OPIS TECHNICZNY

SYSTEM HOSTED SMS

Wersja 1.6.2
Warszawa, lipiec 2015

SPIS TREŚCI

1. WPROWADZENIE

System Hosted SMS (w skrócie HSMS) przeznaczony jest do przesyłania hurtowych ilości wiadomości SMS. System Hosted SMS zostanie udostępniony Klientowi jako usługa. DCS odpowiedzialny jest za opracowanie oraz jego utrzymanie w infrastrukturze DCS. Rolą DCS jest także zapewnienie odpowiednich łączy do sieci GSM.

2. PODSTAWOWE PARAMETRY SYSTEMU HOSTED SMS

DOSTĘPNOŚĆ

DCS gwarantuję, że dostępność Systemu HSMS nie będzie mniejsza niż 99,5% w skali miesiąca kalendarzowego. Planowane i uzgodnione z Klientem przerwy techniczne nie są liczone, jako niedostępność systemu.

DEFINICJA ZNAKÓW I DŁUGOŚCI WIADOMOŚCI SMS

System HSMS może wysyłać wiadomości SMS o maksymalnie długości 4000 znaków. Odpowiada to 27 SMS-om w przypadku stosowania tylko znaków z alfabetu podstawowego GSM7[1] oraz 60 znaków w przypadku stosowania pełnego alfabetu UNICODE.

- 612 znaków jeśli treść wiadomości zawiera tylko znaki z zakresu GSM7

Liczba znaków w wiadomości SMS	Liczba wiadomości SMS jakie otrzyma odbiorca	Liczba SMSów wykazanych w billingu dla Klient
1-160	1	1
161-306	1	2
307-459	1	3
460-612	1	4
Pow. 612	1[2]	(Liczba znaków/153) - zaokrąglone w górę

- 272 znaki jeśli treść wiadomości zawiera znaki wymagające kodowania UNICODE

Liczba znaków w wiadomości SMS	Liczba wiadomości SMS jakie otrzyma odbiorca	Liczba SMSów wykazanych w billingu dla Klient
1-70	1	1
71-134	1	2
135-201	1	3

[1] Alfabet podstawowy GSM7 oznacza tabelę „Basic Character Set” ze specyfikacji GSM 03.38

[2] UWAGA – Nie wszystkie modele telefonów posiadają możliwości składania wiadomości składającej się z większej ilości segmentów. Nie zaleca się stosowania wiadomości dłuższych niż 10 segmentów.

202-268	1	4
Pow. 269	1[3]	(Liczba znaków / 67) - zaokrąglone w górę

3. DOSTĘPNE INTERFEJSY

INTERFEJS WEBSERWIS2SMS

Interfejs Webserwis2SMS przeznaczony jest do zautomatyzowane wysyłek SMSów realizowanych bezpośrednio przez systemy Klienta, jak również umożliwia odbieranie raportów dostarczenia wiadomości SMS.[4]

Interfejs Webserwis2SMS będzie dostępny pod adresami:

- https://api.hostedsms.pl/WS/SmsSender.asmx - Podstawowy Ośrodek przetwarzania DCS

METODY WEB SERVICE

SENDSMS - wysyła pojedynczą wiadomość. Metoda zwraca true jeżeli system DCS przyjął wiadomość do wysłania.

SENDSMSES – wysyła wiadomość do grupy telefonów. Metoda zwraca true jeżeli system DCS przyjął wszystkie wiadomości do wysłania. Operacja jest transakcyjna – albo wszystkie wiadomości zostaną przyjęte do wysłania, albo żadna.

GETUNREADDELIVERYREPORTS – zwraca wszystkie nieprzeczytane raporty klienta, jednocześnie zaznaczając je jako przeczytane. Maksymalnie 100 raportów jest zwracanych na raz. Pusta tablica informuje o tym, że nie ma możliwych raportów do pobrania.

GETDELIVERYREPORTS – zwraca grupę raportów dostarczenia wiadomości, dla podanych identyfikatorów wiadomości. Zwracana jest tablica typu DeliveryReport.

