**Integrazione SMS PHP SDK**

Indice

[1 INTRODUZIONE 4](#__RefHeading__32862_1366233465)

[2 CONFIGURAZIONE DELLA LIBRERIA 4](#__RefHeading__32864_1366233465)

[3 INVIO DI SMS 5](#__RefHeading__32866_1366233465)

[4 STATO DEI MESSAGGI 6](#__RefHeading__32868_1366233465)

[5 STORICO DEI MESSAGGI 8](#__RefHeading__32870_1366233465)

[6 CREDITO SMS DISPONIBILE 9](#__RefHeading__32872_1366233465)

[7 GESTIONE SUBACCOUNTS 10](#__RefHeading__32874_1366233465)

[8 GESTIONE CREDITI SUBACCOUNTS 11](#__RefHeading__32876_1366233465)

1. INTRODUZIONE

[**[ INDICE ]**](#INDICE)

I server Aruba mettono a disposizione degli sviluppatori alcune interfacce di comunicazione HTTP per l'invio e la ricezione dei messaggi, la verifica di validità dei numeri di telefono, e permettono inoltre di recuperare tutti i dati riguardanti lo stato dei messaggi e lo storico.

1. CONFIGURAZIONE DELLA LIBRERIA

[**[ INDICE ]**](#INDICE)

Modificare opportunamente username e password nel file config.php:

<?php

define('SDK\_HOSTNAME','https://adminsms.aruba.it');

define('SDK\_USERNAME','your\_Aruba\_login');

define('SDK\_PASSWORD','your\_Aruba\_password');

define('SDK\_DEFAULT\_PORT',80);

define('SDK\_PROXY','');

define('SDK\_PROXY\_PORT',8080);

?>

Se si utilizza un proxy, vanno configurate nello stesso file le voci PROXY e PROXY\_PORT.

1. INVIO DI SMS

[**[ INDICE ]**](#INDICE)

La classe Sdk\_SMS (file sendsms.php) contiene tutti i dati necessari per inviare un messaggio SMS attraverso il gateway Aruba.

Creato un oggetto di tipo SMS, è necessario impostare:

* tipo di messaggio: ALTA (attributo sms\_type);
* destinatari: numero di telefono dei destinatari del messaggio in formato internazionale; per aggiungere un numero alla lista dei destinatari, utilizzare il metodo add\_recipient($string)
* messaggio: attributo message, imposta il testo del messaggio da inviare, massimo 160 caratteri per un'unico SMS, massimo 1000 caratteri per un SMS multiplo (concatenato). Attenzione: i caratteri ^ { } \ [ ~ ] | € vengono calcolati come 2 caratteri; è possibile utilizzare il metodo length() per conoscere il numero di caratteri totali dell'sms, ed il metodo count\_smss() per richiedere il calcolo del numero di SMS necessari per effettuare l'invio;
* mittente: nel caso di SMS con mittente personalizzato, massimo 16 caratteri numerici per un numero di telefono in formato internazionale oppure 11 caratteri alfanumerici per una stringa di testo (Es: "+123456789012345" oppure "Azienda Spa"), attributo sender.

Opzionalmente, si possono definire:

* data e ora per l'invio di SMS posticipato ( metodo set\_scheduled\_delivery($unix\_timestamp) );
* una stringa alfanumerica da associare all'invio, indispensabile per ottenere successivamente le informazioni riguardanti lo stato dei messaggi (attributo order\_id). La lunghezza massima è di 32 caratteri, tutti i caratteri eccedenti non saranno considerati.

Se l'order\_id non viene specificato, il server ne genererà uno garantito univoco.

L'order\_id potrà essere utilizzato per richiedere lo stato del messaggio per i 30 giorni successivi all'invio.

