DOKUMENTACJA

SOAP API

ver 2.0

Zawartość

Wprowadzenie 3
Wysyłanie wiadomości SMS 4
Przykład #1: wysyłka EkoSMS 8
Przykład #2: wysyłka MaxSMS 10
Raporty dostarczenia 12
Metoda getReportsById() 12
Metoda getReportsByMessageId() 15
Metoda getReportsByTime() 18
Odbieranie odpowiedzi 21
Sprawdzanie stanu konta 23
Konwertowanie wiadomości 25
Walidowanie numeru 27
Dodawanie kontaktów 32
Statusy końcowe operacji 34

Wprowadzenie

SOAP to protokół wywoływania zdalnego dostępu do obiektów, wykorzystujący język XML do kodowania wywołań.

Zawsze aktualne przykłady wykorzystania naszego API oraz gotowe biblioteki znajdziesz na naszej stronie na GitHub.

Szyfrowanie danych

Dla zapewnienia najwyższego poziomu bezpieczeństwa wywołania protokołu transportowego HTTP są przez nas szyfrowane poprzez SSL. Aby korzystać z szyfrowanego połączenia upewnij się, że odnosisz się do bazowego adresu usługi poprzez protokół HTTPS.

Filtrowanie IP

Aby zminimalizować możliwość nieuprawnionego korzystania z konta SMS przez osoby niepowołone, istnieje możliwość ustawienia listy adresów IP uprawnionych do korzystania z API. Przy włączonej filtracji adresów IP, wysyłka poprzez API będzie możliwa tylko przez klienty łączące się z adresu IP znajdującego się na liście. Włączenie filtrowania adresów IP, a także zarządzanie listą adresów, możliwe jest po zalogowaniu się do Panelu SMS i przejściu do sekcji Ustawienia → Ustawienia API.

Poszerzone bezpieczeństwo informacji

Dodatkowo, w ramach rozszerzonych funkcjonalności systemu, jesteśmy w stanie zaoferować Państwu:

- usuwanie informacji o odbiorcach wiadomości, ich treści po określonym przez Państwa czasie; wówczas w historii wysyłek, a także wszędzie w naszych bazach danych, numery odbiorców są zastępowane z postaci typu +48783389136 (numer odbiorcy) na postać

typu +487833XXXXX; natomiast treść jest zastępowana z postaci typu: To jest treść wiadomości SMS na postać typu: Xx xxxx xxxxx xxxxxxxxxx XXX,

- ustanawianie połączeń VPN, co zostało przedstawione w dalszej części dokumentacji. Jeżeli będą Państwo zainteresowani powyższymi dodatkowymi funkcjonalnościami - prosimy o kontakt w celu ustalenia szczegółów.

Połączenie VPN dla zwiększenia bezpieczeństwa

W razie potrzeby możemy utworzyć dla Państwa dostęp do naszego API poprzez VPN. Usługa ta pozwoli Państwu jeszcze bardziej zwiększyć bezpieczeństwo podczas wymiany danych między naszymi systemami, czyniąc całą komunikację dla ewentualnych intruzów zupełnie nieczytelną. W takim wypadku zalecamy użycie OpenVPN, gdzie Państwa system jest klientem naszej platformy. Kiedy będą Państwo zainteresowani tego typu integracją – prosimy o kontakt.

Zasada działania

Większość popularnych technologii posiada ogólnodostępne biblioteki wspierające protokół SOAP (np. SoapClient dla PHP czy Axis2 dla Java). Podstawą tworzenia klienta SOAP jest plik WSDL. Poniżej znajduje się lista adresów plików WSDL dostępnych w systemie PromoSMS.

https://promosms.com/remote/soap/wsdl

Wysyłanie wiadomości SMS

Wysyłka wiadomości odbywa się poprzez wywołanie metody `send()` z odpowiednimi parametrami wejściowymi:

sLogin (type string)

Identyfikator (adres e-mail) użytkownika

sPassword (type string)

Hasło użytkownika w postaci jawnej lub skrótu `md5()`

bReturnSmsid (type boolean)

Ustawienie wartości dla tego parametru na `true` powoduje, że funkcja zwraca tablicę/wektor pojedynczych identyfikatorów dla każdego odbiorcy wiadomości. Liczba zwracanych danych jest zatem większa, jednak daje to możliwość oprogramowaniu klienckiemu odczytanie wszystkich unikalnych identyfikatorów, które zostały przypisane do każdego wysłanego SMSa.

aMessages (type messagesToSend)

Tablica/wektor zawierająca wiadomości

sMessage (type string)

Treść wysyłanej wiadomości

bBinarySms (type boolean)

Parametr opcjonalny. Ustawienie tej wartości jako `true` spowoduje, że zostanie wysłany SMS typu WAP PUSH. W takim wypadku obowiązkowe jest zdefiniowanie wartości parametru sUdh

sUdh (type string)

Jeżeli ma być wysłana wiadomość typu WAP PUSH, należy zdefiniować ten parametr, wprowadzając żądaną wartość nagłówka UDH

bSpecialChars (type boolean)

Ustawienie tej wartości jako `true` spowoduje, że znaki specjalne nie będą zamieniane na odpowiedające im znaki standardowe, lecz wiadomość zostanie wysłana w postaci przekazanej do systemu. Warto jednak pamiętać, że przekazywanie w treści SMSa znaków specjalnych powoduje, że długość SMSa zmniejsza się ze standardowych 160 znaków do 70. Jeżeli zatem chcemy, aby

platforma PromoSMS usuwała wszelkie niestandardowe znaki (znaki specjalne), należy podać wartość tej zmiennej ustawioną na `false`

bLongSms (type boolean)

Ustawienie tej wartości na `true` powoduje, że przez platformę są przyjmowane do wysyłki także wiadomości dłuższe niż 160 znaków. Takie wiadomości będą dostarczone jako jedna całość (potocznie: jako jeden SMS), ale należy liczyć się z wyższym kosztem wysyłki (uzależniony od ilości wiadomości składowych)

iType (type int)

Typ wysyłanego SMSa. Dozwolone typy to:

- 0 - FlashSMS
- 1 - EkoSMS (domyślny)
- 3 - MaxSMS
- 4 - FasterSMS
- 5 - SMS zagraniczny
- 6 - FasterSMS zagraniczny

sFrom (type string)

Nazwa nadawcy SMSa (wyświetlana zamiast numeru telefonu). Aby korzystać z własnej nazwy nadawcy należy ją wcześniej zarejestrować. W przypadku EkoSMS/FlashSMS parametr ten jest opcjonalny

sDeliveryUrl (type string)