GETUNREADINPUTSMSES – zwraca wszystkie nieprzeczytane odebrane smsy, jednocześnie zaznaczając je jako przeczytane. Maksymalnie 100 sms-ów jest zwracanych na raz. Pusta tablica informuje o tym, że nie ma możliwych sms-ów odebranych do pobrania.

GETINPUTSMSES – zwraca grupę odebranych sms-ów, dla podanych kryteriów wyszukiwania. Zwracana jest tablica typu InputSms.

GETVALIDSENDERS - zwraca grupę dostępnych nadpisów..

[3] UWAGA – Nie wszystkie modele telefonów posiadają możliwości składania wiadomości składającej się z większej ilości segmentów. Nie zaleca się stosowania wiadomości dłuższych niż 10 segmentów.

[4] Istnieje również możliwość uruchomienia interfejsu przesyłającego SMS-y wysłane z telefonów komórkowych do klienta. W celu aktywacji dwukierunkowej wersji interfejsu konieczny jest kontakt z DCS.

SYGNATURY METOD

```
public bool SendSmses(
            string UserEmail,
            string Password,
            string[] Phones,
            string Message,
            string Sender,
            string TransactionId,
            DateTime? ValidityPeriod,
            int Priority,
            bool FlashSms,
            string CostCenter,
            out string ErrorMessage,
            out DateTime CurrentTime,
            out Guid[] MessageIds
        )
public bool SendSmses(
            string UserEmail,
            string Password,
            string[] Phones,
            string Message,
            string Sender,
            string TransactionId,
            DateTime? ValidityPeriod,
            int Priority,
            bool FlashSms,
            string CostCenter,
            out string ErrorMessage,
            out DateTime CurrentTime,
```

```
            out Guid[] MessageIds
        )
public bool GetUnreadDeliveryReports(
            string UserEmail,
            string Password,
            out string ErrorMessage,
            out DateTime CurrentTime,
            out DeliveryReport[] DeliveryReports
        )
public bool GetDeliveryReports(
            string UserEmail,
            string Password,
            Guid[] MessageIds,
            bool MarkAsRead,
            out string ErrorMessage,
            out DateTime CurrentTime,
            out DeliveryReport[] DeliveryReports
        )
public bool GetInputSmses(
            string UserEmail,
            string Password,
            DateTime? From,
            DateTime? To,
            string Recipient,
            bool MarkAsRead,
            out string ErrorMessage,
            out DateTime CurrentTime,
            out InputSms[] InputSms
            )
```

```
public bool GetUnreadInputSmses(
            string UserEmail,
            string Password,
            out string ErrorMessage,
            out DateTime CurrentTime,
            out InputSms[] InputSms
            )
public bool GetValidSenders(
            string UserEmail,
            string Password,
            out string[] Senders,
            out string ErrorMessage,
            out DateTime CurrentTime
  )
public struct DeliveryReport
    {
        public Guid ReportId;
        public string Phone;
        public Guid MessageId;
        public int Status;
        public DateTime DeliveryTime;
    }
    public struct InputSms
    {
        public Guid MessageId { get; set; }
        public string Phone { get; set; }
        public string Recipient { get; set; }
        public string Message { get; set; }
        public DateTime ReceivedTime { get; set; }
    }
```

UWIERZYTELNIENIE KLIENTA

Uwierzytelnienie klienta następuje za pomocą pary UserEmail / Password identyfikującej jednoznacznie klienta usługi HSMS. Dane te klient otrzymuje bezpośrednio od firmy DCS.

PARAMETRY METOD / POLA STRUKTUR DELIVERY REPORT / INPUTSMS.

NUMER TELEFONU

```
string Phone
string[] Phones
```

Numer telefonu ma być podawany zawsze w postaci międzynarodowej np: 48xxxxxxxxx.
W przypadku masowego wysyłania wiadomości podawana jest tablica numerów telefonów, zgodnie z powyższym formatem.

NADAWCA – NADPIS

```
string Sender
out string[] Senders,
```

Ciąg alfanumeryczny, który zostanie przekazany do operatora, jako nadpis nadawcy. Należy mieć na uwadze, że w różnych sieciach mogą być różne ustawienia, jeżeli chodzi o dozwoloną formę nadpisu.