Esempio:

require('../sendsms.php');

$sms = new Sdk\_SMS();

$sms->sms\_type = SMSTYPE\_ALTA;

$sms->add\_recipient('+393479057982');

$sms->add\_recipient('+393479876543');

$sms->message = 'hello world!';

$sms->sender = 'test';

$sms->set\_immediate(); // or sms->set\_scheduled\_delivery($unix\_timestamp)

$sms->order\_id = '999FFF111'; // Optional

echo 'About to send a message '.$sms->count\_smss().' SMSs long ';

echo 'to '.$sms->count\_recipients().' recipients </br>';

if ($sms->validate()) {

$res = $sms->send();

if ($res['ok']) {

echo $res['sentsmss'].' SMS sent, order id is '.$res['order\_id'].' </br>';

} else {

echo 'Error sending SMS: '.$sms->problem().' </br>';

}

} else {

echo 'invalid SMS: '.$sms->problem().' </br>';

}

1. STATO DEI MESSAGGI

[**[ INDICE ]**](#INDICE)

Successivamente all'invio dei messaggi è possibile richiederne lo stato, utilizzando la funzione sdk\_get\_message\_status($order\_id). Specificando l'id dell'SMS, si richiede al server l'esito dell'invio e viene restituito un array contenente l'esito della richiesta e, nel caso di esito positivo, una lista di oggetti sdk\_SMS\_STATUS, ognuno dei quali fornisce:

* il numero di telefono del destinatario (attributo recipient);
* lo stato di invio (attributo status);
* data e ora di effettiva ricezione del messaggio, se l'sms è stato correttamente ricevuto dal destinatario. L'attributo dt\_received contiene la data/ora in formato stringa, il metodo get\_sms\_received\_timestamp(), invece, restituisce la data/ora come unix timestamp; è anche possibile utilizzare la funzione strptime() per ottenere un array associativo come da <http://it.php.net/manual/en/function.strptime.php>, la stringa di formato utilizzata e' SDK\_DATE\_TIME\_FORMAT (vedi esempio).

Il numero di oggetti sdk\_SMS\_STATUS restituiti è contenuto nell'array stesso ed ha la stringa count come associazione (si veda esempio a fondo pagina).

L'attributo status può assumere i seguenti valori:

SCHEDULED // posticipato, non ancora inviato

SENT // inviato, non attende delivery

DLVRD // l'SMS è stato correttamente ricevuto

ERROR // errore in invio dell'SMS

TIMEOUT // l'operatore non ha fornito informazioni sull'SMS entro le 48 ore

TOOM4NUM // troppi SMS per lo stesso destinatario nelle ultime 24 ore

TOOM4USER // troppi SMS inviati dall'utente nelle ultime 24 ore

UNKNPFX // prefisso SMS non valido o sconosciuto

UNKNRCPT // numero di telefono del destinatario non valido o sconosciuto

WAIT4DLVR // messaggio inviato, in attesa di delivery

WAITING // in attesa, non ancora inviato

UNKNOWN // stato sconosciuto

Se non si vuole implementare una gestione particolareggiata degli eventuali errori d'invio, il metodo is\_error() restituisce true se il messaggio è in errore, false altrimenti.

Il metodo str\_status() fornisce una descrizione testuale dello stato.

Esempio:

require('../sms\_status.php');

$status = sdk\_get\_message\_status('1F11DC4A77A8F');

if ($status['ok']) {

for ($i=0;$i<$status['count'];$i++) {

if ($status[$i]->is\_error()) {

echo 'Error sending sms to '.$status[$i]->recipient.': ';

echo $status[$i]->str\_status();

} else {

echo 'Message to number '.$status[$i]->recipient.': ';

if ($status[$i]->status == 'DLVRD') {

echo 'correctly delivered on ';

echo date('l ',$status[$i]->get\_sms\_received\_timestamp());

echo ' at ';

echo date('H:i',$status[$i]->get\_sms\_received\_timestamp());

} else {

echo $status[$i]->str\_status();

}

}

echo '<br/>';

}

} else {

echo 'Request failed: '.$status['errcode'].' - '.$status['errmsg'];

}

1. STORICO DEI MESSAGGI

[**[ INDICE ]**](#INDICE)

La funzione sdk\_get\_sms\_history($from,$to) richiede al server Aruba la lista degli SMS inviati nell'intervallo fra due date specifiche; restituisce un array di oggetti sdk\_SENT\_SMS. I parametri $from e $to sono unix timestamp.