Adres URL serwera użytkownika, na który mają być przekazywane raporty dostarczenia SMSa. W adresie URL mogą być podane następujące zmienne specjalne, które zostaną zastąpione odpowiednią wartością przez system podczas przesyłania raportu na serwer klienta:

- %smsID - ID wysłanej wiadomości
- %timestamp - data dostarczenia wiadomości w postaci znacznika czasu

- %number - numer na jaki została wysłana wiadomość
- %report - ID raportu (1 oznacza dostarczone)
- %ownID - jeżeli używasz własnych identyfikatorów, możesz użyć również tej opcji.

aRecipients (type string[])

Tablica/wektor odbiorców wiadomości. W tablicy stringów mają znajdować się więc odbiorcy wiadomości (każdy element tablicy to kolejny odbiorca)

iDate (type int)

PromoSMS może wysłać wiadomość o określonej porze i w określonym dniu. Aby to zrobić, należy przekazać w postaci uniksowego znacznika czasu datę i godzinę wysyłki wiadomości. Parametr ten jest opcjonalny. Domyślnie nie jest ustawiona żadna wartość (wiadomość jest wysyłana od razu)

aUserIndexes (type string[])

Parametr pozwala użytkownikowi zdefiniować własne identyfikatory wiadomości. Liczba elementów tablicy/wektora musi być równa liczbie zdefiniowanych odbiorców wiadomości. Wówczas każdej wiadomości, będzie przyporządkowany kolejny element z poniższej tablicy. Możliwe jest również zdefiniowanie jednego elementu dla tej tablicy/wektora, aby wszystkim wysłanym wiadomościom był przypisany ten sam identyfikator. Parametr jest opcjonalny.

W wyniku wywołania metody `send()` zwracany jest obiekt typu `sendResults`:

iResult (type int)

Status wysłania wiadomości. Statusy są w postaci stringa typu: 001, 032 itp.
Szczegóły i opis wszystkich statusów znajduje się w dalszej części dokumentacji.

sDescription (type string)

Słowny opis statusu

iPrice (type int)

Całkowity koszt wysyłki wyrażony w groszach

iSmsCount (type int)

Całkowita liczba wysłanych wiadomości SMS

vSendRecipientsResults (type sendRecipientsResults[])

Tablica/wektor obiektów typu sendRecipientsResults. Każdy obiekt tego typu reprezentuje stan przekazania do wysłania wiadomości SMS do jednego odbiorcy

sRecipient (type string)

Numer telefonu odbiorcy wiadomości

sSmsId (type string)

Unikalny identyfikator smsid dla danego odbiorcy i danej wiadomości

sOwnId (type string)

Identyfikator wiadomości nadany przez aplikację klienta

sStatus (type string)

Status przekazania do wysłania wiadomości dla danego odbiorcy

Przykład #1: wysyłka EkoSMS

```
Przykład ilustruje najprostszy sposób wysłania wiadomości EkoSMS.
```

Przykładowy kod

```
1.
2. <?php require_once(dirname(__FILE__).'/../../common.php');
3. $sLogin = "login";
4. $sPassword = "haslo";
5.
```

```
6.
7. $client = new SoapClient( 'https://promosms.com/remote/soap/wsdl',
8.  array( 'features' => SOAP_SINGLE_ELEMENT_ARRAYS,
9.  'cache_wsdl' => WSDL_CACHE_NONE ));
10.       $aMessages = array();
11.       $aMessages[0] = array(
12.        'sMessage' => 'To jest jakaś treść',
13.        'iType' => 1,
14.        'aRecipients' => array( "783389136" )
15.       );
16.
17.       try {
18.        $result = $client->send( array( 'sLogin' => $sLogin,
19.        'sPassword' => $sPassword,
20.        'aMessages' => $aMessages,
21.        'bReturnSmsid' => true ) );
22.
23.        echo "Łączna liczba odbiorców wysyłki: " . $result-
   >{'iSmsCount'} . "<br />";
24.        echo "Koszt wysyłki: " . $result->{'iPrice'} . " groszy<br
   />";
25.
26.        foreach($result->{'vSendRecipientsResults'} as $recipient) {
27.        echo "Numer odbiorcy: <b>" . $recipient->{'sRecipient'} .
   "</b><br />";
28.        echo "ID wiadomości: " . $recipient->{'sSmsId'} . "<br />";
29.        echo "ID wiadomości (nadane przez użytkownika): " .
   $recipient->{'sOwnId'} . "<br />";
30.        echo "Status przekazania wiadomości do wysłania: " .
   $recipient->{'sStatus'} . "<br />";
31.        echo "<br />";
32.        }
33.
34.
35.       } catch(Exception $e) {
36.        echo "<b>Błąd podczas wysyłki.</b><br />";
37.        echo "Numer błedu: " . $e->faultstring . "<br />";
38.        echo "Opis błedu: " . $e->detail . "<br />";
39.       }
40.
41.
42.       ?>
43.
```

Przykład #2: wysyłka MaxSMS

Aby wysłać wiadomość typu MaxSMS, należy podać wartość `3` dla parametru `iType` oraz nazwę nadawcy w parametrze `sFrom`. Poniższy przykład ilustruje również sposób używania dodatkowych opcji takich jak: określenie czasu wysyłki, określenie adresu zwrotnego dla raportów, zachowanie znaków specjalnych, dopuszczenie wieloczęściowych wiadomości.

Aby wysyłać wiadomości MaxSMS z własną nazwą nadawcy (np. nazwą swojej firmy lub marki), należy zarejestrować nazwę - w tym celu prosimy o kontakt z Biurem Obsługi Klienta pod adresem e-mailbiuro@PromoSMS.pl.