Listę wszystkich dozwolonych nadpisów przypisanych do klienta zwraca metoda GetValidSenders. W przypadku potrzeby korzystania z innych nadpisów konieczny jest kontakt z DCS celem aktywnacji nadpisu.

ODBIORCA

```
string Recipient
```

Numer na który przyszedł SMS wejściowy.

Może być podany jako parametr do wyszukiwania odebranych SMS-ów.

TREŚĆ WIADOMOŚCI

```
string Message
```

System automatycznie będzie kodował wiadmości SMS jako GSM7 lub Unicode, w zalezności od tego czy w treści znajdowały się jakieś znaki spoza zestawu GSM7

Maksymalna długość wiadomości SMS w systemie HSMS to maksimum 4000 znaków

IDENTYFIKATOR TRANSAKCJI

```
string TransactionId
```

Unikalny zestaw znaków identyfikujący próbę wysłania SMS-a / grupy SMS-ów.
Kolejne próby z tym samym identyfikatorem:

- Zwrócą wartość true i te same parametry wyjściowe jeżeli wywołanie nastąpiło z tymi samymi parametrami.
- Zwrócą wartość false i komunikat błędu w przypadku gdyby parametry się różniły.

Maksymalna wielkość 128 znaków.

CZAS WAŻNOŚCI SMS-A

```
DateTime? ValidityPeriod
```

Czas ważności SMS-a podana jako chwila w czasie. W przypadku w którym czas bieżący jest większy niż czas ważności SMS-a próba wysyłki będzie odrzucona.
System HSMS nie wyśle SMS-a po upływie czasu ważności.
Jeżeli przyjmuje wartość NULL, DCS przyjmuje, że czas ważności nie został ustawiony i ważność wiadomości SMS zostanie ustawiona na domyślną dla danej sieci GSM.
Czas ważności musi zostać podany według czasu obowiązującego w Warszawie.

PRIORYTET

```
int Priority
```

Liczba z przedziału 0-3, gdzie 0 oznacza najniższy priorytet, a 3 najwyższy.

Wiadomość z niższym priorytetem zostanie wysłana jedynie wówczas, jeżeli w systemie nie czekają na wysłanie żadne wiadomości z wyższym priorytetem.

FLASH SMS

```
bool FlashSms,
```

Wartość **Prawda** oznacza, że wiadomość zostanie wysłana jako SMS typu Flash. Sms typu Flash jest wyświetlany na telefonie bezpośrednio po odebraniu i nie jest zapamiętywany w pamięci telefonu.

Ze względu na ograniczenia techniczne standardu GSM wiadomość typu Flash może składać się tylko z jednego segmentu i zawierać jedynie znaki z alfabetu GSM7. Maksymalna długość wiadomości Flash to 160 znaków.

CENTRUM KOSZTÓW

```
string CostCenter,
```

Nazwa centrum kosztów. Aby używać centrum kosztów najpierw musi zostać skonfigurowane w panelu klienta. Tylko wartości z listy stworzonych centrów kosztów są dozwolone.

W przypadku wyłączonej funkcjonalności pole powinno przyjmować wartość null

OZNACZ JAKO PRZECZYTANE

```
bool MarkAsRead,
```

Zaznacz jako przeczytane. W przypadku pobierania raportów dostarczenia lub odebranych SMS-ów dla konkretnych wiadomości istnieje opcja zaznaczenia ich jako przeczytanie, aby nie były już zwracane przy pomocy metody GetUnreadDeliveryReports.

ZAKRES CZASU

```
DateTime? From,

DateTime? To,
```

Parametry do wyszukiwania odebranych SMS-ów.

OPIS BŁĘDU.

```
out String ErrorMessage
```

Tekstowy opis błędu, w postaci zrozumiałej dla człowieka, w języku angielskim. W wypadku poprawnej operaci pole powinno być puste. Wszystkie metody interfejsu zwracając wartość logiczną **Prawda/Fałsz** informującą o prawidłowym zakończeniu operacji. Wartość ErrorMessage ma znaczenie tylko w przypadku jeżeli metoda zwróciła wartość **Fałsz**
Zmienna wyjściowa.

CZAS BIEŻĄCY.

```
out DateTime CurrentTime
```

Celem ominięcia problemów związnych z czasem system DCS podaje czas według własnego zegara.
Zmienna wyjściowa.

