SMS Gateway SDK

2.44

Wygenerowano przez Doxygen 1.7.6.1

Tue Oct 16 2012 14:57:33

Spis treści

1 SMS bezpośredni

1.1 Definicje

W niniejszym dokumencie zastosowano następujące słownictwo:

- *Partner* - użytkownik systemu Cashbill uruchamiający usługę SMS Bezpośredni.
- *Klient* - użytkownik sieci GSM korzystający z usługi partnera.

1.2 Opis usługi

Usługa SMS bezpośredni opiera się na bezpośredniej komunikacji SMS pomiędzy partnerem, a klientem końcowym. Każdy SMS wysłany przez klienta końcowego zostaje dostarczony do systemu informatycznego partnera, który może w dowolnej chwili odpowiedzieć wiadomością.

System Cashbill nie ingeruje w żadnen sposób w przesyłane treści.

W zależności od wariantu podpisanej umowy wysyłanie SMS może zostać obwarowane dodatkowymi ograniczeniami.

Do uruchomienia usługi SMS bezpośredni konieczne jest przygotowanie aplikacji obsługującej wiadomości. Gotowe rozwiązania są dostępne w pakiecie instalacyjnym, jednak do ich uruchomienia niezbędna jest ingerencja programisty.

1.2.1 Wymagania

Aby uruchomić SMS bezpośredni partner musi spełnić następujące wymagania:

- Formalne:
  1. Założenie konta w systemie Cashbill.
  2. Dostarczenie do personelu Cashbill regulaminu usługi, którą będzie obsługiwać SMS bezpośredni.
  3. Akceptacja regulaminu przez operatorów sieci komórkowych, w których będzie działała usługa.

4. Publikacja zaakceptowanego regulaminu.

- Techniczne:

1. Dostępny w publicznej sieci internet serwer zdolny do komunikacji SOAP.

2 Odbieranie wiadomości SMS

Przesyłanie wiadomości SMS do systemu partnera odbywa się z wykorzystaniem protokołu SOAP. Niniejszy pakiet instalacyjny zawiera gotowe oprogramowanie, zwalniające partnera z konieczności przygotowania własnego klienta i serwera SOAP, jak i gotowe szablony i pakiety realizujące większość typowych zadań w obsłudze SMS.

2.1 Obsługa wiadomości handlerem SMS

Zalecaną metodą obsługi wiadomości przychodzących jest przygotowanie własnej klasy handlera wiadomości (implementującej iSMSHandler) i dodanie jej do obieku bramki metodą addSMSHandler.

przykład:

```
require_once 'cashbillSms.php';

class cMojaObslugaSMS implements iSMSHandler
{
   public function handle( cMessageSMSIn $sms )
   {
     // tutaj należy dopisać obsłużenie przychodzącej wiadomości SMS
   }
}

$gateway = new cGateway( '<klucz tajny>', '<prefix>' );
$gateway->addSMSHandler( new cMojaObslugaSMS() );
$gateway->handleInput();
```

2.2 Obsługa wiadomości kolejkami

Alternatywną metodą obsługi wiadomości SMS jest odczytanie ich bezpośrednio z kolejki wiadomości bramki metodą receiveSMS. Każde wywołanie receiveSMS zwraca koleją odebraną wiadomość lub *null*, gdy kolejka jest pusta. Obsługa wiadomości sprowadza się do:

```
require_once 'cashbillSms.php';

$gateway = new cGateway( '<klucz tajny>', '<prefix>' );
$gateway->addSMSHandler( new cMojaObslugaSMS() );
$gateway->handleInput();

while( $sms = $gateway->receiveSMS() )
{
        // obsłużenie wiadomości SMS
}
```

2.3 Polskie znaki

Obecna wersja protokołu obsługuje polskie znaki diakrytyczne zarówno w wiadomościach przychodzących jak i w wychodzących. Niniejsze API obsługuje kodowanie utf-8.

Należy pamiętać, iż wysyłanie wiadomości z polskimi znakami powoduje zmianę kodowania SMS skutkującą zmniejszeniem długości pojedynczego SMS ze 160 znaków do 80. Dla wiadomości wieloczęściowych oznacza to skrócenie maksymalnego rozmiaru z 900 znaków do 450.