Przykładowy kod

```
1.
2. $sLogin = "login";
3. $sPassword = "haslo";
4.
5.
6. $client = new SoapClient( 'https://promosms.com/remote/soap/wsdl',
7.  array( 'features' => SOAP_SINGLE_ELEMENT_ARRAYS,
8.  'cache_wsdl' => WSDL_CACHE_NONE ));
9. $aMessages = array();
10.       $aMessages[0] = array(
11.        'sMessage' => 'To jest jakaś treść',
12.        'bSpecialChars' => true,
13.        'bLongSms' => false,
14.        'iType' => 3,
15.        'sFrom' => 'SMSInfo',
16.        'sDeliveryUrl' =>
17.
   'https://PromoSMS.pl/raporty?smsid=%smsID&data_raportu=%timestamp&ty
   p_raportu=%report',
18.        'aRecipients' => array( "783389136",
19.        "725053503",
20.        "600623615",
21.        "506439729",
22.        "531100112" ),
23.        'iDate' => strtotime("2015-01-15 20:35"),
24.        'aUserIndexes' => array("id_dla_783389136",
```

```
25.         "id_dla_725053503",
26.         "id_dla_600623615",
27.         "id_dla_506439729",
28.         "id_dla_531100112" ),
29.       );
30.
31.       try {
32.         $result = $client->send( array( 'sLogin' => $sLogin,
33.         'sPassword' => $sPassword,
34.         'aMessages' => $aMessages,
35.         'bReturnSmsid' => true ) );
36.
37.         echo "Łączna liczba odbiorców wysyłki: " . $result-
    >{'iSmsCount'} . "<br />";
38.         echo "Koszt wysyłki: " . $result->{'iPrice'} . " groszy<br
    />";
39.
40.         foreach($result->{'vSendRecipientsResults'} as $recipient) {
41.         echo "Numer odbiorcy: <b>" . $recipient->{'sRecipient'} .
    "</b><br />";
42.         echo "ID wiadomości: " . $recipient->{'sSmsId'} . "<br />";
43.         echo "ID wiadomości (nadane przez użytkownika): " .
    $recipient->{'sOwnId'} . "<br />";
44.         echo "Status przekazania wiadomości do wysłania: " .
    $recipient->{'sStatus'} . "<br />";
45.         echo "<br />";
46.         }
47.
48.
49.       } catch(Exception $e) {
50.         echo "<b>Błąd podczas wysyłki.</b><br />";
51.         echo "Numer błedu: " . $e->faultstring . "<br />";
52.         echo "Opis błedu: " . $e->detail . "<br />";
53.       }
54.
55.
56.
57.
```

Raporty dostarczenia

Aby odpytać system o stan wysłanej wiadomości SMS, można użyć jednej z następujących metod:

- `getReportsById()` — zwraca raport wiadomości na podstawie identyfikatora wiadomości (własnego lub systemowego),
- `getReportsByMessageId()` — zwraca raport wiadomości na podstawie identyfikatora wysyłki,
- `getReportsByTime()` — zwraca raporty wiadomości w zadanym przedziale czasu.

Metoda getReportsById()

Metoda zwraca raport wiadomości na podstawie identyfikatora wiadomości (własnego lub systemowego). Przyjmuje następujące parametry wejściowe:

sLogin (type string)

Login (email) użytkownika

sPassword (type string)

Hasło użytkownika

aIndexes(type string[])

Tablica/wektor identyfikatorów wiadomości, których status wysłania chcemy sprawdzić

bSystemIndexes (type boolean)

Parametr przyjmując wartość `true` traktuje tablicę/wektor identyfikatorów wiadomości jako identyfikatory systemowe. Jeżeli natomiast jest wartość `false`, wówczas wspomniana tablica/wektor jest traktowana jako tablica/wektor stringów zawierająca identyfikatory własne, nadane przez aplikację kliencką

Przykładowy kod

```
1.
2. <?php
3. $sLogin = "login";
4. $sPassword = "haslo";
5.
6.
7. $client = new SoapClient( 'https://promosms.com/remote/soap/wsdl',
8.  array( 'features' => SOAP_SINGLE_ELEMENT_ARRAYS,
9.  'cache_wsdl' => WSDL_CACHE_NONE ));
10.       /*
11.        * Poniższe wywołanie funkcji getreportsbyid() zwróci obiekt
   wiadomości, której
12.        * wewnętrzny identyfikator w systemie to:
   5edf0d988a61e4a38dbeeef0be92fac2599b79ee
13.        */
14.
15.       // Sprawdzamy status wiadomosci na podstawie identyfikatora
   systemowego
16.
17.       $aResults = $client->getReportsById(array('sLogin' =>
   $sLogin, 'sPassword' => $sPassword, 'aIndexes' =>
   array("3191086511278f6e0e0f4751c47350941eb56411"), 'bSystemIndexes'
   => true) );
18.
19.       foreach((isset($aResults->{'vReports'}) &&
   is_array($aResults->{'vReports'}) ? $aResults->{'vReports'}
   :array()) as $message) {
20.        echo "<b>Wiadomość o identyfikatorze wewnętrznym " .
   $message->{'cSmsId'} . " ma następujące właściwości:</b><br />";
21.        echo "Odbiorca wiadomości: " . $message->{'sRecipient'} .
   "<br />";
22.        echo "Identyfikator wewnętrzny: " . $message->{'sSmsId'} .
   "<br />";
23.        echo "Identyfikator użytkownika: " . $message->{'sOwnId'} .
   "<br />";
```

```
24.         echo "Identyfikator wysyłki: " . $message->{'iMessageId'} .
   "<br />";
25.         echo "Odbiorca wiadomości: " . $message->{'sRecipient'} .
   "<br />";
26.         echo "Data i godzina ostatniej aktualizacji raportu: " .
   date("d-m-Y H:i:s", $message->{'iReportTime'}) . "<br />";
27.         echo "Status dostarczenia: ";
28.
29.         // Poniżej przedstawiamy poglądowo, który status liczbowy
   jakiemu statusowi doręczenia odpowiada
30.         switch($message->{'iReportStatus'}) {
31.         case 0:
32.         echo "przekazane do wysłania";
33.         break;
34.         case 1:
35.         echo "dostarczono";
36.         break;
37.         case 2:
38.         case 16:
39.         echo "błąd operatora/błędny numer (niedostarczone)";
40.         break;
41.         case 4:
42.         case 8:
43.         echo "wysłane, oczekuje na dostarczenie";
44.         break;
45.         }
46.         echo "<br /><br />";
47.       }
48.
49.       /*
50.         * Poniższe wywołanie wykorzystuje do sprawdzenia wysyłek nie
   identyfikator wewnętrzny wiadomości,
51.         * lecz identyfikator użytkownika przypisany do danej
   wiadomości.
52.         * Funkcja zwraca tablicę obiektów wiadomości, które mają
   dany identyfikator.
53.         * W odróżnieniu od identyfikatora wewnętrznego,
   identyfikator użytkownika może być przypisany
54.         * taki sam do dowolnej liczby wiadomości.
55.         */
56.
57.       // Sprawdzamy status wiadomosci na podstawie identyfikatora
   przypisanego przez uzytkownika
58.       try {
59.         $message = $client->getReportsById(array('sLogin' =>
   $sLogin, 'sPassword' => $sPassword, 'aIndexes' =>
   array("wlasny_identyfikator1"), 'bSystemIndexes' => false) );
60.       } catch(Exception $e) {
```

```
61.         echo "<b>Błąd podczas wysyłki.</b><br />";
62.         echo "Numer błedu: " . $e->faultstring . "<br />";
63.         echo "Opis błedu: " . $e->detail . "<br />";
64.       }
65.
66.       echo '<pre>';
67.       var_dump($message);
68.       echo '</pre>';
69.       ?>
70.
71.
```

