SMSarena.pl Spółka z ograniczoną odpowiedzialnością

Dokumentacja interfejsu API XML

Ver. 1.0 2012

1.Zawartość dokumentu

Niniejszy dokument zawiera informacje na temat sposobu komunikowania się z platformą SMSarena.pl korzystając z „interfejsu API XML".
Rozdział 2 zawiera informacje o interfejsie, sposobie i parametrach połączenia. Niezastosowanie się do nich może spowodować problemy z korzystaniem z interfejsu.
W rozdziale 3 znajdują się szczegółowe informacje na temat formatowania wiadomości wysyłanych i odbieranych z serwera w ramach komunikacji zplatformą.
W rozdziale 4 znajdziecie Państwo przykłady wykorzystania typowych narzędzi do budowania aplikacji klienckich.

1.Ogólne informacje o interfejsie

2.1 Zastosowanie

Opisany w dokumencie interfejs służy do wysyłania i odbierania SMS-ów za pomocą platformy SMSarena.pl. Opisany protokół ma zastosowanie jedynie dla platformy SMSarena.pl (https://www.smsarena.pl) i stanowi jej cześć składową.

2.2 Wersja protokołu

Zestaw komend opisanych w niniejszym dokumencie jest stały dla danej wersji protokołu. Następne wersje protokołu nie będą musiały wspierać poprzedniej wersji protokołu. Rozszerzania protokołu o nowe wersje nie będą miały wpływu na wersje poprzednie o ile ich wsparcie nie wygaśnie, o czym odpowiednio wcześnie klient zostanie poinformowany.
Niniejszy dokument opisuje wersję 1.0 protokołu.

2.3 Połączenie

Interfejs wykorzystuje połączenie HTTPS i zestawiane jest w taki sam sposób w jaki przeglądarka WWW łączy się z serwisem zabezpieczonym przy pomocy HTTPS. Połączenie powinno być zestawiane raz na cały okres komunikacji. Aplikacja kliencka powinna połączyć się ponownie, gdy z dowolnego powodu nastąpi przerwanie połączenia. W zależności od narzędzi wykorzystywanych przez program/skrypt kliencki oprogramowanie połączenia może wyglądać różnie. Przykłady zestawienia połączenia znajdują się w rozdziale 4.
Parametry połączenia potrzebne do zestawienia połączenia, czyli:

- adres – adres IP lub nazwa DNS serwera do jakiego należy się podłączyć,
- port – numer portu do jakiego łączymy się z serwerem,
- url – ścieżka do skryptu obsługującego interfejs (w naszym wypadku „/xml_post") znaleźć można w panelu (menu „Usługi")

Pełny url może wyglądać jak poniżej:
https://api.smsarena.pl/xml_post

UWAGA: Interfejs aktywuje się jedynie, gdy otrzyma w parametrze **POST** zmienną **„xml"**

3 Specyfikacja protokołu API XML

3.1 Komunikacja z serwerem

Interfejs działa w architekturze klient-serwer. Do aplikacji klienckiej należy obowiązek zainicjowania i utrzymania połączenia. Zaleca się, aby przy ciągłym odpytywaniu serwera, połączenie było zestawione przez cały czas.
Po zestawieniu połączenia komunikacja odbywa się za pomocą wywołań POST protokołu HTTP. Wszystkie niezbędne dla wykonania komendy informacje są przesyłane za pomocą parametru ***xml*** umieszczanego w wywołaniu jako zmienna POST. Wspomniany parametr *xml* musi zawierać tekst dokumentu xml.

Tekst xml-owy musi być poprawnie sformatowany i nie powinien zawieraćdodatkowych, zbędnych i niezrozumiałych dla serwera tagów (elementów iatrybutów XML).

Uwaga: Tekst w zmiennej xml, nie może być „escape'owany" jak to byłoby przy parametrze wysyłanym przy pomocy metody GET.

Wiadomości odebrane z serwera również są sformatowane do postaci pliku xml. Jeśli serwer dostanie w wywołaniu POST zmienną *xml*, niezależnie od tego, czy wiadomość wysyłana w tej zmienne ma prawidłową postać, zwrot będzie typu *text/xml* i będzie zawierał wynik wykonania wysłanego polecenia lub w przypadku błędu użytkownika informację o błędzie.