3 Wysyłanie wiadomości SMS

Przesyłanie wiadomości SMS do systemu Cashbill odbywa się z wykorzystaniem protokołu SOAP. Niniejszy pakiet instalacyjny zawiera gotowe oprogramowanie, zwalniające partnera z konieczności przygotowania własnego klienta i serwera SOAP, jak i gotowe szablony i pakiety realizujące większość typowych zadań w obsłudze SMS.

3.1 Tworzenie i wysyłanie wiadomości

Zalecaną metodą obsługi wiadomości wychodzących jest przygotowanie nowego obiektu cMessageSMSOut i wysłanie go poprzez metodę doSendMessageSMS bramki cGateway.

przykład:

```
require_once 'cashbillSms.php';

$gateway = new cGateway( '<klucz tajny>', '<prefix>' );
$sms = new cMessageSMSOut( '48xxxyyyzzz', '71880', 'Przykładowa wiadomość' );
$raport = $gateway->doSendMessageSMS( $sms );
```

4 Cykl życia wiadomości

Każda wiadomość SMS przetwarzana przez system Cashbill przechodzi przez szereg operacji. Zmiany statusu wiadomości przesyłane są do partnera w postaci raportów cMessageReport.

Aby odbierać i przetwarzać raporty o dostarczaniu wiadomości partner powinien utworzyć klasę implementującą iReportHandler i dodać ją do bramki z wykorzystaniem addReportHandler. Każdy cMessageReport zawiera identyfikator statusu, momentu wystąpienia oraz opis diagnostyczny.

W ramach niniejszego pakietu bramki dostarczony został gotowy handler obsługi raportów cReportLogger, zapisujący wszystkie zmiany statusów wiadomości w pliku.

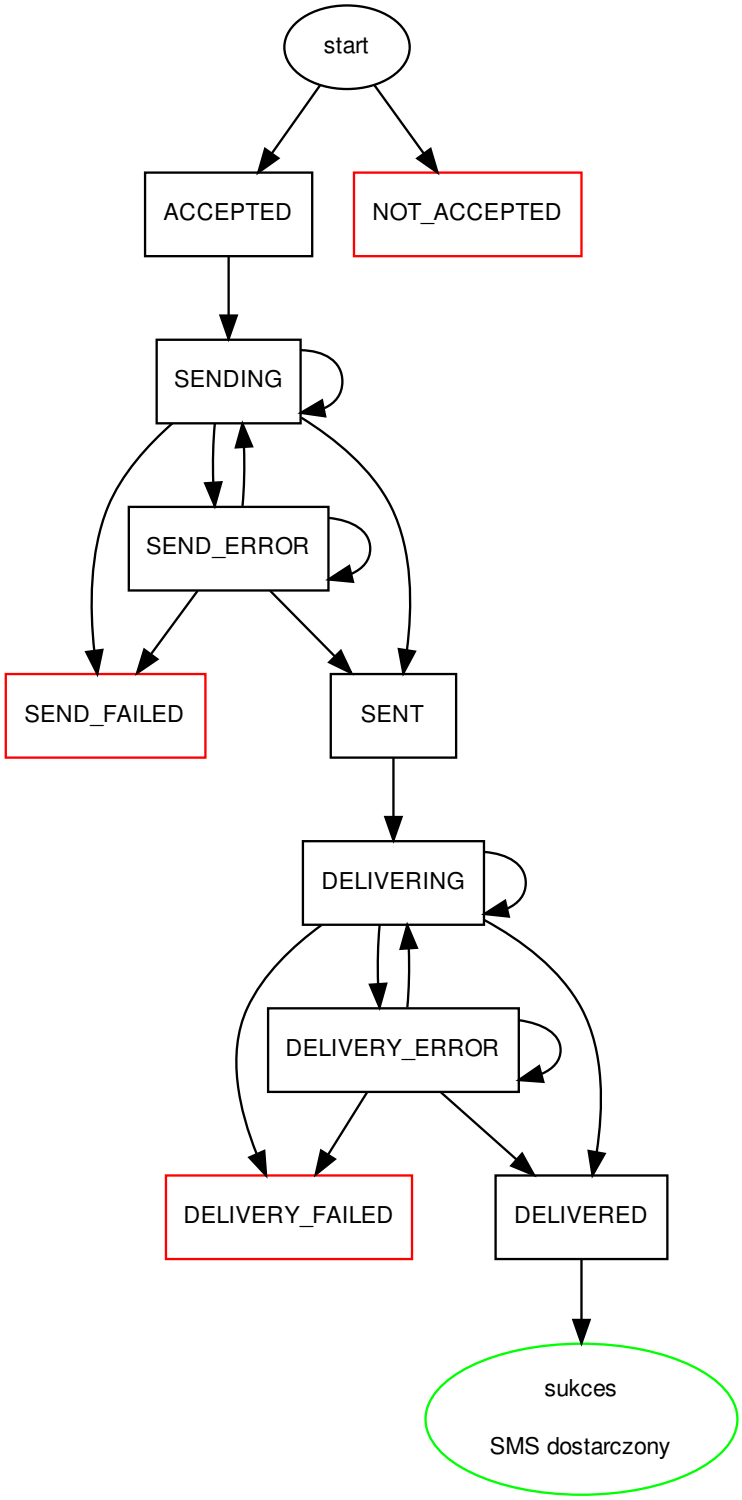
Przykładowe przetwarzanie raportu:

```
$gateway = new cGateway( '<klucz tajny>', '<prefiks>' );
$gateway->addReportHandler( new cReportLogger() );
$gateway->handleInput();
```