Metoda getReportsByMessageId()

Metoda zwraca raport wiadomości na podstawie identyfikatora wysyłki. Przyjmuje następujące parametry wejściowe:

sLogin (type string)

Login (email) użytkownika

sPassword (type string)

Hasło użytkownika

iContentId (type int)

Unikalny identyfikator wiadomości, nadawany podczas wysyłki. Jest on zwracany przez metodę send w momencie, kiedy podczas wysyłki zmienna bReturnSmsid przyjmuje wartość `false` (wówczas metoda send zamiast zwracać tablicę/wektor unikalnych identyfikatorów dla każdego odbiorcy, zwraca unikalny identyfikator dla całej wysyłki)

iLimit(type int)

Opcja przydatna w przypadku, kiedy trzeba pobrać dużą liczbę statusów (np. kilkadziesiąt tysięcy). Wówczas możemy ograniczyć liczbę zwracanych elementów w tablicy/wektorze. Jeżeli ma być zwrócona tablica/wektor jednak wszystkich odbiorców wiadomości, należy podać `0`.

iOffset (type int)

Parametr ten określa, od jakiego momentu ma być zwracana tablica/wektor. Opcja przydatna w momencie, kiedy jako `iLimit` zostanie podana wartość większa niż `0`.

Przykładowy kod

```
1.
2. <?php
3. $sLogin = "login";
4. $sPassword = "haslo";
5.
6.
7. $client = new SoapClient( 'https://promosms.com/remote/soap/wsdl',
8.  array( 'features' => SOAP_SINGLE_ELEMENT_ARRAYS,
9.  'cache_wsdl' => WSDL_CACHE_NONE ));
10.       /*
11.        * Poniższe wywołanie funkcji getreportsbyid() zwróci obiekt
   wiadomości, której
12.        * wewnętrzny identyfikator w systemie to:
   5edf0d988a61e4a38dbeeef0be92fac2599b79ee
13.        */
14.
15.       /*
16.        * Poniższe wywołanie zwróci nam tablicę obiektów wszystki
   wiadomości,
17.        * których identyfikatorem wysyłki jest: 544822.
18.        * Dwa kolejne parametry to:
19.        * - iLimit - ogranicz liczbę wzwórconych elementów do iLimit
   . Wartość 0
20.        * oznacza, że chcemy otrzymac tablicę wszystkich elementów
21.        * - iOffset - zacznij wyniki od elementu iOffset. Wartość 0
   oznacza, że
22.        * chcemy otrzymać tablicę elementów "od początku"
23.        */
24.       $aMessages = $client->getReportsByMessageId( array('sLogin'
   => $sLogin, 'sPassword' => $sPassword, 'iContentId' => 1357486,
   'iLimit' => 0, 'iOffset' => 0) );
25.
26.       foreach((isset($aMessages->{'vReports'}) &&
   is_array($aMessages->{'vReports'}) ? $aMessages->{'vReports'} :
   array()) as $message) {
27.        echo "Odbiorca wiadomości: " . $message->{'sRecipient'} .
   "<br />";
```

```
28.         echo "Identyfikator wewnętrzny: " . $message->{'sSmsId'} .
    "<br />";
29.         echo "Identyfikator użytkownika: " . $message->{'sOwnId'} .
    "<br />";
30.         echo "Identyfikator wysyłki: " . $message->{'iMessageId'} .
    "<br />";
31.         echo "Odbiorca wiadomości: " . $message->{'sRecipient'} .
    "<br />";
32.         echo "Data i godzina ostatniej aktualizacji raportu: " .
    date("d-m-Y H:i:s", $message->{'iReportTime'}) . "<br />";
33.         echo "Status dostarczenia: ";
34.
35.         // Poniżej przedstawiamy poglądowo, który status liczbowy
    jakiemu statusowi doręczenia odpowiada
36.         switch($message->{'iReportStatus'}) {
37.         case 0:
38.         echo "przekazane do wysłania";
39.         break;
40.         case 1:
41.         echo "dostarczono";
42.         break;
43.         case 2:
44.         case 16:
45.         echo "błąd operatora/błędny numer (niedostarczone)";
46.         break;
47.         case 4:
48.         case 8:
49.         echo "wysłane, oczekuje na dostarczenie";
50.         break;
51.         }
52.       }
53.       ?>
54.
55.
```

Metoda getReportsByTime()

Metoda zwraca raporty wiadomości w zadanym przedziale czasu. Przyjmuje następujące parametry wejściowe:

sLogin (type string)

Login (email) użytkownika

sPassword (type string)

Hasło użytkownika

iTimeFrom(type int)

Czas (w postaci uniksowego znacznika czasu), od którego momentu chcemy pobrać informację o statusach wysłanych wiadomości

iTimeTo (type int)

Czas (w postaci uniksowego znacznika czasu), do którego momentu chcemy pobrać informację o statusach wysłanych wiadomości