3.2 Kodowanie znaków

Dane w wywołaniu POST (tekst xml) muszą zostać zakodowane w standardzie UTF-8. Zaleca się, żeby dane w zmiennej *xml* zaczynały się od następującego nagłówka:

<?xml version="1.0" encoding="UTF-8"?>

3.3 Autoryzacja

Dane autoryzacyjne (wskazanie użytkownika i hasła dostępu) są niezbędne do wykonania każdego z poleceń. Informacje te zawiera się przy każdym wywołaniu wysyłanym do serwera, a zamieszcza się je w atrybutach elementu *request.* Poniżej fragment kodu XML, zawierający poprawnie wstawione parametry autoryzacyjne:
<request version="1.0" login="htguser123" pass="8TyMI8LN" action="action_string">

Wartości parametrów autoryzacyjnych dostępne są w panelu administracyjnym.

Zmienne dla parametru ***action.***

- ***send*** – parametr do wysyłki SMS-ów,
- ***status*** – parametr do pobierania statusów sms-ów,
- ***sms_in*** – parametr do odbioru sms-ów przychodzących

3.4 Ramka komunikatu

Każda wiadomość wysłana i odebrana z serwera ma postać dokumentu XML. Dokument XML powinien zawierać nagłówek postaci:
<?xml version="1.0" encoding="UTF-8"?>
Wiadomości wysyłane do serwera opakowane są za pomocą elementu *request*, a wiadomości odebrane za pomocą elementu *response*.

3.4.1 Request komenda wysyłana do serwera

Komenda wysyłana do serwera musi mieć postać:

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.0" login="user_name" pass="pass_string" action="action_string">
...
...
</request>
```

Wewnątrz elementu <*request*> zamieścić należy parametr wywołania zależne od wybranej komendy.
Wszystkie atrybuty elementu <*request*> są obowiązkowe.

3.4.2 Response odpowiedź z serwera

Odpowiedź z serwera ma postać:

```
<?xml version="1.0" encoding="UTF-8"?>
<response version="1.0" code="code_int" code_body="code_string" action="action_string" >
...
</response>
```

Wewnątrz elementu <*response*> znajdują się wyniki wykonania wywoływanej komendy.
Zmienne *code* oraz *code_body* zawierają informacje o aktualnym statusie zapytania.

3.4.3 Odpowiedź z informacją o statusie zapytania

W przypadku źle sformatowanego polecenia lub wystąpieniu innych problemów odpowiedź z serwera zmienna *code* oraz *code_body* zawiera następujące informacje:

code	code_body
200	SMS-y zostały zakolejkowane
201	Pobrano statusy
100	Błędne dane logowania
101	Nie otrzymano numerów id dla statusów
102	Nie znaleziono żadnych rekordów
103	Nie podano daty

3.5 Komendy

3.5.1 Wysłanie SMS a

send

Akcja *send* służy do zainicjowania wysłania SMS. Funkcja nie oczekuje na wysłanie SMS-a. Odpowiedź z serwera przychodzi w najkrótszym możliwym czasie. Informacja o tym, czy i kiedy SMS został wysłany lub odebrany przez telefon na jaki SMS został nadany, dostępna jest za pomocą akcji *status.* Funkcja zwraca wewnętrzny dla platformy identyfikator SMS-a. Identyfikator ten używany jest przy wykonywaniu akcji *status*.

Uwaga: SMS zostanie wysłany jedynie w godzinach oznaczonych jako okres wysyłania SMS (więcej w Ustawieniach w Panelu Administracyjnym)

3.5.1.1 Komenda (send_sms)

Format wiadomości:

```
        <?xml version="1.0" encoding="UTF-8"?>
        <request version="1.0" login="user_name" pass="password_string" action="action_string">
<send_sms>
    <id_user>Wartość dowolna</id_user>
    <msisdn>Numer_MSISDN</msisdn>
    <body>Tutaj treść SMS-a</body>
    <sender>Nadawca</sender>
    <type_sms>Kod typu smsa</type_sms>
    <send_after>yyyy-mm-dd hh:mi:ss</send_after>
    </send_sms>
</request>
```

Elementy obowiązkowe:

- **<msisdn>** – numer MSISDN. Platforma SMSarena.pl powinna zaakceptować większość możliwych formatów o ile numer jest napisany jednoznacznie. Przykładowe formaty, które zostaną zaakceptowane:
  - 601 234 567
  - +48 601-234-567
  - 601234567
  - +48601234567
  - 601234567
- **<body>** – treść SMS-a. Maksymalna liczba znaków jaką można przesłać jest równa 160 (dla wiadomości jednosmsowych) lub 459 (dla wiadomości wieloczęściowych - wysłane zostaną 3 SMS-y) . Jeśli w tekście znajdą się polskie znaki i chcecie Państwo je przesłać, nie wolno przekroczyć 70 znaków (201 - dla wieloczęściowych

Elementy opcjonalne:

- **<id_user>** - W tym polu można zamieści własne informacje o SMS-ie, co mogło by ułatwić rozpoznanie we własnych bazach
- **<send_after>** – data planowanej wysyłki SMS. Platforma wyśle SMSa o wyznaczonej godzinie niezależnie od ustawień limitów w Panelu administratorskim. Jeśli parametr nie zostanie podany SMS zostanie wysłany w najbliższym czasie o ile limity wysyłki nie kolidują zaktualnym czasie.
- **<sender>** – Ustawia inne niż domyślne pole nadawcy dla SMS-ów w usłudze typu "SMSpro". Pominięcie pola spowoduje użycie domyślnego pola nadawcy.

Uwaga! Nowe pola nadawcy muszą zostać zgłoszone/dodane w panelu administracyjnym. Użycie pola, które nie zostało wcześniej dodane lub nie zostało zaakceptowane przez serwis, spowoduje użycie domyślnego pola.

- **<type_sms>** – typ SMS-a (zwyczajny, lub wyskakujący SMS, tzw. „flash sms") .Kody typów:
  - n - zwykły SMS – wartość domyślna
  - f – wyskakujący SMS („flash sms")

- w – SMS WAPPUSH aby wysłać należy w <ulr> podać adres www, a w <body> opis

3.5.1.2 Odpowiedź

Format wiadomości

```
<?xml version="1.0" encoding="UTF-8"?>
<response version="1.0" code=”code_int” code_body=”code_string” action=”action_string” >
<send_sms>
<id></id>
<id_user>Wartosć dowolna</id_user>
<msisdn> Numer_MSISDN </msisdn>
<status>Status Wiadomości</status>
<status_code>Kod statusu</status_code>
<send_at>Data Wysłania</send_at>
<delivered_at>Data Dostarczenia</delivered_at>
<updated_at>Data aktualizacji rekordu</updated_at>
</send_sms>
</response>
```

3.5.1.3 Przykłady

Wysłanie SMS o treści „Przesyłka numer 123456 została nadana.” na numer +48601234567

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.0" login="user_name" pass="password_string" action=”send”>
<send_sms>
<msisdn>+48601234567</msisdn>
<body> Przesyłka numer 123456 została nadana. </body>
</send_sms>
</request>
```

Wysłanie SMS o treści „Przesyłka numer 123456 została nadana.” na numer 601234567 jako wiadomość flash o wyznaczonej godzinie. SMS zostanie wysłany z polem nadawcy "SMSarena.pl" oraz zostanie przypisany własny id „SMS-001”

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.0" login="user_name" pass="password_string" action=”send”>
<send_sms>
<id_user>SMS-001</id_user>
<msisdn>+48601234567</msisdn>
<body> Przesyłka numer 123456 została nadana. </body>
<sender>SMSarena.pl</sender>
<type_sms>f</type_sms>
<send_after>2012-01-01 01:00:00</send_after>
</send_sms>
</request>
```

3.5.2 Sprawdzanie stanu wysłanego SMSa (status)

akcja *status* służy do sprawdzania statusu nadanego SMS-a (-ów). Za pomocą komendy można uzyskać aktualny status jednego lub kilku SMSów. Za pomocą jednego wywołania komendy można sprawdzić kilka SMS-ów.
Wysłanie akcji *status* bez identyfikatorów spowoduje zwrócenie błędu.

3.5.2.1 Komenda

Format wiadomości:

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.0" login="user_name" pass="password_string" action=”status”>
<status>
<id>12345,123456</id>
</status>
</request>
```

- **<id>** – wewnętrzny identyfikator SMS-a, nadany przy wysyłaniu. Elementów <*id*> można załączyć jeden lub kilka po przecinku

3.5.2.2 Odpowiedź

Format wiadomości

```
<?xml version="1.0" encoding="UTF-8"?>
<response version="1.0" code="code_int" code_body="code_string" action="status" >
<status>
<id>12345</id>
<id_user>Wartość dowolna</id_user>
<msisdn> Numer_MSISDN </msisdn>
<status>Status Wiadomości</status>
<status_code>Kod statusu</status_code>
<send_at>Data Wysłania</send_at>
<delivered_at>Data Dostarczenia</delivered_at>
<updated_at>Data aktualizacji rekordu</updated_at>
</send_sms>
<status>
<id>123456</id>
<id_user>Wartość dowolna</id_user>
<msisdn> Numer_MSISDN </msisdn>
<status>Status Wiadomości</status>
<status_code>Kod statusu</status_code>
<send_at>Data Wysłania</send_at>
<delivered_at>Data Dostarczenia</delivered_at>
<updated_at>Data aktualizacji rekordu</updated_at>
</status>

</response>
```