Esempio:

require('../sms\_history.php');

$history = sdk\_get\_sms\_history(time()-(7\*24\*60\*60),time());

if ($history['ok']) {

for ($i=0;$i<$history['count'];$i++) {

echo 'SMS id '.$history[$i]->order\_id.' sent from '.$history[$i]->sender;

echo ' to '.$history[$i]->recipients\_count.' recipients<br/>';

}

} else {

echo 'Request failed: '.$history['errcode'].' - '.$history['errmsg'];

}

1. CREDITO SMS DISPONIBILE

[**[ INDICE ]**](#INDICE)

La funzione sdk\_get\_credits() richiede al server Aruba la propria disponibilità di credito; restituisce un array di oggetti sdk\_CREDIT. Ogni oggetto sdk\_CREDIT contiene: il tipo, la nazione (null se credito internazionale) e la quantità di credito disponibile.

Esempio:

require('../credits.php');

$credits = sdk\_get\_credits();

if ($credits['ok']) {

for ($i=0;$i<$credits['count'];$i++) {

if (!$credits[$i]->is\_international()) {

echo 'You can send '.$credits[$i]->availability.' smss';

echo ' of type '.$credits[$i]->credit\_type;

echo ' in '.$credits[$i]->nation.' </br>';

} else {

if ($credits[$i]->credit\_type == 'EE') {

echo 'You can send '.$credits[$i]->availability;

echo ' smss in foreign countries </br>';

}

}

}

} else {

echo 'Request failed: '.$status['errcode'].' - '.$status['errmsg'];

}

1. GESTIONE SUBACCOUNTS

[**[ INDICE ]**](#INDICE)

Un utente superaccount può utilizzare alcuni metodi dell'oggetto SMSConnection per verificare e gestire i propri subaccounts. Nello specifico

- sdk\_getSubaccounts(), restituisce una List<Subaccount> contenente tutti i subaccount dell'utente corrente

- sdk\_createSubaccount(Subaccount) crea e restituisce un nuovo subaccount; login e password sono generati da Aruba, e devono essere letti dall'oggetto Subaccount restituito  
- sdk\_lockSubaccount(Subaccount) e sdk\_unlockSubaccount(Subaccount), permettono di attivare/disattivare un subaccount; in entrambi i casi viene restituito un Subaccount

Esempio:

**require**('../subaccounts.php');

**echo** 'Listing subaccounts:</br>';

$subs = sdk\_get\_subaccounts();

**if** ($subs['ok']) {

**for** ($i=0;$i<$subs['count'];$i++) {

**echo** 'Subaccount login: '.$subs[$i]->login.' active: '.$subs[$i]->active;

**echo** ' type: '.$subs[$i]->credit\_mode.' name: '.$subs[$i]->name.' '.$subs[$i]->surname.' <br/>';

}

} **else** {

**echo** 'Request failed: '.$subs['errcode'].' - '.$subs['errmsg'];

}

**echo** 'Creating new subaccount:</br>';

$subaccount = **new** sdk\_SUBACCOUNT();

$subaccount->credit\_mode = 2;

$subaccount->company\_name = 'azienda prova';

$subaccount->fiscal\_code = 'XXXXXX80S11L111X';

$subaccount->vat\_number = '012345678901234';

$subaccount->name = 'testname';

$subaccount->surname = 'testsurname';

$subaccount->email = 'testmail@your.it';

$subaccount->address = 'testaddress';

$subaccount->city = 'testcity';

$subaccount->province = 'BO';

$subaccount->zip = '38300';

$subaccount->mobile = '3490000000';

$subaccount->active = **true**;

$subaccount->createSubaccount();

**echo** 'Subaccount login: '.$subaccount->login.' password: '.$subaccount->password.' active: '.$subaccount->active.' <br/>';

**echo** 'Disabling subaccount:</br>';

$subaccount->lockSubaccount();

**echo** 'Subaccount login: '.$subaccount->login.' password: '.$subaccount->password.' active: '.$subaccount->active.' <br/>';

**echo** 'Enabling subaccount:</br>';

$subaccount->unlockSubaccount();

**echo** 'Subaccount login: '.$subaccount->login.' password: '.$subaccount->password.' active: '.$subaccount->active.' <br/>';

1. GESTIONE CREDITI SUBACCOUNTS

[**[ INDICE ]**](#INDICE)

Un utente superaccount può dare/togliere crediti ai propri subaccounts.