Przykładowy kod

```
1.
2. <?php
3. $sLogin = "login";
4. $sPassword = "haslo";
5. $client = new SoapClient( 'https://promosms.com/remote/soap/wsdl',
6.  array( 'features' => SOAP_SINGLE_ELEMENT_ARRAYS,
7.  'cache_wsdl' => WSDL_CACHE_NONE ));
8. /*
9.  * Poniższe wywołanie funkcji getreportsbyid() zwróci obiekt
   wiadomości, której
10.         * wewnętrzny identyfikator w systemie to:
   5edf0d988a61e4a38dbeeef0be92fac2599b79ee
11.         */
12.
```

```
13.       /*
14.        * Poniższe wywołanie zwróci nam tablicę obiektów wszystki
   wiadomości,
15.        * które zostały wysłane od 2011-11-14 00:00 (czyli
   1321225200)
16.        * do 2011-11-15 15:00 (czyli 1321365600).
17.        */
18.       $aMessages = $client->getReportsByTime( array('sLogin' =>
   $sLogin, 'sPassword' => $sPassword, 'iTimeFrom' => 1321225200,
   'iTimeTo' => 1321365600) );
19.
20.       foreach((isset($aMessages->{'vReports'}) &&
   is_array($aMessages->{'vReports'}) ? $aMessages->{'vReports'} :
   array()) as $message) {
21.        echo "Odbiorca wiadomości: " . $message->{'sRecipient'} .
   "<br />";
22.        echo "Identyfikator wewnętrzny: " . $message->{'sSmsId'} .
   "<br />";
23.        echo "Identyfikator użytkownika: " . $message->{'sOwnId'} .
   "<br />";
24.        echo "Identyfikator wysyłki: " . $message->{'iMessageId'} .
   "<br />";
25.        echo "Odbiorca wiadomości: " . $message->{'sRecipient'} .
   "<br />";
26.        echo "Data i godzina ostatniej aktualizacji raportu: " .
   date("d-m-Y H:i:s", $message->{'iReportTime'}) . "<br />";
27.        echo "Status dostarczenia: ";
28.
29.        // Poniżej przedstawiamy poglądowo, który status liczbowy
   jakiemu statusowi doręczenia odpowiada
30.        switch($message->{'iReportStatus'}) {
31.        case 0:
32.        echo "przekazane do wysłania";
33.        break;
34.        case 1:
35.        echo "dostarczono";
36.        break;
37.        case 2:
38.        case 16:
39.        echo "błąd operatora/błędny numer (niedostarczone)";
40.        break;
41.        case 4:
42.        case 8:
43.        echo "wysłane, oczekuje na dostarczenie";
44.        break;
45.        }
46.       }
47.       ?>
```

Dane wyjściowe - struktura Report

Wszystkie metody pobierające raporty dostarczenia w rezultacie swojego działania zwracają strukturę `Report`, której definicja jest następująca:

sRecipient (type string)

Numer telefonu odbiorcy SMSa

sSmsId (type string)

Unikalny systemowy identyfikator dla SMSa wysłanego do danego odbiorcy

sOwnId (type string)

Opcjonalny identyfikator dla SMSa wysłanego do danego odbiorcy, nadany przez aplikację kliencką

iMessageId (type int)

Unikalne ID wiadomości, która została wysłana do danego odbiorcy

iReportStatus (type int)

Status wysłania wiadomości. W dalszej części dokumentacji znajduje się tabela opisująca dokładnie znaczenie każdego zwracanego statusu

iReportTime (type int)

Czas (w postaci uniksowego znacznika czasu) otrzymania danego statusu wysłania wiadomości (`iReportStatus`)

Odbieranie odpowiedzi

W przypadku realizacji wysyłki EkoSMS, istnieje możliwość, aby odbiorca wiadomości odpowiedział na nią. Odpowiedzi takie dostępne są zarówno przez Panel SMS jak i poprzez SOAP API.

Odbiór wiadomości odbywa się poprzez wywołanie metody `getMessages()` z odpowiednimi parametrami wejściowymi:

sLogin (type string)

Identyfikator (adres e-mail) użytkownika

sPassword (type string)

Hasło użytkownika w postaci jawnej lub skrótu `md5()`

iLastMessageId (type int)

Każda odebrana wiadomość SMS ma swój unikalny identyfikator w postaci liczby. Jeżeli chcemy odbierać wiadomość tylko od danej wiadomości o znanym nam identyfikatorze - należy go tutaj podać. Jeżeli więc wcześniej przy odbiorze wiadomości ostatnia odczytana wiadomość miała numer 359 i teraz chcemy, aby funkcja zwróciła tylko nowsze wiadomości - to jako wartość tego parametru należy podać `359`. Jeżeli system ma zwrócić wszystkie wiadomości przychodzące - parametr powinien przyjąć wartość `0`

iTimeFrom (type int)

Jeżeli zamiast podawania identyfikatora wiadomości chcemy pobrać wszystkie wiadomości od danego momentu to należy go tutaj podać w postaci uniksowego znacznika czasu. Jeżeli system ma zwrócić wszystkie wiadomości przychodzące od początku - parametr powinien przyjąć wartość `0`

Metoda `getMessages()` w rezultacie swojego działania zwracają strukturę, której definicja jest następująca:

sSender (type string)

Numer GSM, który przysłał wiadomość

sSim (type string)

Numer GSM, na który została dostarczona wiadomość

sMessage (type string)

Treść odebranej wiadomości SMS

iMsgId (type int)

Unikalny identyfikator odbieranej wiadomości SMS

iReceiveTime (type int)

Czas (w postaci uniksowego znacznika czasu) odbioru danej wiadomości SMS

Przykładowy kod

```
1.
2. <?php
3. $sLogin = "login";
4. $sPassword = "haslo";
5.
6.
7. $client = new SoapClient( 'https://promosms.com/remote/soap/wsdl',
8.  array( 'features' => SOAP_SINGLE_ELEMENT_ARRAYS,
9.  'cache_wsdl' => WSDL_CACHE_NONE ));
10.       /*
11.        * Poniższe wywołanie funkcji getMessages() zwróci wszystkie
   wiadomości odebrane dla
12.        * danego uzytkownika. Parametr trzeci pozwala określić ID
   ostatniej odebranej
13.        * wiadomości (jeżeli np. wcześniej ostatnia odebrana
   wiadomość miała ID 270, to
14.        * teraz podając jako trzeci parametr 270 - zostaną pobrane
   wszystkie nowsze
15.        * wiadomości od wiadomości o ID 270). Parametr czwarty
   pozwala natomiast pobrać
16.        * wszystkie wiadomości, które zostały odebrane po podanym (w
   postaci uniksowego
17.        * znacznika czasu) wiadomości.
18.        */
19.
```

```
20.       // Sprawdzamy status wiadomosci na podstawie identyfikatora
  systemowego
21.       $aMessages = $client->getMessages( array('sLogin' => $sLogin,
  'sPassword' => $sPassword, 'iLastMessageId' => 0, 'iTimeFrom' => 0
  ));
22.
23.
24.       foreach((isset($aMessages->{'vReceivedMessages'}) &&
  is_array($aMessages->{'vReceivedMessages'}) ? $aMessages-
  >{'vReceivedMessages'} : array()) as $message) {
25.         echo "Nadawca wiadomości: " . $message->{'sSender'} . "<br
  />";
26.         echo "Numer GSM, na który przyszła odpowiedź: " . $message-
  >{'sSim'} . "<br />";
27.         echo "Treść wiadomości: " . $message->{'sMessage'} . "<br
  />";
28.         echo "Identyfikator wiadomości: " . $message->{'iMsgId'} .
  "<br />";
29.         echo "Czas odbioru wiadomości: " . date("d-m-Y H:i:s",
  $message->{'iReceiveTime'}) . "<br />";
30.       }
31.       ?>
32.
33.
```