W całym cyklu życia wiadomości statusy mogą występować w następującej kolejności określonej poniższym grafem przejść.

Uwaga! Ze względu na transmitowanie danych o statusach przez publiczną sieć internetową i związane z tym możliwe opóźnienia może dojść do sytuacji, gdy wiadomości dotrą w innej kolejności. By ustalić prawdziwą koleność statusów wiadomości należy posłużyć się polem określającym czas wystąpienia w cMessageReport.

Czasy zmiany statusów podawane są z dokładnością do mikrosekundy.


start
ACCEPTED
NOT_ACCEPTED
SENDING
SEND_ERROR
SEND_FAILED
SENT
DELIVERING
DELIVERY_ERROR
DELIVERY_FAILED
DELIVERED
sukces
SMS dostarczony

5 Diagnostyka i testy

5.1 plikowy

W ramach niniejszego pakietu aplikacji dołączony został gotowy mechanizm diagnostyczny, umożliwiający zlokalizowanie większości błędów i problemów w procesie wdrażania usługi.

Aby uruchomić loggery diagnostyczne wystarczy po inicjalizacji bramki SMS dodać:

```
$smsGateway->addReportHandler   ( new cReportLogger()   );
$smsGateway->addSMSHandler      ( new cSMSLogger()      );
$smsGateway->addSMSErrorHandler( new cSMSErrorLogger() );
```

Każdy z loggerów przyjmuje opcjonalny parametr w postaci ścieżki do pliku, w którym powinien notować infrormacje. W przypadku pominięcia ścieżki, przyjmowany jest domyślny plik logu: log/sms.log.

6 Zabezpieczenia przed błedami

6.1 Unikalność identyfikatora

Transmisja danych w publicznej sieci internetowej obarczona jest ryzykiem wystąpienia błędu niezależnego od partnera i systemu Cashbill. Z tego względu protokół transmisji danych wyposażony został w mechanizmy automatycznej kontroli i korekcji błędu.

Protokół transmisji dopuszcza wielokrotne przesyłanie tej samej wiadomości jeżeli tylko istnieje jakaś wątpliwość co do poprawności jej przetworzenia. Aby uniknąć wielokrotnego wysłania tej samej wiadomości do klienta każda wiadomość *musi* być wyposażona w unikalny identyfikator.

Jeżeli w transmisji pojawi się ponownie już obsłużona wiadomość (wiadomość o identyfikatorze który został już poprawnie przetworzony), protokół transmisji przewiduje odpowiedź "ACCEPTED" oraz zignorowanie wiadomości.

6.1.1 Identyfikatory wiadomości wychodzacych

Jeżeli partner pominie identyfikator na dowolnym etapie wysyłania wiadomości bramka postara się utworzyć odpowiedni identyfikator automatycznie. Kryteria tworzenia identyfikatora zależne są od metody tworzenia wiadomości wychodzącej:

1. Dla wiadomości utworzonej metodą reply z cMessageSMSIn identyfikator powstaje na podstawie id wiadomości inicjującej oraz numeru odpowiedzi.
2. Dla wiadomości utworzonej bezpośrednio z cMessageSMSOut, w przypadku braku identyfikatora tworzony jest on na podstawie numeru docelowego, treści oraz czasu dostarczenia.

