SMESKOM
KOMUNIKACJA SMS DLA BIZNESU

DOKUMENTACJA PROTOKOŁU SMESX

Platforma SMeSKom - instrukcja korzystania z interfejsu HTTPS
Protokół w wersji 2.2

Autor smeskom@smeskom.pl

Data 16.06.2009
Wersja 2.2 (rev. 1)

SOFTIKA
ROZWIĄZANIA NA MIARĘ POTRZEB

Spis treści

1 Zawartość dokumentu

Niniejszy dokument zawiera informacje na temat sposobu komunikowania się z platformą SMeSKom korzystając z „interfejsu HTTPS”.

Rozdział 2 zawiera informacje o interfejsie, sposobie i parametrach połączenia. Proszę zwrócić uwagę na punkt 2.4, w którym wyjaśnione są ograniczenia protokołu. 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 z platformą.

W rozdziale 4 znajdziecie Państwo przykłady wykorzystania typowych narzędzi do budowania aplikacji klienckich.

2 Ogólne informacje o interfejsie

2.1 Zastosowanie

Opisany w dokumencie interfejs służy do wysyłania i odbierania SMS-ów za pomocą platformy SMeSKom. Opisany protokół ma zastosowanie jedynie dla platformy SmeSKom (http://www.smeskom.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 2.2 protokołu. Wersja ta jest jedynie rozszerzeniem wersji 2.1 o możliwość zmiany pola nadawcy SMS-a.

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 „/smesx”)

znaleźć można w panelu (menu „Interfejsy” dla wybranego serwisu)

Pełny url może wyglądać jak poniżej:

https://smesx1.smeskom.pl:2200/smesx

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

2.4 Ograniczenia częstości wydawania komend

Komunikacja w ramach protokołu odbywa się na zasadzie wydawania poleceń do serwera i tylko w tym kierunku. Aplikacja kliencka oczekując na SMS-a zwrotnego lub sprawdzając status wysyłania SMS-a musi zastosować się do pewnych reguł, wymienionych poniżej. Przekraczając wskazane wartości, aplikacja kliencka narażona jest na czasowe i całkowite zablokowanie interfejsu lub odrzucenie komendy.

- Operacja odbioru SMS-a, przy braku potwierdzenia, ze w systemie są nieprzeczytane SMS-y, nie może odbywać się częściej niż raz na 3 sekundy. Przykładowe zachowanie aplikacji klienckiej może wyglądać następująco:
  - aplikacja zestawia połączenie (pierwszym zapytaniem HTTPS)
  - w pętli co minutę sprawdza, czy w systemie są SMS-y przychodzące (za pomocą komendy odbierającej SMS)
  - jeśli wynik komendy zawiera wskazanie, że w systemie są jeszcze nieprzeczytane SMS-y przychodzące, czyta SMS-y dopóki w systemie nie będzie więcej nieprzeczytanych wiadomości
  - jeśli wynik nie zawiera informacji o nieprzeczytanych SMS-ach, usypia na 3 minuty
- Operacja sprawdzania statusu wysyłania SMS(-ów), nie może odbywać się częściej niż raz na 3 sekundy. Większą ilość SMS-ów sprawdza się przy pomocy jednego wysłania komendy.
- Średnia ilość zapytań do serwera (wszystkie komendy wysłane za pomocą interfejsu) nie może przekraczać 30 operacji na minutę.
- Zalecane wartości
  - Opóźnienie przy kolejnych sprawdzeniach, czy w systemie oczekują SMS-y: powyżej 30 sekund. O
  - Sprawdzenie statusu wysyłanej wiadomości: powyżej 10 sekund. Optymalna metoda opisana w dokumencie przewiduje , że opóźnienie zależy od czasu ostatniej wysyłki SMS.
  - Opóźnienie wysyłania wiadomości: brak

