Passerelle SMS - Interface XML-RPC

Table des matières

Notez que la dernière version de ce document est toujours disponible sur https://sms.netoxygen.ch/api2/documentation/ ; vous pouvez comparer le numéro de version dans le pied de page pour vous assurer d'utiliser la version la plus récente disponible.

Description des fonctions disponibles

L'URL du serveur est : https://sms.netoxygen.ch/api2/ (ou http://sms.netoxygen.ch/api2/).

Envoyer un SMS

fonction : send_message
paramètres : user, pass, dest, message[,sender,date,type]
retourne : un tableau de tableaux

→ Le troisième paramètre peut être soit une chaîne de caractères, soit un tableau contenant plusieurs chaînes, une par destinataire.

→ Un 5ème paramètre optionnel permet de définir l'expéditeur. Si le champ est vide ou si l'utilisateur n'a pas les droits nécessaires pour “forcer” un expéditeur particulier, l'expéditeur par défaut est utilisé. S'il n'y a pas d'expéditeur par défaut, “textobox.ch” est utilisé.

→ Un 6ème paramètre optionnel permet de définir la date à laquelle le message doit être envoyé (fuseau horaire de référence : Europe/Zurich). Si le paramètre est vide, contient une date invalide ou une date dans le passé, le message est envoyé immédiatement. Le format à utiliser est le format ISO, eg: “2006-12-06 20:05:00”. Pour spécifier une date mais utiliser l'expéditeur par défaut, il suffit de passer une chaîne vide comme 5ème paramètre.

→ Un 7ème paramètre optionnel permet de définir le type d'envoi. Il peut prendre 2 valeurs : “text“ ou “flash“, par défaut, si rien n'est défini, le type est “text“. Un SMS de type “flash“ apparaît directement sur l'écran du terminal du destinataire sans qu'il n'ait besoin d'ouvrir le SMS.

→ Le tableau retourné comprend pour chaque destinataire un identifiant unique ainsi que le statut d'envoi ; l'identifiant unique peut être utilisé pour interroger ultérieurement le statut d'acheminement.

Vérifier un couple utilisateur/mot de passe

fonction : auth
paramètres : user, pass
retourne : TRUE/FALSE

Consulter le solde de crédits

fonction : get_credits
paramètres : user, pass
retourne : le nombre de crédits restants ou FALSE

Consulter le statut du compte

Fonction : account_status
paramètres : user, pass
retourne : 'regular' ou 'free'

Obtenir le « group id » d'un compte

fonction : get_guid
paramètres : user, pass
retourne : l'identifiant unique du groupe (compte) auquel appartient l'utilisateur ou FALSE

→ Cet identifiant unique est une chaîne de 36 caractères au format UUID aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee.

→ Cette fonctionnalité est avant tout destinée à permettre aux applications utilisant l'API d'enregistrer des préférences globales pour un compte, et pas uniquement pour un utilisateur.

→ Le résultat de cette fonction ne devrait pas être mis en cache pour une trop longue durée, puisqu'on ne peut pas exclure qu'un utilisateur soit ultérieurement « déplacé » vers un autre compte.

Normaliser un numéro

fonction : clean_number
paramètres : user, pass, numéro
retourne : un numéro

→ ex : clean_number('022/364 8000') retourne '41223648000'

Consulter le statut d'acheminement d'un message

fonction : get_status
paramètres : user, pass, identifiant
retourne : un tableau de tableaux

→ Le troisième paramètre peut être soit une chaîne de caractères, soit un tableau contenant plusieurs chaînes, une par message dont on souhaite interroger le statut.

→ Le tableau retourné comprend les informations sur l'acheminement pour chaque identifiant passé en paramètre (code d'erreur et code de statut).

Annuler l'acheminement d'un message planifié

fonction : cancel_message
paramètres : user, pass, identifiant
retourne : un tableau de tableaux

→ Le troisième paramètre peut être soit une chaîne de caractères, soit un tableau contenant plusieurs chaînes, une par message dont on souhaite interroger le statut.

→ Le tableau retourné comprend une valeur booléenne indiquant si l'annulation du message planifié a pu être effectué avec succès pour chaque identifiant passé en paramètre.