In base alla tipologia di credito del subaccount, sono disponibili metodi differenti.

- Ai subaccount con tipo credito *USE\_SUPER\_CREDIT* non è possibile dare/togliere crediti.

- Ai subaccount con tipo credito *HAS\_CREDIT* è possibile dare/togliere crediti tramite il metodo sdk\_moveCredits(CreditMovement). Per visualizzare i crediti del subaccount si utilizza il metodo sdk\_getSubaccountCredits(Subaccount), che restituisce una lista di Credit.

- Ai subaccount con tipo credito *USE\_BOTH\_CREDITS* è possibile dare crediti con il metodo sdk\_createPurchase(CreditMovement) e togliere credito con il metodo sdk\_deletePurchase(CreditMovement). Per visualizzare gli acquisti del subaccount, si utilizza il metodo sdk\_getPurchases(Subaccount), che restituisce una List<CreditMovement>.

Esempio:

**require('../credit\_movements.php');**

**require**('../credits.php');

**echo** 'Giving credits to subaccount xxxS\_004<br>';

$cm1= **new** sdk\_CREDIT\_MOVEMENT();

$cm1->subaccount\_login = 'xxxS\_004';

$cm1->super\_to\_sub = 'y';

$cm1->amount=35;

$cm1->message\_type='GP';

$cm1->moveCredits();

**echo** 'Listing credits for subaccount xxxS\_004<br>';

$credits = sdk\_get\_subaccount\_credits('xxxS\_004');

**if** ($credits['ok']) {

**for** ($i=0;$i<$credits['count'];$i++) {

**if** (!$credits[$i]->is\_international()) {

**echo** 'You can send '.$credits[$i]->availability.' smss';

**echo** ' of type '.$credits[$i]->credit\_type;

**echo** ' in '.$credits[$i]->nation.' </br>';

} **else** {

**if** ($credits[$i]->credit\_type == 'EE') {

**echo** 'You can send '.$credits[$i]->availability;

**echo** ' smss in foreign countries </br>';

}

}

}

} **else** {

**echo** 'Request failed: '.$credits['errcode'].' - '.$credits['errmsg'];

}

**echo** 'Creating purchase for subaccount xxxS\_001<br>';

$cm2= **new** sdk\_CREDIT\_MOVEMENT();

$cm2->subaccount\_login = 'xxxS\_001';

$cm2->price = '56.7';

$cm2->sms\_types = explode(';', 'GP;GS');

$cm2->price\_per\_message = explode(';', '0.02;0.009');

$cm2->createPurchase();

**echo** 'Listing purchases for subaccount xxxS\_001<br>';

$purchases = sdk\_get\_subaccount\_purchases('xxxS\_001');

**if** ($purchases['ok']) {

**for** ($i=0;$i<$purchases['count'];$i++) {

**echo** ($purchases[$i]->super\_to\_sub ? '--> ' : '<-- ');

**echo** $purchases[$i]->amount;

**echo** '(price '.$purchases[$i]->price.', available '.$purchases[$i]->available\_amount.') ppm: ';

**for** ($j=0;$j<count($purchases[$i]->sms\_types);$j++) {

**echo** $purchases[$i]->sms\_types[$j].' '.$purchases[$i]->price\_per\_message[$j].', ';

}

**echo** '<br>';

}

} **else** {

**echo** 'Request failed: '.$purchases['errcode'].' - '.$purchases['errmsg'];

}

**echo** 'Deleting last purchase<br>';

$purchases[0]->deletePurchase();