3 Specyfikacja protokołu SmesX

3.1 Komunikacja z serwerem

Interfejs działa w architekturze klient-serwer. Do aplikacji klienckiej należy obowiązek zainicjowania i utrzymania połączenia. Aplikacja kliencka powinna zostać tak napisana, aby uwzględniała opisane w punkcie 2.4 zasady. 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 zawierać dodatkowych, zbędnych i niezrozumiałych dla serwera tagów (elementów i atrybutów 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="2.2" user="htguser123"
password="8TyMI8LN">
```

Wartości parametrów autoryzacyjnych dostępne są w panelu

administracyjnym po wybraniu serwisu, a potem wejście do menu „Interfejsy”.

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  protocol=”SmesX” version="2.2"
user="user_name" password="password_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 protocol=”SmesX” version="2.2" >
  <execution_status>success</execution_status>
 ...
</response>
```

Wewnątrz elementu <*response*> znajdują się wyniki wykonania wywoływanej komendy.

Atrybut *execution_status* zawiera tekst „*success*” - w przypadku poprawnego wykonania komendy lub „*failed*” w przypadku błędu.

3.4.3 Odpowiedź z informacją o błędzie

W przypadku źle sformatowanego polecenia lub wystąpieniu innych problemów odpowiedź z serwera ma stałą postać:

```
<?xml version="1.0" encoding="UTF-8"?>
<response  protocol=”SmesX” version="2.2" >
  <execution_status>failed</execution_status>
  <fail_code>numer</fail_code>
  <fail_description>słowny opis błędu</fail_description>
</response>
```

Uwaga: w stosunku do wersji 1.0 protokołu nastąpiła zmiana nazw elementów z *<failure_code>* na *<fail_code>* oraz z *<failure_description>* na *<fail_description>*

3.5 Komendy

3.5.1 Wysłanie SMS -a (send_sms)

Komenda *send_sms* 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 komendy *sms_status.* Funkcja zwraca wewnętrzny dla platformy identyfikator SMS-a. Identyfikator ten używany jest przy wykonywaniu komendy *sms_status*.

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

3.5.1.1 Komenda

Format wiadomości:

```
<?xml version="1.0" encoding="UTF-8"?>
<request  protocol=”SmesX” version="2.2"
user="user_name" password="password_string">
  <send_sms>
    <msisdn>Numer_MSISDN</msisdn>
    <body>Tutaj treść SMS-a</body>
    <expire_at>yyyy-mm-dd hh:mi:ss</expire_at>
    <sender>Nadawca</sender>
    <sms_type>Kod typu smsa</sms_type>
    <send_after>yyyy-mm-dd hh:mi:ss</send_after>
  </send_sms>
</request>
```

Elementy obowiązkowe:

- **<msisdn>** – numer MSISDN. Platforma SmeSKom powinna zaakceptować większość możliwych formatów o ile numer jest napisany jednoznacznie. Przykładowe formaty, które zostaną zaakceptowane:
  - 601 234 567
  - +48 601-234-567
  - (0) 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). Jeśli

długość tekstu przekracza te wartości, polskie znaki zostaną automatycznie zamienione na ich odpowiedniki bez ogonków. Więcej na ten temat dowiedzieć się można na naszych stronach („Reguły wysyłania SMS-ów”)

Elementy opcjonalne:

- **<expire_at>** – data wygaśnięcia ważności. Platforma nie pozwoli na wysłanie SMS po tym terminie. Jeśli parametr nie zostanie podany czas wygaśnięcia zostanie ustawiony na 3 dni do przodu.
- **<send_after>** – data planowanej wysyłki SMS. Platforma wyśle SMS-a 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 z aktualnym czasie.
- **<sender>** – Ustawia inne niż domyślne pole nadawcy dla SMS-ów w usłudze typu "Powiadomienia". 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.