Sprawdzanie stanu konta

Sprawdzanie stanu konta odbywa się poprzez wywołanie metody `getAccountInfo()` z odpowiednimi parametrami wejściowymi:

sLogin (type string)

Identyfikator (adres e-mail) użytkownika

sPassword (type string)

Hasło użytkownika w postaci jawnej lub skrótu `md5()`

Metoda `getAccountInfo()` w rezultacie swojego działania zwraca strukturę, której definicja jest następująca:

iBaseBalance (type int)

Stan konta klienta wyrażony w całkowitej liczbie groszów netto (czyli jeżeli zwracana wartość to 100, tj. 1 zł netto)

iBonusBalance (type int)

Stan środków darmowych/promocyjnych przyznanych użytkownikowi (wyrażony w groszach)

iCreditBalance (type int)

Przyznany użytkownikowi kredyt (wyrażony w groszach). Jeżeli więc zwracana jest np. wartość 1000, to oznacza to, iż dozwolony jest ujemny stan konta do wysokości 10 zł netto

Przykładowy kod

```
1.
2. <?php
3. $sLogin = "login";
4. $sPassword = "haslo";
5.
6.
7. $client = new SoapClient( 'https://promosms.com/remote/soap/wsdl',
8.  array( 'features' => SOAP_SINGLE_ELEMENT_ARRAYS,
9.  'cache_wsdl' => WSDL_CACHE_NONE ));
10.       /*
11.        * Poniższa funkcja ma za zadanie:
12.        * - zwrócić informacje o stanie konta
13.        * - zwrócić informację o środkach darmowych
14.        * - zwrócić informację o kwocie poniżej dostępnych środków
   (kredyt)
15.        */
16.
17.       try {
18.        $aAccountInfo = $client->getAccountInfo( array('sLogin' =>
   $sLogin, 'sPassword' => $sPassword) );
19.        echo "Stan środków podstawowych: " . $aAccountInfo-
   >{'iBaseBalance'} . "<br />";
20.        echo "Stan środków bonusowych: " . $aAccountInfo-
   >{'iBonusBalance'} . "<br />";
21.        echo "Przyznany kredyt: " . $aAccountInfo-
   >{'iCreditBalance'} . "<br />";
```

```
22.       } catch(Exception $e) {
23.        echo "<b>Błąd podczas sprawdzania.</b><br />";
24.        echo "Numer błedu: " . $e->faultstring . "<br />";
25.        echo "Opis błedu: " . $e->detail . "<br />";
26.       }
27.       ?>
28.
29.
```

Konwertowanie wiadomości

Wiadomość przy wysyłce jest konwertowana do standardu GSM, a następnie obliczana jest ilość wiadomości składowych oraz koszt realizacji takiej wysyłki. Konwersja taka odbywa się poprzez wywołanie metody`convertMessage()` z odpowiednimi parametrami wejściowymi:

sLogin (type string)

Identyfikator (adres e-mail) użytkownika

sPassword (type string)

Hasło użytkownika w postaci jawnej lub skrótu `md5()`

sContent (type string)

Treść wiadomości, na podstawie której ma być obliczona liczba znaków oraz liczba SMSów składowych

bSpecialChars (type bool)

Jeżeli parametr przyjmie wartość `true`, wówczas znaki specjalne są dozwolone w wiadomości i nie zostaną automatycznie zastąpione na swoje odpowiedniki w znakach standardowych. Jeżeli natomiast zostanie ustawiona wartość `false`, to wówczas wszelkie znaki specjalne zostaną przekonwertowane i dopiero wówczas zostanie obliczona długość wiadomości i liczba SMSów składowych.

Metoda `convertMessage()` w rezultacie swojego działania zwraca strukturę, której definicja jest następująca:

iCharsNumber (type int)

Łączna liczba znaków wiadomości

iSmsParts (type int)

Liczba SMSów składowych

sContent (type string)

Treść wiadomości. Jeżeli w parametrach wejściowych bSpecialChars przyjęło wartość `false`, to wówczas zostanie tutaj zwrócona nie oryginalna treść wiadomości, lecz wiadomość przekonwertowana

Przykładowy kod

```
1.  <?php
2. $text = "To jest sobie jakaś treść z polskimi literkami. Treść jest specjalnie dłuższa,
3. aby pokazać, że SMSy z polskimi znakami są inaczej liczone. Tak samo np. znak ENTER liczy się
4. jako dwa znaki (zawsze).";
5.
6. // drugi parametr określa, czy znaki specjalne mają być usunięte czy nie
7.
8. echo "<b>W poniższej wiadomości znaki specjalne NIE zostały 'wycięte':</b><br />";
9. // z poniższej wiadomości NIE zostaną usuniete znaki specjalne
10.       $result = $client->convertMessage( array('sLogin' => $sLogin, 'sPassword' => $sPassword, 'sContent' => $text, 'bSpecialChars' => true) );
11.       echo "Ilość znaków w wiadomości: " . $result->{'iCharsNumber'} . "<br />";
12.       echo "Liczba składowych wiadomości SMS: " . $result->{'iSmsParts'} . "<br />";
13.       echo "Właściwa treść wiadomości: " . $result->{'sContent'} . "<br />";
14.
15.       echo "<br /><b>W poniższej wiadomości znaki specjalne zostały 'wycięte':</b><br />";
16.       // z poniższej wiadomości zostaną usuniete znaki specjalne
```

```
17.       $result = $client->convertMessage( array('sLogin' => $sLogin,
   'sPassword' => $sPassword, 'sContent' => $text, 'bSpecialChars' =>
   false) );
18.       echo "Ilość znaków w wiadomości: " . $result-
   >{'iCharsNumber'} . "<br />";
19.       echo "Liczba składowych wiadomości SMS: " . $result-
   >{'iSmsParts'} . "<br />";
20.       echo "Właściwa treść wiadomości: " . $result->{'sContent'} .
   "<br />";
21.
22.       $text = "Należy pamiętać, że niektóre znaki (np. ENTER)
   zawsze są liczone jako dwa znaki.
23.       Ta wiadomość ma 139 znaków, ale dla GSM ma ona 140 znaków.";
24.
25.       echo "<br /><b>Pokazanie, że są znaki specjalne liczone jako
   dwa znaki:</b><br />";
26.       // z poniższej wiadomości zostaną usuniete znaki specjalne
27.       $result = $client->convertMessage( array('sLogin' => $sLogin,
   'sPassword' => $sPassword, 'sContent' => $text, 'bSpecialChars' =>
   false) );
28.       echo "Ilość znaków w wiadomości: " . $result-
   >{'iCharsNumber'} . "<br />";
29.       echo "Liczba składowych wiadomości SMS: " . $result-
   >{'iSmsParts'} . "<br />";
30.       echo "Właściwa treść wiadomości: " . $result->{'sContent'} .
   "<br />";
31.
32.       ?>
```