IDENTYFIKATOR WIADOMOŚCI

```
out Guid? MessageId

out Guid[] MessageId

Guid[] MessageIds,
```

Identyfikator wiadomości podany jako Guid, zwracany celem połaczenia wiadomości z raportem dostarczenia. Zmienna ma znaczenie wyłącznie jeżeli metody SendSms lub SendSmses zwracają wartość **Prawda**

Zmienna wyjściowa dla metod SendSms i SendSmses oraz zmienna wejściowa dla metody GetDeliveryReports.

RAPORT DOSTARCZENIA

```
public struct DeliveryReport
        {
            Guid ReportId;
            string Phone;
            Guid MessageId;
            int Status;
            DateTime DeliveryTime;
        }
```

Struktura z informacjami zawierająca pojedynczy opis raportu dostarczenia. Znaczenie pól wyjaśniono w pozostałych punktach.

IDENTYFIKATOR RAPORTU DOSTARCZENIA

```
Guid ReportId;
```

Jednoznacznie identyfikuje raport dostarczenia wiadomości.

STATUS DOSTARCZENIA

```
int Status
```

-1	wiadomość nie dostarczona (finalnie).
-2	wiadomość niedostarczona z powodu przedawnienia. Wiadomość nie została dostarczona w podanym czasie ValidityPeriod
-3	Wiadomosc została odrzucona przez operatora
+1	wiadomość dostarczona.
0	stan nieustalony, nie powinien wystepować.

Jako zasadę można badać warunek >0 i <0 . Kolejne statusy jeżeli będą dodawane będą zachowywać zasadę, że ujemne wartości dotyczą niedostarczenia wiadomości a dodatnie dotyczą dostarczenia wiadomości.

CZAS DOSTARCZENIA.

```
DateTime DeliveryTime
```

Czas uznania, że SMS został / finalnie nie został dostarczony. Według czasu w Warszawie.

CZAS ODEBRANIA

```
DateTime ReceivedTime
```

Czas, w którym SMS został zapisany w systemie DCS.

ODEBRANY SMS

```
public struct InputSms

{

    public Guid MessageId { get; set; }

    public string Phone { get; set; }

    public string Recipient { get; set; }

    public string Message { get; set; }

    public DateTime ReceivedTime { get; set; }

}
```

Wszystkie pola zgodne z definicjami podanymi wcześniej.

BEZPIECZEŃSTWO INTERFEJSU WEBSERWIS2SMS

- Dostęp do Interfejsu będzie wymagał autoryzacji loginem (adres e-mail) i hasłem
- Całość komunikacji pomiędzy DCS i Klient będzie szyfrowana z użyciem certyfikatów wydanych przez zaufane centrum certyfikacji.

INTERFEJS SIMPLEAPI

Interfejs SimpleAPI przeznaczony jest dla klientów chcących w możliwie jak najprostszy sposób wysyłać wiadomości SMS, nie potrzebujących zaawansowanych możliwości WebService2SMS.

Interfejs jest dostępny pod adresem https://api.hostedsms.pl/SimpleApi

Interfejs został wykonany w technologii ASP.NET Web API[5].

METODA SENDSMS

Interfejs oferuje jedną metodę SendSms z następującymi parametrami:

- UserEmail – Login do systemu
- Password – Hasło do systemu
- Sender – Przypisany dla klienta nadpis
- Phone – Numer telefonu odbiorcy
- Message – Treść wiadomości
- v – Opcjonalny parametr do mechanizmu antyspamowego (patrz niżej)

[5] Patrz: http://www.asp.net/web-api

W odpowiedzi system zwraca jedną z poniższych dwóch wartości:

- MessageId – Identyfikator wiadomości
- ErrorMessage – Komunikat błędu.

Znaczenie parametrów jest dokładnie takie samo jak w przypadku interfejsu WebService2SMS

Zapytania są wykonywane za pomocą HTTP POST. Formatowanie zapytania i odpowiedzi odbywa się za pomocą nagłówków Content-Type i Accept (patrz przykłady poniżej)

MECHANIZM ANTYSPAMOWY

Aby ochronić odbiorcę SMS-ów przed zalewem SMS-ów API wyposażone jest w mechanizm antyspamowy.

