

- lib/protocolproperties.php
- lib/userproperties.php
- lib/bulksend.php
- events.php

Cada uno de estos ficheros se encarga de una tarea en concreto y no hace falta que se editen ni que se modifiquen.

Todos ellos, excepto el lib/constants.php, están implementados como clases. Para la clase VirtualSMS es interesante que extendáis la clase **events.php** con vuestra propia implementación. Tenéis como ejemplo el fichero tests/myevents.php.

Si se decide sobreescribir el fichero debeis de ir con cuidada en el momento de actualizar la version.

En la API simple solo se configura el parámetro customized sender **setCustumizedSender('CS')**. Para quitarlo debes de establecer el valor a vacío, esto es cadena vacía.

La API simple no reconoce los comandos DELIVER, INCOMINGMO, CHECKER y ACUSE. Esto significa que si la API simple recibe uno de estos comandos lo ignorara y no realiza ninguna acción. Esto es interesante, por que si enviara un ACK de un INCOMING ese usuario nunca sabría de ese mensaje recibido. Se puede extrapolar lo mismo para los DELIVER, CHEKER y ACUSE. Esto significa que ignora los comandos enviados por el servidor y no los trata.

Para la API completa sí están activas las respuestas a estos comandos. Para configurar los comandos que debe de tratar vuestra aplicación PHP tenéis la clase ProtocolProperties.php

Por defecto están desactivados, de esta manera se ahorra en espacio de memoria. Por ejemplo, os puede interesar tener activo los ACUSE, pero la resta de comandos desactivados.

Otra gran diferencia es que la API simple no mantiene en memoria los comandos enviados ni recibidos.

Mientras que la clase VirtualSMS mantiene 3 arrays en memoria:

- Los comandos del cliente
- Los comandos del servidor
- Los sms fallados en los envíos múltiples

2.2. Recomendaciones

Si el objetivo de tu aplicación es realizar un envío de un SMS al completar un formulario web. Te recomendamos la API simple.

Si por el contrario quieres realizar un envío múltiple y controlar los SMS que han fallado deberás utilizar la API completa.

Si su dominio esta alojado en un servidor compartido y no tiene privilegios para modificar las opciones de Apache o PHP le recomendamos que se

mire los ejemplos que encontrara en http://soporte.lleida.net/?p=211.

3. Descripción de las clases

Las clases incluidas dentro del directorio lib son clases privadas por lo que no vamos a explicar sus funcionalidades.

3.1. SimpleSMS

function SimpleSMS($user, $password)
Constructora. Parámetros usuario y contraseña de Lleida.net. No realiza la conexión.

function connect()
Conecta con el servidor por socket. Devuelve el estado de la conexión o el motivo del error.

function disconnect()
Se desconecta del servidor. Realiza una petición QUIT, si recibe un BYE desconecta, y establece a null el socket. Devuelve SMSMASS_CONN_CLOSED

function getConnectionStatus()
alias de function isConnected()
Devuelve TRUE si el socket esta conectado, FALSE en otro caso.

function isLogged()
Devuelve TRUE si el socket esta conectado y el usuario registrado, FALSE en otro caso.

function setDebugMode($debug)
Establece el valor para realizar el debug del protocolo.

function getDebugMode()
Devuelve el valor de debug del protocolo.

function setAllowAnswer($estat)
Para activar la respuesta al numero largo $estat debe de valer TRUE, para

desactivar el AllowAnswer $estat debe de valer FALSE. Si esta activo el Customized Sender no es valido. Si se produce un error también devuelve su código.

function acuseOn($mail)
Activa los ACUSE. Si $mail es cadena vacía o null devuelve la constante SMSMASS_NOOK_INVALID_EMAIL. Si se produce un error también devuelve su código como por ejemplo SMSMASS_NOOK_NOT_CONNECTED_TO_HOST.

function acuseOnCertifiedSMS($mail, $lang='ES', $type='D')
Activa los ACUSE certificados. Por defecto el idioma es el castellano y el tipo Certificado. $type puede valer D (certificado) y T (contrato). Si $mail es una cadena vacía o null devuelve SMSMASS_NOOK_INVALID_EMAIL. Si se produce otro error devuelve su código.

function acuseOff()
Desactiva los ACUSE. Devuelve SMSMASS_NOOK_NOT_CONNECTED_TO_HOST si se produce un error.

function getCredit()
Devuelve el crédito actual

function getPrice($num)
Devuelve la tarifa actual para ese numero o el código de error.

function getOperatorInfo($num)
Devuelve "PREFIJO CODIGOPAIS CODIGOOPERADORA.º -1 si se produce un error. Por ejemplo: "34652 34652 04"

function isInternationalNumberFormat($str)
Determina si la variable $str es un numero de telefono en formato internacional.

function getValidNumber($num)
Retorna la variable $num en formato internacional.

function getFileContentBase64($file)
Función para leer el contenido de un fichero del sistema de archivos y codificarlo en Base64 para su envío como mensaje binario WAP o MMS . Parámetro $file es la ruta absoluta del fichero. Devuelve una cadena en Base64 del contenido del fichero o null si se produce un error.