Liste des statuts et des codes d'erreurs

Codes d'erreur

Les statuts du message (c.f. prochaine section) donne plus de détails sur le code d'erreur actuel.

code	description	commentaire
1	OK	Le message a été acheminé ou est en cours d'acheminement.
2	Gateway error	Problème avec la passerelle.
3	Message routing error	Problème de routage du SMS.
4	Message format error	Problème avec le format du message.
5	Invalid destination number	Numéro de destination invalide, le message ne sera pas reçu.
6	Invalid source number	Numéro source invalide, le message ne sera pas envoyé.
7	Empty message	Le message était vide.
8	Max message length exceeded	La taille maximale de message a été dépassée.
9	Pending	Le message est en attente.
10	Message canceled	Le message planifié a été annulé avant l'envoi.

Codes de statut

code	description	commentaire
1	Delivered to the recipient device	Le message a été reçu par le destinataire.
2	Delivered to the recipient network	Le message est en attente sur le réseau du destinataire.
3	Message submitted	Le message a été soumis au réseau du destinataire.
4	Message error, delivery failed	Une erreur liée au message est survenue.
5	Routing error, delivery failed	Une erreur liée au routage du SMS est survenue.
6	Message expired, delivery cancelled	Le destinataire n'a pas reçu le message avant son expiration.
7	Gateway error	Erreur de la passerelle.
8	Message in queue	Le message est en cours de traitement par notre passerelle.
9	Message scheduled	Le message est planifié pour un envoi dans le futur.

Support technique

En cas de difficultés ou de questions, nous vous invitons à :

- consulter les questions fréquemment posées (FAQs) sur notre site Web : https://netoxygen.ch/support/faqs/
- nous contacter à l'adresse support@netoxygen.ch
- nous contacter au +41 (0)22 364 8000 (ou au numéro 24/7 en dehors des heures de bureau pour les titulaires d'un contrat de support)

Exemples d'utilisation

Envoi de SMS depuis PHP

```php
<?php
require_once("XML/RPC.php");  // fait partie de PEAR

$cli = new XML_RPC_Client("/api2/", "sms.netoxygen.ch");
//$cli->debug = true;   // décommenter en cas de besoin

$msg = new XML_RPC_Message("send_message",                // la fonction appelée
    array(new XML_RPC_Value("user33", "string"),          // nom d'utilisateur (compte)
          new XML_RPC_Value("mypass", "string"),          // mot de passe
          new XML_RPC_Value("0792223344", "string"),      // destinataire
          new XML_RPC_Value("message de test", "string")  // message
    )
  );

$answer = $cli->send($msg);
?>
```

Vous trouverez une implémentation complète ainsi que plus d'exemple à l'adresse :
https://github.com/NetOxygen/smsapiv2clients/tree/master/PHP

Programme simple d'envoi de SMS en Perl

```
#!/usr/bin/perl
use RPC::XML::Client;
my $cli = RPC::XML::Client->new('http://sms.netoxygen.ch/api2/');
my $resp = $cli->send_request('send_message', ('myuser', 'mypass', @ARGV[0], @ARGV[1]));
```

Programme Perl plus complet (gère correctement les accents et les erreurs)

```
#!/usr/bin/perl

use strict;

unless (scalar(@ARGV) == 2) {
  die("Usage: sendsms.pl destination message\n");
}

use RPC::XML qw (smart_encode);
use RPC::XML::Client;
use Encode;

my $user = smart_encode('myuser');
my $pass = smart_encode('mypass');

my $dst = new RPC::XML::base64(Encode::encode('utf8', @ARGV[0]));
my $msg = new RPC::XML::base64(Encode::encode('utf8', @ARGV[1]));

my $cli = RPC::XML::Client->new('http://sms.netoxygen.ch/api2/');
my $resp = $cli->send_request('send_message', ($user, $pass, $dst, $msg));

unless (ref $resp) {
  die("Fatal error: XML::RPC communication failed. ($resp)\n");
}

if ($resp->is_fault) {
  die("Fatal error: XML::RPC response returned with fault status. (code: ".$resp->code.",
".$resp->string.")\n");
}

my ($d, $s, $i) = @{@{$resp->value}[0]};
unless ($s) {
  die("Error: unable to send message to $d. (Error code: $i)\n");
}

print "$i\n";
```

.Net

Il existe plusieurs implémentations d'XML-RPC pour .Net, on peut par exemple utiliser celle-ci : http://xml-rpc.net/ (C# ou VB.NET).