Powtórne wywołanie API danego dnia z tymi samymi parametrami spowoduje zwrócenie zapamiętanej odpowiedzi i nie spowoduje wysłania SMS-a.

W przypadku w którym istnieje potrzeba wysłania drugiego SMS-a o tej samej treści na ten sam numer telefonu można to zrobić ustawiając opcjonalny parametr „v” na dowolną unikalną wartość

PRZYKŁADY

Technologia ASP.NET WEB API pozwala na wykonywanie zapytań w różnym formacie. Poniżej przykład dla wywołania w postaci Request – FormData / Response – Json:

PRZYKŁAD – FORMDATA / JSON

Poniższy przykład pokazuje wywołanie serwisu za pomocą formatu Form Data oraz za pomocą formatu JSON .

W obu przypadkach odpowiedź będzie taka sama i zwrócona będzie w formacie JSON

Wywołanie (FORM DATA):

```
POST https://api.hostedsms.pl/SimpleApi HTTP/1.1
Accept: */*
Content-Type: application/x-www-form-urlencoded; charset=UTF-8

UserEmail=useremail%40dcs.pl&Password=correctpassword&Sender=TEST&Phone=48xxxxxxxxx&Message=TEST
```

Wywołanie (JSON)

```
POST https://api.hostedsms.pl/SimpleApi HTTP/1.1
Accept: application/xml
Content-Type: application/json; charset=UTF-8

{"UserEmail":"useremail@dcs.pl","Password":"correctpassword", "Sender":"TEST",  "Phone":"48502505988", "Message" : "TEST712"}
```

Odpowiedź Pozytywna:

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Server: Microsoft-IIS/7.0
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Tue, 25 Feb 2014 15:59:38 GMT
Content-Length: 52

{"MessageId":"ebc789b2-d111-4e9c-8a0e-d3ae0896364c"}
```

Odpowiedź Negatywna

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Server: Microsoft-IIS/7.0
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Tue, 25 Feb 2014 15:55:42 GMT
Content-Length: 38

{"ErrorMessage":"Invalid Credentials"}
```

PRZYKŁAD - XML

Poniżej odpowiednik poprzedniego przykładu w formacie XML

Wywołanie

```
POST https://api.hostedsms.pl/SimpleApi HTTP/1.1
Accept: application/xml
Content-Type: application/xml; charset=UTF-8

<SimpleSms>
  <UserEmail>useremail@dcs.pl</UserEmail>
  <Password>correctpassword</Password>
  <Sender>TEST</Sender>
  <Phone>48xxxxxxxxx</Phone>
  <Message>TEST712</Message>
</SimpleSms>
```

UWAGA: W przypadku wywołania XML kolejność parametrów ma znaczenie.

Odpowiedź Pozytywna

```
HTTP/1.1 200 OK
```

```
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/xml; charset=utf-8
Expires: -1
Server: Microsoft-IIS/8.0
X-AspNet-Version: 4.0.30319
X-SourceFiles: =?UTF-8?B?QzpccHJvamVjdHNcRENTLkhvc3RlZFNtc1xEQ1MuSFNNUy5TaW1wbGVBcGk=?=
X-Powered-By: ASP.NET
Date: Mon, 03 Mar 2014 16:06:32 GMT
Content-Length: 138

<MessageSend xmlns:i="http://www.w3.org/2001/XMLSchema-instance"><MessageId>1eba6eb2-888f-41cb-b146-9a1debdf8621</MessageId></MessageSend>
```

Odpowiedź Negatywna

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/xml; charset=utf-8
Expires: -1
Server: Microsoft-IIS/8.0
X-AspNet-Version: 4.0.30319
X-SourceFiles: =?UTF-8?B?QzpccHJvamVjdHNcRENTLkhvc3RlZFNtc1xEQ1MuSFNNUy5TaW1wbGVBcGk=?=
X-Powered-By: ASP.NET
Date: Mon, 03 Mar 2014 16:11:55 GMT
Content-Length: 115

<Error xmlns:i="http://www.w3.org/2001/XMLSchema-instance"><ErrorMessage>Invalid Credentials</ErrorMessage></Error>
```