Del siguiente grupo de métodos los parámetros $text, $data, $URL, $recipients pueden ser arrays. El formato valido para $dateTime es YYYYMMDDHHmm . Por ejemplo, 197609081600 == 1976.August.8 at 16:00 == 8 de Agosto de 1.976 a las 16:00 .

Todos ellos tienen un primer parámetro que es el ID del envío, para que lo podáis gestionar vosotros, este parámetro debe de ser una variable. Todos ellos devuelven el correspondiente código de error si se produce un error.

function sendTextSMS($idEnvio, $text, $recipients, $dateTime)

function sendBinarySMS($idEnvio, $data, $recipients, $dateTime)
Debes de codificar el parámetro $data en Base64.

function sendUnicodeTextSMS($idEnvio, $text, $recipients, $dateTime)
Debes de codificar el parámetro $text en Base64. Mirar base64_encode() de php.

function sendWapPush($idEnvio, $subject, $text, $recipients, $mimeType, $fileData)
Debes de codificar el contenido del fichero en Base64, para ello puedes utilizar getFileContentBase64($filename).

function sendMMS($idEnvio, $subject, $text, $recipients, $mimeType, $fileData)
Debes de codificar el contenido del fichero en Base64, para ello puedes utilizar getFileContentBase64($filename).

function sendWapLink($idEnvio, $subject, $URL, $recipients)

function sendMMSMSG($idEnvio, $recipients, $mms)
Esta función permite enviar MMS en formato SMIL.

3.2. VirtualSMS

function VirtualSMS($user, $password) .
Constructora. Parámetros usuario y contraseña del masivo. Crea un objeto Properties con valores por defecto. Crea otro objeto Events estándar. No realiza el connect().

function setEvents($e)
Establece los el objeto Events. Acepta objetos extendidos de Events.

function getEvents()
Devuelve el objeto Events.

function setProtocolProperties($prop)
Establece el objeto ProtocolProperties. Esta clase la explico mas extensamente al final del documento.

function getProtocolProperties()
Devuelve el objeto ProtocolProperties.

function setUserProperties($prop)
Establece el objeto UserProperties. La idea es poder tener varios objetos UserProperties, con diferentes configuraciones y poder utilizar-los indistintamente.

function getUserProperties()
Devuelve el objeto UserProperties.

function checkall($num)
Realiza una petición de validación del numero $num. Por defecto, si el DEBUG esta activo, se almacena el resultado en el fichero vsms.log. Se recomienda extender la clase Events para capturar este tipo de respuestas.

function checknetwork($num)
Realiza una petición de validación del numero $num. Por defecto, si el DEBUG esta activo, se almacena el resultado en el fichero vsms.log. Se recomienda extender la clase Events para capturar este tipo de respuestas.

Los siguientes métodos son idénticos a la los de la clase SimpleSMS. Con la salvedad que no devuelven ningún valor, sino que delegan el control a la clase Events.

function connect()
function disconnect()
function getConnectionStatus()

function setAllowAnswer($estat)

function acuseOn($mail)
function acuseOnCertifiedSMS($mail, $lang='ES', $type='D')
function acuseOff()

function getCredits()
function getPrice($num)
function getOperatorInfo($num)

function getFileContentBase64($file)

function sendTextSMS($idEnvio, $text, $recipients, $dateTime='')
function sendBinarySMS($idEnvio, $data, $recipients, $dateTime='')
function sendUnicodeTextSMS($idEnvio, $text, $recipients, $dateTime='')
function sendWapPush($idEnvio, $subject, $text, $recipients, $mimeType, $fileData)
function sendMMS($idEnvio, $subject, $text, $recipients, $mimeType, $fileData)
function sendWapLink($idEnvio, $subject, $URL, $recipients)

3.3. Events

Los eventos se lanzaran al recibir las respuestas del servidor. Por ejemplo, al hacer un LOGIN si todo va bien saltara el evento connected(). Si hay algún error saltara el evento connectionError($errCode, $errMsg) .

function Events()
Constructora.

function connected()
Se produce cuando se ha establecido la conexión con el servidor y el usuario introducido se ha identificado correctamente en el sistema.