Walidowanie numeru

Metody `validateNumbers()` umożliwia sprawdzanie poprawności numeru pod względem syntaktycznym. Sprawdzane jest więc, czy np. numer 531100112 ma poprawną formę i jest możliwym numerem w polskich sieciach GSM, czy nie. Funkcja dodatkowo zwraca następujące informacje:

- czy numer jest prawidłowy
- czy numer jest stacjonarny, komórkowy itp.
- czy numer jest poprawny dla danego regionu (domyślnie - dla Polski)
- czy numer jest prawdopodobnie prawdziwy

- jaki jest prawdopodobny region świata, do którego należy dany numer (np. Polska, Wielka Brytania, Ukraina itp.)
- sformatowania postać numeru telefonu (czyli np. numer w postaci +48 531-100-11-2 zostanie przekształcony na 48531100112)
-

  sLogin (type string)

  Identyfikator (adres e-mail) użytkownika

  sPassword (type string)

  Hasło użytkownika w postaci jawnej lub skrótu `md5()`

  iCountryCode (type int)

  Domyślny numer kierunkowy kraju, dla którego sprawdzane są numery (np. dla Polski jest to 48)

  vNumbersToValidate (type string[])

  Tablica/wektor numerów telefonicznych, które aplikacja kliencka chce sprawdzić

Metoda `validateNumbers()` w rezultacie swojego działania zwracają strukturę, której definicja jest następująca:

sOrginalNumber (type string)

Oryginalna postać numeru, którego dotyczy zapytanie

bValidNumber (type boolean)

Czy numer jest prawidłowy

bValidForRegionNumber (type boolean)

Czy numer ma prawidłową postać dla regionu świata, który został podany na wejściu funkcji (parametr`iCountryCode`)

bPossibleNumber (type boolean)

Czy numer jest prawdopodobnie prawdziwy

iRegionId (type int)

Kod regionu świata, do którego najprawdopodobniej należy dany numer

sRegionName (type string)

Postać znakowa danego regionu świata (np. dla Polski - `PL`)

sFormattedNumber (type string)

sformatowana postać numeru, który poprawnie przeszedł walidację. Dla niepoprawnych numerów zwracany jest string `0000`

iTypeOfNumber (type int)

Typ numeru

Przykładowy kod

```
1.
2. <?php
3. $sLogin = "login";
4. $sPassword = "haslo";
5.
6.
7. $client = new SoapClient( 'https://promosms.com/remote/soap/wsdl',
8.  array( 'features' => SOAP_SINGLE_ELEMENT_ARRAYS,
9.  'cache_wsdl' => WSDL_CACHE_NONE ));
10.       // Type of phone numbers.
11.       $PhoneNumberType = array();
12.       // numer stacjonarny
13.       $PhoneNumberType[0] = "FIXED_LINE";
14.       // numer komórkowy
15.       $PhoneNumberType[1] = "MOBILE";
16.       // W niektórych krajach możliwe jest, że dany numer jest
   równocześnie numerem
17.       // komórkowym/stacjonarnym
18.       $PhoneNumberType[2] = "FIXED_LINE_OR_MOBILE";
19.       // Darmowy numer
```

```
20.       $PhoneNumberType[3] = "TOLL_FREE";
21.       // Number o podwyższonej płatności
22.       $PhoneNumberType[4] = "PREMIUM_RATE";
23.       // Koszt za połączenie jest współdzielony między odbiorcą i
   nadawcą (np. numer typu 801..)
24.       $PhoneNumberType[5] = "SHARED_COST";
25.       // Numer VoIP
26.       $PhoneNumberType[6] = "VOIP";
27.       // Numer osobisty, możliwe że jest równocześnie komórkowy jak
   i stacjonarny
28.       // patrz tutaj: http://en.wikipedia.org/wiki/Personal_Numbers
29.       $PhoneNumberType[7] = "PERSONAL_NUMBER";
30.       // Pager
31.       $PhoneNumberType[8] = "PAGER";
32.       // tzw. "Universal Access Numbers" or "Company Numbers"
33.       // Jeden numer firmowy, ale połączenie może być odpowiednio
   przełączane
34.       $PhoneNumberType[9] = "UAN";
35.       // Nieznany typ numeru
36.       $PhoneNumberType[10] = "UNKNOWN";
37.
38.
39.       /*
40.        * Poniższa tablica zawiera różnego rodzaju numery telefonów,
41.        * zarówno poprawne numery komórkowe, jak również numery
   stacjonarne
42.        * i numery w ogóle nieprawidłowe.
43.        */
44.       $aNumbers = array(
45.        "48783389136",
46.        "0048531100112",
47.        "(77)54-53-363",
48.        "48506439729",
49.        "600623615",
50.        "726-505-312",
51.        "+48 695 275 208",
52.        "725 05 35 03",
53.        "+385 (0)99 251 7416",
54.        "725 05 35 03",
55.        "+44 798 152 00 40",
56.        "44 798 152 00 40",
57.        "00 (44) 798 152 00 40",
58.        "112",
59.        "997",
60.        "0 801 300 800",
61.        "801 300 800",
62.        "+48 (prefiks) 42 6 300 800",
63.        "+30 23940 20336",
```

```
64.         "30 23940 20336",
65.         "783389136",
66.         "774552190",
67.         "+48774552190",
68.         "226301123",
69.         "(43)8281060",
70.         "+1 (312) 337-8166",
71.         "800 800 098"
72.         );
73.
74.
75.        /*
76.         * Poniższa funkcja ma za zadanie:
77.         * - sformatować wszystkie numery do jednego, prawidłowego
   formatu,
78.         * - usunąć wszelkie numery nieprawidłowe (w tym stacjonarne)
79.         */
80.
81.        try {
82.         $aVerifiedNumbers = $client->validateNumbers( array(
   'sLogin' => $sLogin, 'sPassword' => $sPassword, 'vNumbersToValidate'
   => $aNumbers, 'iCountryCode' => 48) );
83.        } catch(Exception $e) {
84.         echo "<b>Błąd podczas sprawdzania.</b><br />";
85.         echo "Numer błedu: " . $e->faultstring . "<br />";
86.         echo "Opis błedu: " . $e->detail . "<br />";
87.        }
88.
89.        foreach((isset($aVerifiedNumbers->{'vValidatedNumbers'}) &&
   is_array($aVerifiedNumbers->{'vValidatedNumbers'}) ?
   $aVerifiedNumbers->{'vValidatedNumbers'} : array()) as $oNumber) {
90.         echo "sOrginalNumber: " . $oNumber->{'sOrginalNumber'} .
   "<br />";
91.         echo "bValidNumber: " . $oNumber->{'bValidNumber'} . "<br
   />";
92.         echo "bValidForRegionNumber: " . $oNumber-
   >{'bValidForRegionNumber'} . "<br />";
93.         echo "bPossibleNumber: " . $oNumber->{'bPossibleNumber'} .
   "<br />";
94.         echo "iTypeOfNumber: " . $PhoneNumberType[$oNumber-
   >{'iTypeOfNumber'}] . "<br />";
95.         echo "iRegionId: " . $oNumber->{'iRegionId'} . "<br />";
96.         echo "sRegionName: " . $oNumber->{'sRegionName'} . "<br />";
97.         echo "sFormattedNumber: " . $oNumber->{'sFormattedNumber'} .
   "<br />";
98.         echo "=======================================<br />";
99.        }
100.       ?>
```