Parametry:

- **<id>** Wewnętrzny identyfikator SMS-a
- **<id_user>** Własny identyfikator SMS-a
- **<msisdn>** Numer odbiorcy SMS-a
- **<status>** Aktualny status wiadomości

  - *new* – wiadomość oczekuje na wysłanie
  - *queued* – wiadomość w trakcie wysyłania
  - *send* – wiadomość wysłana
  - *delivered* – wiadomość dostarczona
  - *undelivered* – wiadomość nie dostarczona
  - *rejected* – numer odrzucony przez operatora
  - *failed* – wiadomość odrzucana (Patrz status_code)

- **<status_code>** Kod błędu w przypadku statusus failed

  - 500 - Nie poprawny numer nadawcy
  - 501 - Brak treści SMS
  - 502 - Usługa zawieszona przez system
  - 503 - Usługa zawieszona przez uzytkownika
  - 504 - Usługa oczekuje na usunięcie
  - 505 - API MYSQL dla tej usługi została wyłaczona przez Administrację
  - 506 - API MYSQL dla tej uslugi została wyłaczona przez Uzytkownika
  - 507 - Bład INSERT SQL
  - 508 - SMS jest za długi
  - 509 - Brak usługi
  - 510 - Brak wystarczających ilości dostepnych sms-ów oraz kwoty na pobrycie wyslania sms-a
  - 511 - Bład ustalenia kanału wyjściowego. należy PILNIE powiadomić SERWIS
  - 512 - Numer Odbiorcy nie jest obsługiwany.
  - 513 - Numer zagraniczny nie jest obsługiwany
  - 514 - Operator GSMS nie jest obsługiwany
  - 515 - Klucz sesji jest niepoprawny
  - 516 - Brak sms-ów do pobrania lub id sms-a jest nie poprawne

- **<send_at>** - data wysłania format *yyyy-mm-dd hh:mm:ss*
- ***<delivered_at>*** - data dostarczenia format *yyyy-mm-dd hh:mm:ss*
- ***<updated_at>*** - data modyfikacji rekordu format *yyyy-mm-dd hh:mm:ss*

Uwaga: Sekcje <*status*> w odpowiedzi powinny odpowiadać sekcjom <*id*> w wywołaniu komendy. Jeśli sekcji <*status*> jest mniej oznacza to, że albo podano w wywołaniu ten sam identyfikator więcej niż raz lub SMS o podanym identyfikatorze nie istnieje w systemie.

3.5.3 Odebranie SMS a (sms_in)

Akcja sms_in próbuje odczytać SMS-a przychodzącego (SMS-a wysłanego do serwisu z telefonu komórkowego). Jeśli zostały spełnione warunki wskazane w wywołaniu komendy zwracany jest SMS, jeśli nie odpowiedź nie zawiera SMS-a.

System zwraca SMS-y starsze od podanej dacie

3.5.3.1 Komenda

Format wiadomości:

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="1.0" login="user_name" pass="password_string" action="sms_in">
<sms_in>
<received_at>yyyy-mm-dd hh:mm:ss</received_at>
</status>
</sms_in >
```

Elementy obowiązkowe:

- **< *received_at* >** – należy podać datę po której przyszły sms-y

3.5.3.2 Odpowiedź

Format wiadomości:

```
<?xml version="1.0" encoding="UTF-8"?>
    <request version="1.0" login="user_name" pass="password_string" action="sms_in">
      <sms_in>
            <id>Id nadane przez system</id>
             <msisdn>Numer nadawcy</msisdn>
             <body>Treść SMS-a</body>
            <received_at>Data odebrania</received_at>
            <updated_at>Data aktualizacji rekordu</updated_at>
      <sms_in>
      </response>
```

Parametry:

- **<id>** – wewnętrzny identyfikator SMS-a w systemie.
- **<msisdn>** – numer MSISDN telefonu, który przysłał wiadomość do serwisu.
- **<body>** – treść odebranej wiadomości
- **<received_at>** – czas odebrania wiadomości przez serwis w formacie „yyyy-mm-dd hh:mm:ss”.
- ***<updated_at>*** - data modyfikacji rekordu format *yyyy-mm-dd hh:mm:ss*

4 Zastosowania po stronie klienta

Rozdział ten nie wyczerpuje tematu podłączenia do platformy za pomocą interfejsu API XML. Praktycznie każdy nowoczesny język posiada biblioteki ułatwiające połączenie HTTPS i wspierające język XML. Poniżej pokazujemy przykłady wykorzystania niektórych z nich.

4.1 PHP

4.1.1 Wstęp

PHP jest językiem skryptowym najpowszechniej spotykam w aplikacjach webowych. Posiada olbrzymią ilość wbudowanych i opcjonalnych bibliotek.

4.1.2 Biblioteki

W przykładzie wykorzystana została biblioteka „CURLl" oraz „SimpleXML". Biblioteka „SimpleXML" posłużyła tylko do parsowania requestu i nie jest niezbędna.
Więcej informacji można znaleźć w dokumentacji do PHP:
SimpleXML: *http://pl2.php.net/manual/pl/ref.simplexml.php*
CURL: *http://pl2.php.net/manual/pl/ref.curl.php*

4.1.3 Przykład

Poniżej znajduje się prosty przykład wysłania sms-ów:

```
<?

$url =      'https://api.smsarena.pl/xml_post/';
$login =    'login';
$pass =     'pass';
$action =  'send';
$xml_request='<?xml version="1.0" encoding="UTF-8"?>
          <request version="1.0" login="'.$login.'" pass="'.$pass.'" action="'.$action.'">
          <send_sms>
               <msisdn>+48660332967</msisdn>
               <body>Tresc SMS-a</body>
          </send_sms>
          </request>';
$ch = curl_init();
curl_setopt( $ch,  CURLOPT_URL, $url);
curl_setopt( $ch, CURLOPT_HTTPAUTH, CURLAUTH_ANY);
curl_setopt( $ch, CURLOPT_TIMEOUT, 30);
curl_setopt( $ch, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt( $ch, CURLOPT_HEADER, false);
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt( $ch, CURLOPT_POST, true);
curl_setopt( $ch, CURLOPT_POSTFIELDS, "xml=".urlencode( $xml_request ));
$ret=curl_exec($ch);

if (curl_errno($ch)!=0) {
   echo "Error: ".curl_error($ch);
} else {
   $info = curl_getinfo($ch);
   echo "HTTP Code: ".$info['http_code']."<br>"; // 200 - HTTPS ok

   $xml_response=simplexml_load_string($ret);
echo 'CODE BODY'.$xml_response['code_body']."<br>";
echo 'ID SMS-a: '.$xml_response->send_sms->id."<br>";
echo 'Status: '.$xml_response->send_sms->status."<br>";
echo 'Status Code: '.$xml_response->send_sms->status_code."<br>";
}
curl_close($ch);
?>
```

4.2 Ruby

4.2.1 Wstęp

Język Ruby uzyskał swoją popularność w przeciągu ostatnich 2-3 lat. Swoją sławę zawdzięcza głównie środowisku Ruby on Rails, które pozwala na bardzo szybkie tworzenie aplikacji webowych.

4.2.2 Biblioteki

W przykładzie wykorzystana zostały biblioteki „Net::HTTPS" oraz „XMLSimple". Biblioteka „XMLSimple" posłużyła tylko do parsowania requestu i nie jest niezbędna.

Więcej informacji można znaleźć w dokumentacji Ruby: XMLSimple: *http://xml-simple.rubyforge.org/*
Net::HTTP: *http://www.ruby-doc.org/stdlib/libdoc/net/http/rdoc/index.html*

4.2.3 Przykład

Poniżej znajduje się prosty przykład sprawdzenia, czy platforma przechowuje nieodczytane SMS-y.

```
require 'net/https'
require 'xml_simple'
# zmienne połączenia
url =      'https://api.smsarena.pl/ '
login =   'login'
pass =    'pass'
action =  'send'
port ='80 '
xml_request='<?xml version="1.0" encoding="UTF-8"?>
          <request version="1.0" login="+login+ " pass="+pass+" action="+action+">
          <send_sms>
               <msisdn>+48660332967</msisdn>
               <body>Tresc SMS-a</body>
          </send_sms>
          </request>'
# inicjacja obiektu odpowiedzialnego za połączenie
http = Net::HTTP.new(host, port)
http.use_ssl=true
begin
# wykonanie komendy
ret = http.post("xml_post/",'xml='+xml_request)
if ret.code.to_i==200
xml_response = XmlSimple.xml_in(ret.body)
puts "Status: "+xml_response["code_status"]
puts "SMS ID: "+xml_response["send_sms"][0]"id"][0]
puts "Status:".xml_response["send_sms"][0]["status"][0]
else
puts "Error: #{res.code} - #{res.message} "
end
rescue Error=>e
puts "Connection error:" + e.to_s
end
```