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

3.5.1.2 Odpowiedź

Format wiadomości

```
<?xml version="1.0" encoding="UTF-8"?>
<response  protocol=”SmesX” version="2.2">
  <execution_status>success</execution_status>
  <send_sms>
    <id>12345</id>
    <expire_at>yyyy-mm-dd hh:mi:ss</expire_at>
  </send_sms>
</response>
```

3.5.1.3 Przykłady

Wysłanie SMS o treści „Przesyłka numer 123456 została nadana.” na numer 601-234-567

```
<?xml version="1.0" encoding="UTF-8"?>
<request  protocol=”SmesX” version="2.2"
user="user_name" password="password_string">
  <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 601-234-567 jako wiadomość flash o wyznaczonej godzinie. SMS zostanie wysłany z polem nadawcy "Smeskom"

```
<?xml version="1.0" encoding="UTF-8"?>
<request  protocol=”SmesX” version="2.2"
user="user_name" password="password_string">
  <send_sms>
    <msisdn>+48601234567</msisdn>
    <sender>Smeskom</sender>
    <body>Przesyłka numer 123456 została nadana.</body>
        <sms_type>f</sms_type>
        <send_after>2008-08-20 12:30:00</send_after>
  </send_sms>
</request>
```

3.5.2 Sprawdzanie stanu wysłanego SMS-a (sms_status)

Komanda <*sms_status*> służy do sprawdzania statusu nadanego SMS-a (-ów). Za pomocą komendy można uzyskać aktualny status jednego lub kilku SMS-ów. Za pomocą jednego wywołania komendy można sprawdzić do 10-u SMS-ów jednocześnie.

W wersji 2.0 protokołu dodano możliwość sprawdzania nieodczytanych statusów. Wysłanie komendy <*sms_status*> bez identyfikatorów spowoduje odczytanie zmian w statusach SMS-ów, ale tylko tych, których przy pomocy tej komendy jeszcze nie odczytaliśmy. Zasięg wyszukiwania zmian w statusach SMS-ów to ostatnie 24godziny.

Wysyłając pustą komendę <*sms_status*> otrzymamy listę SMS-ów (maksymalnie 10), których statusy zmieniły się od czasu ostatniego wysłania tej komendy jako pustej. Tak więc wysyłając cyklicznie tą komendę mamy pewność, że wszystkie zmiany w statusach zostaną zauważone i przeczytane przez nas.

3.5.2.1 Komenda

Format wiadomości:

```
<?xml version="1.0" encoding="UTF-8"?>
<request  protocol=”SmesX” version="2.2"
user="user_name" password="password_string">
  <sms_status>
    <id>12345</id>
    <id>12346</id>
     ...
  </sms_status>
</request>
```

Elementy obowiązkowe: brak

Elementy opcjonalne:

- **<id>** – wewnętrzny identyfikator SMS-a, nadany przy wysyłaniu. Elementów <*id*> można załączyć jeden lub kilka. Maksymalnie można zawrzeć 10 elementów <*id*> w jednej wiadomości. Jeśli nie podamy żadnego elementu <*id*> zostaną zwrócone nieodczytane statusy (patrz

punkt 3.5.2)

3.5.2.2 Odpowiedź

Format wiadomości:

```
<?xml version="1.0" encoding="UTF-8"?>
<response  protocol="SmesX" version="2.2">
  <execution_status>success</execution_status>
  <sms_status>
    <sms>
        <id>12345</id>
        <status>status</status>
        <sent_at>yyyy-mm-dd hh:mi:ss</sent_at>
        <delivered_at>yyyy-mm-dd hh:mi:ss</delivered_at>
        <fail_code>number</fail_code>
    </sms>
    <sms>
        <id>12346</id>
        <status>status</status>
        <sent_at>yyyy-mm-dd hh:mi:ss</sent_at>
        <delivered_at>yyyy-mm-dd hh:mi:ss</delivered_at>
        <fail_code>number</fail_code>
    </sms>
     ...
  </sms_status>
</response>
```

Parametry:

- **<id>** – wewnętrzny identyfikator SMS-a wychodzącego
- **<status>** – status wiadomości. Wartość elementu może być jedna z:
  - new – nowy SMS, status new zostaje ustawiony przy wysłaniu komendy <*send_sms*> (nadaniu wiadomości SMS)
  - processed – system rozpoczął przetwarzanie wiadomości, wiadomość została przeznaczona do wysyłki
  - sent – wiadomość została poprawnie wysłana przez system
  - delivered – wiadomość została odebrana przez adresata (przyszedł tzw. „raport doręczenia")
  - failed – nie udało się wysłać wiadomości. Informacja o błędzie została zawarta w elemencie <*fail_code*>
- **<sent_at>** – czas wysłania wiadomości w formacie „yyyy-mm-dd hh:mi:ss" (np.: „2007-11-10 14:20:00"). Element ten nie zawiera żadnych znaków, jeśli wiadomość nie została jeszcze wysłana.
- **<delivered_at>** – czas odebrania wiadomości przez adresata w formacie „yyyy-mm-dd hh:mi:ss" (np.: „2007-11-10 14:20:00"). Element ten nie zawiera żadnych znaków, jeśli wiadomość nie została jeszcze odebrana (SMS nie dotarł lub nie mógł być odebrany przez odbiorcę).
- **<fail_code>** – kod błędu, wartość całkowita, numeryczna. Wartość *0* jest wpisana, jeśli SMS został wysłany poprawnie lub, gdy jeszcze nie

doszło do wysłania.

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

3.5.2.3 Przykłady

Zapytanie o SMS-y, któych status zmienił się od czasu ostatniego takiego samego zapytania:

```
<?xml version="1.0" encoding="UTF-8"?>
<request  protocol=”SmesX” version="2.2"
user="htguser12" password="asb7fgioe9">
  <sms_status>
  </sms_status>
</request>
```

UWAGA: Jeśli zwrócone zostanie 10 SMS-ów należy zapytać się jeszcze raz o następne SMSy (o ile są). Jeśli zwrócone zostanie 0 lub od 1 do 9 elementów nie należy się pytać dalej.

Zapytanie o SMS-a o identyfikatorze 12345:

```
<?xml version="1.0" encoding="UTF-8"?>
<request  protocol=”SmesX” version="2.2"
user="htguser12" password="asb7fgioe9">
  <sms_status>
    <id>12345</id>
  </sms_status>
</request>
```

Przykładowa odpowiedź dla SMS-a, który został wysłany, ale jeszcze nie dotarł do odbiorcy:

```
<?xml version="1.0" encoding="UTF-8"?>
<response  protocol=”SmesX” version="2.2">
  <execution_status>success</execution_status>
  <sms_status>
    <sms>
        <id>12345</id>
        <status>sent</status>
        <sent_at>2007-11-08 11:23:43</sent_at>
        <delivered_at>-</delivered_at>
        <fail_code>0</fail_code>
    </sms>
  </sms_status>
</response>
```

3.5.2.4 Optymalne sprawdzanie zmian statusów SMS-ów

Zalecana metoda odczytywania/synchronizacji statusów z własnym systemem to wykorzystanie komendy <sms_status> bez podawania identyfikatorów. Metoda ta polega na sprawdzeniu SMS-ów, których status zmienił się od czasu ostatniego odpytywania o zmiany w statusach. Tego typu odpytanie powinno

wykonywać się z częstotliwością zależną od czasu ostatniej zmiany statusu. Jeśli SMS nie został doręczony do telefonu od razu, SMSC operatora będzie próbować dostarczyć go jeszcze kilka razy, ale za każdym razem z coraz większą przerwą. W związku z tym najbardziej optymalne będzie, gdy zaraz po wysłaniu SMS-a odpytujemy się częściej, ale gdy ostatnio wysłany SMS nie został dostarczony od razu odpytujemy się rzadziej, aż do maksymalnego czasu odpytywania (np. 15 minut).

Oczekiwanie na następne odczytanie statusów powinno zwiększać się po każdorazowym odpytaniu (aż do wartości maksymalnej), a wysyłka nowego SMS-a powinna resetować ten czas do wartości minimalnej.

3.5.3 Odwołanie zaplanowanego SMS-a (revoke_sms)

Do odwołania SMS-a, którego zaplanowaliśmy na czas przyszły użyć należy komendy *<revoke_sms>*. Składnia komendy jest analogiczna do komendy sprawdzającej status. Możliwe jest jednoczesne odwołanie do 10 SMS-ów, a zwrot z komendy pokazuje, czy udało się rozpocząć procedurę odwołania, czy nie.

3.5.3.1 Komenda

Format wiadomości:

```
<?xml version="1.0" encoding="UTF-8"?>
<request  protocol=”SmesX” version="2.2"
user="user_name" password="password_string">
  <revoke_sms>
    <id>12345</id>
    <id>12346</id>
     ...
  </revoke_sms>
</request>
```

Elementy obowiązkowe:

- **<id>** – wewnętrzny identyfikator SMS-a, nadany przy wysyłaniu. Elementów *<id>* można załączyć jeden lub kilka. Maksymalnie można zawrzeć 10 elementów *<id>* w jednej wiadomości.

3.5.3.2 Odpowiedź

Format wiadomości:

```
<?xml version="1.0" encoding="UTF-8"?>
<response  protocol=”SmesX” version="2.2">
  <execution_status>success</execution_status>
  <revoke_sms>
    <sms>
        <id>12345</id>
        <revoke_status>status</revoke_status>
    </sms>
    <sms>
        <id>12346</id>
        <revoke_status>status</revoke_status>
    </sms>
```

```
    ...
  </revoke_sms>
</response>
```

Parametry:

- **<id>** – wewnętrzny identyfikator SMS-a wychodzącego
- **<revoke_status>** – status polecenia odwołania SMS-a. Wartość elementu może być jedna z:
  - revoke requested – odwołanie SMS-a zostanie przeprowadzone w ciągu najbliższych kilku sekund. Do sprawdzenia czy odwołanie powiodło się, należy użyć komendy „sms_status”. Zmiana statusu na „failed” i ustawienie „fail_code” na 410 oznacza poprawne zakończenie procedury odwołania. Inny fail_code może oznaczać niemożność odwołania. Patrz kody błędów.
  - revoke impossible– SMS został już przekazany dalej do wysyłki – odwołanie nie jest możliwe.

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

3.5.3.3 Przykłady

Odwołanie SMS-a o id: 12345:

```
<?xml version="1.0" encoding="UTF-8"?>
<request  protocol=”SmesX” version="2.2"
user="htguser12" password="asb7fgioe9">
  <revoke_sms>
    <id>12345</id>
  </revoke_sms>
</request>
```

Przykładowa odpowiedź dla SMS-a, którego poprawnie przeznaczone do odwołania:

```
<?xml version="1.0" encoding="UTF-8"?>
<response  protocol=”SmesX” version="2.2">
  <execution_status>success</execution_status>
  <revoke_sms>
    <sms>
        <id>12345</id>
        <revoke_status>revoke requested</revoke_status>
    </sms>
  </revoke_sms>
</response>
```

3.5.4 Odebranie SMS -a (receive_sms)

Komenda 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.

Komenda pracuje w dwóch trybach (tryb ustawia się za pomocą elementu *<type>*).

Tryb 1. (*<type>unread</type>*) Odbieranie wiadomości, która nie została jeszcze oznaczona jako przeczytana.

Każda nowa wiadomość, która przyjdzie do serwisu jest oznaczona jako „nieprzeczytana”. Można ją zaznaczyć jako przeczytaną za pomocą:

- komendy *<mark_read>.* Patrz punkt 3.5.5, aby dowiedzieć się więcej.
- ustawiając element *<mark>* w poleceniu *<receive_sms>* wartością „*true”* (jest to również domyślna wartość dla tego elementu)

W tym trybie system próbuje dostarczyć SMS, który nie został jeszcze oznaczony jako „przeczytany”

Tryb 2. (*<type>time</type>*) Odbieranie dowolnych wiadomości w zadanym czasie.

W tym trybie system nie zwraca uwagi na oznaczenie SMS-a (czy został oznaczony jako przeczytany, czy nie). Zwraca najstarszy SMS w zadanym przez parametry czasie.

Uwaga: W obu trybach bardzo istotny jest element ***<after_id>*** w wywołaniu polecenia. Wartość elementu *<after_id>* może zawierać identyfikator SMS-a, który został odebrany przez system tuż przed SMS-em, który chcemy odczytać. Np.: jeśli próbujemy odczytać SMSy odebrane przez system pomiędzy godziną 13-ą, a 15-ą danego dnia i przyszło ich kilka w tym okresie należy:

- przy pierwszym wywołaniu *<receive_sms>* pominąć parametr *<after_id>*
- wywoływać *<receive_sms>* dotąd, aż nie zwróci żadnego SMS-a. Za każdym razem parametr *<after_id>* ustawiamy wartością identyfikatora SMS-a odebranego w poprzednim wykonaniu polecenia *<receive_sms>*

3.5.4.1 Komenda

Format wiadomości:

```
<?xml version="1.0" encoding="UTF-8"?>
<request  protocol=”SmesX” version="2.2"
user="user_name" password="password_string">
  <receive_sms>
    <type>type</type>
    <start_time>yyyy-mm-dd hh:mi:ss</start_time>
    <stop_time>yyyy-mm-dd hh:mi:ss</stop_time>
    <after_id>number</after_id>
    <mark>boolean</mark>
  </receive_sms>
</request>
```

Elementy obowiązkowe:

- **<type>** – wybór trybu odczytu SMS-ów przychodzących. Ten parametr może mieć dwie wartości: „*time*”, „*unread*”. Znaczenie tych wartości i

działania poszczególnych trybów zostało opisane w punkcie 3.5.4.

- **<start_time> -** parametr obowiązkowy jedynie w trybie **„time”**. Wartość przekazana w tym parametrze oznacza początek odpytywanego okresu. Format parametru „yyyy-mm-dd hh:mi:ss”.

Elementy opcjonalne:

- **<stop_time>** – parametr mający sens jedynie dla trybu **„time”**. Oznacza koniec odpytywanego okresu czasu. Format parametru „yyyy-mm-dd hh:mi:ss”. Jeśli nie podamy tego parametru – wartość zostanie ustawiona na aktualny czas.
- **<mark>** – domyślna wartość: „true”. Jeśli wartość ta jest ustawiona na „true”, odczytany za pomocą wywołania SMS zostanie oznaczony jako „przeczytany”. Jeśli wartość ustawimy na „false”, SMS pozostanie „nieprzeczytany”.
- **<after_id>** – identyfikator SMS, który odczytany został w poprzednim wykonaniu. Parametr ten ma znaczenie jeśli odbieramy wiele SMS-ów w zadanym czasu. Jeśli nie podamy tego parametru, zostanie zwrócony najstarszy SMS w zadanym czasie.

3.5.4.2 Odpowiedź

Format wiadomości:

```
<?xml version="1.0" encoding="UTF-8"?>
<response  protocol=”SmesX” version="2.2">
  <execution_status>success</execution_status>
  <receive_sms>
    <contain_sms>boolean</contain_sms>
    <has_more>boolean</has_more>
    <sms>
        <id>12345</id>
        <msisdn>MSISDN number</msisdn>
        <body>Treść wiadomości</body>
        <inserted_at>yyyy-mm-dd hh:mi:ss</inserted_at>
        <received_at>yyyy-mm-dd hh:mi:ss</received_at>
        <marked_at>yyyy-mm-dd hh:mi:ss</marked_at>
    </sms>
  </receive_sms>
</response>
```

Parametry:

- **<contain_sms>** – jedna z wartości: „true”, „false”. „true” oznacza, że udało się odczytać SMS-a, a wiadomość zawiera sekcję *<sms>* ze szczegółami odczytanego SMS-a. Wartość „false” oznacza, że dla zadanych parametrów nie ma odebranych SMS-ów.
- **<has_more>** – jedna z wartości: „true”, „false”. „true” oznacza, że dla zadanych parametrów są jeszcze SMS-y, które można odczytać. Wartość „false” oznacza, że dla zadanych parametrów nie ma więcej odebranych SMS-ów w systemie.
- **<id>** – wewnętrzny identyfikator SMS-a w systemie.
- **<msisdn>** – numer MSISDN telefonu, który przysłał wiadomość do

serwisu.

- **<body>** – treść odebranej wiadomości
- **<inserted_at>** – czas nadejścia SMS-a do systemu. Czas w formacie „yyyy-mm-dd hh:mi:ss” (np.: „2007-11-10 14:20:00”).
- **<received_at>** – czas odebrania wiadomości przez modem. Zazwyczaj ten czas jest od kilku do kilkudziesięciu sekund starszy od elementu <*inserted_at*>. Element w formacie „yyyy-mm-dd hh:mi:ss”.
- **<marked_at>** – czas oznaczenia wiadomości jako „przeczytanej”. Jeśli pole to jest puste, oznacza to, że wiadomość nie została jeszcze oznaczona jako „przeczytana”.

3.5.4.3 Przykłady

Ciągłe odczytywanie wiadomości przychodzących może odbywać się za pomocą cyklicznie wykonywanego polecenia:

```
<?xml version="1.0" encoding="UTF-8"?>
<request  protocol=”SmesX” version="2.2"
user="user_name" password="password_string">
  <receive_sms>
    <type>unread</type>
  </receive_sms>
</request>
```

Jednoczesne ustawienie parametru <*type*> na wartość „unread” oraz pozostawienie bez zmian parametru <*mark*> (domyślna wartość „true”) powodują, że jeśli w systemie oczekuje na odczytanie SMS przychodzący, zostanie on zwrócony i jednocześnie zaznaczony jako „przeczytany”. Dzięki temu ponowne wykonanie tej samej komendy spowoduje zwrócenie następnego nieprzeczytanego SMS-a.

3.5.5 Zaznaczanie odebranych SMS-ów jako przeczytane (mark_read)

<mark_read> jest pomocniczym poleceniem w procesie odbierania SMS-ów. Komenda ta pozwala na zaznaczenie wiadomości odebranej jako przeczytana przez użytkownika.

Dwuetapowe odczytywanie SMS-ów:

- etap 1. odczytanie SMS-a z parametrem <mark> równym „false”
- etap 2. zaznaczenie SMS-a jako „przeczytany”

pozwala na zwiększenie bezpieczeństwa odczytu SMS-ów.

Takie odczytywanie SMS-ów nie pozwoli na hipotetyczną sytuację, gdy do aplikacji klienckiej nie dojdzie wiadomość zwrotna (np. z powodu problemów z siecią)

3.5.5.1 Komenda

Format wiadomości:

```
<?xml version="1.0" encoding="UTF-8"?>
<request  protocol=”SmesX” version="2.2"
user="user_name" password="password_string">
  <mark_read>
    <id>12345</id>
    <id>12346</id>
     ...
  </mark_read>
</request>
```

Elementy obowiązkowe:

- **<id>** – wewnętrzny identyfikator SMS-a, nadany w momencie odebrania SMS-a i przekazywany przez polecenie *<receive_sms>*. Elementów *<id>* można załączyć jeden lub kilka. Maksymalnie można zawrzeć 10 elementów *<id>* w jednej wiadomości.

Elementy opcjonalne: brak

3.5.5.2 Odpowiedź

Format wiadomości:

```
<?xml version="1.0" encoding="UTF-8"?>
<response  protocol=”SmesX” version="2.2">
  <execution_status>success</execution_status>
  <mark_read>
    <sms>
        <id>12345</id>
        <marked_at>yyyy-mm-dd hh:mi:ss</marked_at>
    </sms>
    <sms>
        <id>12346</id>
        <marked_at>yyyy-mm-dd hh:mi:ss</marked_at>
    </sms>
     ...
  </mark_read>
</response>
```

Parametry:

- **<id>** – wewnętrzny identyfikator SMS-a przychodzącego
- **<marked_at>** – data oznaczenia SMS-a. Jeśli SMS został oznaczony wcześniej, SMS nie zostanie ponownie zaznaczony. Data w tym polu jest zawsze datą pierwszego oznaczenia.

3.5.5.3 Przykłady

Polecenie oznaczenia wiadomośći:

```
<?xml version="1.0" encoding="UTF-8"?>
<request  protocol=”SmesX” version="2.2"
user="htguser32” password="cjs73md2g">
  <mark_read>
    <id>12346</id>
  </mark_read>
</request>
```

Zwrot z wykonania polecenia:

```
<?xml version="1.0" encoding="UTF-8"?>
<response  protocol=”SmesX” version="2.2">
  <execution_status>success</execution_status>
  <mark_read>
    <sms>
        <id>12346</id>
        <marked_at>2007-11-01 09:00:02</marked_at>
    </sms>
  </mark_read>
</response>
```

4 Zastosowania po stronie klienta

Rozdział ten nie wyczerpuje tematu podłączenia do platformy za pomocą interfejsu SmesX. 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 sprawdzenia, czy platforma przechowuje nieodczytane SMS-y.

```
<?php

// zmienne połączenia
$host= "www.smeskom.pl";
```

```
$port= 2001;
$user="htguser124";
$pass= "8TyMI8LN";

// poprawnie sformatowana komenda
$xml_request='<?xml version=”1.0” encoding=”UTF-8” ?>
              <request version="2.2" user="'.$user.'" password="'.$pass.'">
                <receive_sms>
                   <type>unread</type>
                   <mark>true</mark>
                </receive_sms>
              </request>';

// inicjacja obiektu odpowiedzialnego za połączenie
$ch = curl_init();
$url = "https://".$host.":".$port."/smesx";
//stałe opcje
curl_setopt($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);

// zmienna opcja
curl_setopt($ch, CURLOPT_POSTFIELDS, "xml=".urlencode($xml_request));

// wykonanie komendy
$ret=curl_exec($ch);

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

  $xml_response=simplexml_load_string($ret);
  echo "Status: ".$xml_response->execution_status."\n";
  echo "SMS Received: ".$xml_response->receive_sms->contain_sms."\n";
  echo "More SMSes waiting: ".$xml_response->receive_sms->has_more."\n";
  echo "XML String response: \n".$ret."\n";
}
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
port=2001
host="www.smeskom.pl"
user="htguser123"
pass="8TyMI8LN"

# poprawnie sformatowana komenda
xml_request='<?xml version="1.0" encoding="UTF-8"?>
  <request version="2.2" user="'+user+'" password="'+pass+'">
    <receive_sms>
      <type>unread</type>
      <mark>true</mark>
    </receive_sms>
  </request>'

# inicjacja obiektu odpowiedzialnego za połączenie
http = Net::HTTP.new(host, port)
http.use_ssl=true

begin
  # wykonanie komendy
  ret = http.post("/smesx",'xml='+xml_request)
  if ret.code.to_i==200
    xml_response = XmlSimple.xml_in(ret.body)
    puts "Status: "+xml_response["execution_status"]
    puts "SMS Received: "+xml_response["receive_sms"][0]"contain_sms"][0]
    puts "More SMSes waits:".xml_response["receive_sms"][0]["has_more"][0]
    puts "XML String response: \n"+ret.body
  else
    puts "Error: #{res.code} - #{res.message}  "
  end
rescue Error=>e
  puts "Connection error:" + e.to_s
end
```

Załącznik 1. Specyfikacja kodów błędu

Kody błędów zostały przeniesione do oddzielnego pliku dokumentacji.