Dodawanie kontaktów

SOAP API umożliwia dodawanie kontaktów do Książki Adresowej za pomocą metody `addContacts()`, która przyjmuje następujące parametry:

sLogin (type string)

Identyfikator (adres e-mail) użytkownika

sPassword (type string)

Hasło użytkownika w postaci jawnej lub skrótu `md5()`

vContacts (type Contact)

Tablica/wektor obiektów typu Contact. Każdy obiekt tego typu reprezentuje kontakt i składa się z następujących właśności:

sName (type string)

Nazwa kontaktu

sPhone (type string)

Numer telefonu

vLabels (type string)

Lista identyfikatorów etykiet rozdzielonych przecinkiem (np. `klienci,vip`)

Przykładowy kod

```
1.
2. <?php
3. $sLogin = "login";
4. $sPassword = "haslo";
5.
6.
7. $client = new SoapClient( 'https://promosms.com/remote/soap/wsdl',
8.  array( 'features' => SOAP_SINGLE_ELEMENT_ARRAYS,
```

```
9.  'cache_wsdl' => WSDL_CACHE_NONE ));
10.
11.
12.
13.       try {
14.
15.
16.       $aContacts = array(
17.        'sName' => 'Testowy kontakt',
18.        'sPhone' => '504304204'
19.        );
20.
21.
22.       $aLabels = 'klienci,vip';
23.
24.
25.       $result = $client->addContacts( array('sLogin' => $sLogin,
26.        'sPassword' => $sPassword,
27.        'vContacts' => $aContacts,
28.        'vLabels' => $aLabels) );
29.
30.
31.       } catch(Exception $e) {
32.        echo "Numer błedu: " . $e->faultstring . "<br />";
33.        echo "Opis błedu: " . $e->detail . "<br />";
34.       }
35.       ?>
36.
```

Statusy końcowe operacji

Za każdym razem po wysłaniu wiadomości poprzez HTTP/SOAP SSL API, zwracany jest status wysyłki. Jest on potwierdzeniem, czy wiadomość została wysłana prawidłowo. Poniżej znajduje się lista możliwych zwracanych statusów wraz z opisem.

Status zwracane przy wysyłaniu wiadomości SMS

Status	Opis statusu
001	Wysłano prawidłowo
002	Wiadomość prawidłowo przekazana do wysyłki, jednak z przyczyn technicznych zostanie wysłana w późniejszym czasie (dotyczy FasterSMS)
011	Nie przekazano potrzebnych parametrów - brak danych GET/POST
021	Zły login lub hasło
022	Brak wystarczających środków na koncie do realizacji tego typu wysyłki
023	Problem z wyslaniem wiadomości. Skontaktuj się z BOK.
031	Pole nadawcy jest zbyt długie (maksymalnie 16 znaków przy nadawcy numerycznym i 11 znaków przy nadawcy alfanumerycznym)
032	Nie podałeś/-aś ani jednego adresata wiadomości lub podany adresat jest nieprawidłowy (nieprawidłowy/nieistniejący numer komórkowy lub numer stacjonarny)

033	Nie podałeś/-aś treści wiadomości
034	Podano czas z przeszłości jako preferowany czas wysyłki
035	Liczba identyfikatorów własnych jest różna od liczby odbiorców wysyłanej wiadomości
036	Został wybrany typ wiadomości MaxSMS, ale nie został określony nadawca wiadomości (maksymalnie 11 znaków alfanumerycznych)
037	Zezwalasz tylko na wysyłanie wiadomości składających się z jednego SMSa, a Twoja wiadomość jest dłuższa
038	Treść wiadomości przekracza maksymalną dozwoloną pod względem technicznym długość (więcej niż 4 SMSy składowe na wiadomość)i
039	Wybrałeś wysyłkę typu FasterSMS i równocześnie próbujesz wysłać wiadomość do więcej niż jednego adresata, co jest niedozwolone (FasterSMS służy do wysyłania pojedynczych wiadomości SMS, np. z hasłem jednorazowym)
040	Nazwa nadawcy wiadomości nie została zarejestrowana. Skontaktuj się z BOK.
080	Adres IP, z którego przyszło wywołanie API nie znajduje się na liście adresów uprawnionych do korzystania z API. Zaloguj się do Panelu SMS, aby skonfigurować ustawienia API.
041	Wiadomość zawiera znaki spoza dozwolonego zakresu znaków UTF-8.
091	Błąd połączenia z bazą. Skontaktuj się z firmą PromoSMS

092 Błąd podczas wysyłki wiadomości. Skontaktuj się z firmą PromoSMS