function disconnected()
Se produce cuando se pierde la conexión con el servidor, ya sea porque se ha llamado al método Disconnect o bien porque se ha perdido la conexión por un problema en la comunicación con el servidor. Use el evento Disconnect para detectar el momento en que se pierde la conexión con el servidor y pueda volver conectar si procede.

function deliveryReceipt($timeStamp, $sendTimeStamp, $recipient, $text,

$status)
Se produce cuando se notifica por parte del servidor o la operadora un acuse de recibo. Para que se reciba este evento es necesario establecer la propiedad MailDeliveryReceipt al valor ”INTERNAL”.

function deliver($idDeliver, $timeStamp, $sender, $recipient, $text)
Se produce cuando se recibe un mensaje de texto enviado a un número corto por parte de un usuario, a través del servidor SMS. Los SMS deliver deben responderse siempre. Si no se envía la respuesta pasadas 48 horas el SMS es descartado por el sistema. En el caso de no responderse al evento Deliver durante una conexión, el servidor envía de nuevo el evento Deliver al inicio de la siguiente conexión que se establezca.

function replyDeliver($idDeliver, $status)
Se produce cuando se recibe la confirmación del estado de un mensaje tras ser enviado a través del método DeliverSendBinarySMS o DeliverSendTextSMS.

function incomingMo($idIncomingMo, $timeStamp, $sender, $recipient, $text)
Se produce cuando se recibe un mensaje de texto enviado a un número largo por parte de un usuario, a través del servidor SMS.

function reply($idEnvio, $status, $data)
Se produce cuando se recibe la confirmación del estado de un mensaje tras ser enviado a través del método SendBinarySMS, SendMMS, SendUnicodeTextSMS, SendTextSMS, SendWapLink o SendWapPush.

function connectionError($errCode, $errMsg)
Se produce cuando se detecta cualquier tipo de error en el control. En la mayoría de los casos en el momento que se produce un error se cierra la conexión con el servidor. Las excepciones son los casos:

SMSMASS_ERR_ANOTHER_CONNECTION_IN_PROGRESS SMSMASS_ERR_PROTOCC
function noCredit()
Se produce cuando se detecta que el saldo es inferior a 1 crédito. Aunque este evento nos indica que el saldo es inferior a un crédito, en cualquier momento es posible consultar la propiedad Credit para saber el saldo actual.

function operatorInfo($phoneOrPrefix, $MCC, $MNC)
Se produce cuando se recibe la respuesta del método getOperatorInfo($num). Nos indica el código MCC (Mobile Country Code) y MNC (Mobile Network

Code) del número de teléfono o prefijo solicitado.

function checkall($id, $timeStamp, $num, $ok, $rcode, $mcc, $mnc)
Se produce cuando se recibe la respuesta del método checkall($num)

function checknetwork($id, $timeStamp, $num, $ok, $rcode, $mcc, $mnc)
Se produce cuando se recibe la respuesta del método checknetwork($num)

3.4. Properties

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

UserProperties

function UserProperties($user, $passwd)
Constructora. Los valores por defecto son:
$m_User = ”
$m_Pass = ”
$m_Credit = 0
$m_MailDeliveryReceipt = ”
$m_CustomizedSender = ”
$m_AllowAnswer = true
$m_Cert = false
$m_Lang = ’ES’
$m_Type = ’D’

Para establecer y obtener los valores de la clase:
function setUser($newUser)
function setPassword($newPass)
function setLang($lang)
function setCredit($credit)
function setMailDeliveryReceipt($newMailDeliveryReceipt)
function setCustomizedSender($newCustomizedSender)
function setAcuseType($type)
function setAllowAnswer($newAllowAnswer)
function setCertifiedSMS($newCert)
function setDefaultProperties($user, $passwd)

function getAcuseType()

function getUser()
function getPassword()
function getCredit()
function getMailDeliveryReceipt()
function getCustomizedSender()
function getLang()

function isAllowAnswer()
function isCertifiedSMS()

ProtocolProperties

function ProtocolProperties()
Constructora. Los valores por defecto son:
$debug = true
$m_Host = 'sms.lleida.net'
$m_Port = 2048
$m_ackAcuses = false
$m_ackChecker = false
$m_ackIncoming = false
$m_ackDeliver = false

Para establecer y obtener los valores de el servidor, puerto y debug de la aplicación.
function setHost($newHost)
function setPort($newPort)
function setDebugMode($deb)

function getDebugMode()
function getHost()
function getPort()

Para configurar los flags del protocolo.
function activeAllAck()
function deactiveAllAck()

function setAcusesAck($bln)
function setCheckerAck($bln)
function setIncomingAck($bln)
function setDeliverAck($bln)

function isActivatedAcusesAck()
function isActivatedCheckerAck()
function isActivatedIncomingAck()
function isActivatedDeliverAck()