6.1.2 Identyfikatory wiadomości przychodzacych

Każda wiadomość dostarczana przez system Cashbill do serwera partnera posiada unikalny identyfikator. Ta sama wiadomość może zostać dostarczona wielokrotnie, jeżeli system Cashbill uzna, iż istnieje ryzyko błędu transmisji. W szczególności, ponowne dostarczenie wiadomości nastąpi gdy serwer partnera nie odpowiedział zgodnie z protokołem SOAP. Ta sytuacja może mieć następujące przyczyny:

1. Występują problemy z transmisją sieciową.
2. Wystąpiła awaria serwera partnera.
3. Oprogramowanie partnera zawiera błędy.
4. Serwer partnera zbyt długo przetwarza wiadomość (dłużej, niż timeout protokołu).
5. Serwer partnera odpowiedział wyjątkiem (SoapFault)

Ponowne dostarczanie wiadomości zapewnia, że każda wiadomość wysłana przez klienta zostanie dostarzcona do partnera.

W przypadku większości usług opartych o SMS bezpośredni ponowne przetworzenie wiadomości od klienta jako nowej jest nieporządane - skutkuje przyznaniem wielu kodów dostępu, zaliczeniem większej niż faktyczna liczby głosów w sondzie, itp.

Z tego względu zalecamy by dla wszystkich usług weryfikować unikalność identyfikatora. W przypadku stwierdzenia ponownego przetworzenia tej samej wiadomości handler partnera może zarządać zignorowania wiadomości wyjątkiem:

```
throw new cGatewayException( 'Wiadomość została już obsłużona',
     cGatewayException::IGNORE_MESSAGE );
```

Partner może wykorzystać dowolny opis wyjątku do celów diagnostycznych.

6.1.3 Gotowe rozwiazania

Aby uprościć implementację usług opartych o SMS bezpośredni w niniejszym pakiecie zawarto gotowe rozwiązania umożliwiające przechwycenie wielokrotnej transmisji tej samej wiadomości.

Uruchomienie gotowej obsługi wielokrotnej transmisji tej samej wiadomości Wystarczy *przed* właściwym handlerem obsługi wiadomości umieścić kod:

```
$smsGateway->addSMSHandler( new cSMSUniqeIdMySqlCheck( <parametry połączenia>
     [,nazwa tabeli] ) );
```

A po zakończeniu obsługi wiadomości:

```
$smsGateway->addSMSHandler( new cSMSUniqeIdMySqlDone( <parametry połączenia> [
     ,nazwa tabeli] ) );
```

W obu przypadkach klasy oczekują przekazania tego samego połączenia do bazy danych. Połączenie jest deklarowane z wykorzystaniem klasy cMySqlConnection.

Uzupełniony kod obsługi wiadomości wygląda tak:

```
$mysql = new cMySqlConnection( <host>, <login>, <hasło>, <baza danych> );
$smsGateway = new cGateway( <klucz tajny>, <prefiks> );

$smsGateway->addSMSHandler( new cSMSUniqeIdMysqlCheck( $mysql ));
$smsGateway->addSMSHandler( new cMojaObslugaSms() );
$smsGateway->addSMSHandler( new cSMSUniqeIdMysqlDone ( $mysql ));

$smsGateway->handleInput();
```

Gotowe rozwiązania zawarte w plikach template_∗.php zawierają te linie, jeżeli tylko jest konieczna do poprawnego funkcjonowania danego schematu usługi.

7 Konwencje kodowania i nazewnictwa

7.1 Konwencje nazewnicze

W niniejszej aplikacji i przykładach stosowane jest jednolite nazewnictwo klas, zmiennych, metod i funkcji. Wszystkie nazwy zaczynają się od małej litery, a kolejne słowa w nazwie zaczynają się dużą literą, np: *przykładowaNazwaPliku*.

7.1.1 Deklaracje i definicje klas

- przedrostki
  1. przedrostek *i* stosowany jest dla interface, na przykład: *iInterface*
  2. przedrostek *a* stosowany jest dla klas abstrakcyjnych, na przykład *aAbstrakt*
  3. przedrostek *c* stosowany jest dla pozostałych klas, na przykład *cKlasa*

7.1.2 Zmienne

- Nazwy zmiennych podlegają standardowym konwencjom wielkości liter, np: $przykladowaZmienna.
- API bramki nie korzysta z żadnych zmiennych globalnych.

7.1.3 Stałe

- Nazwy stałych zawierają wyłącznie duże litery. Słowa rozdzielane są znakiem podkreślenia, np: PRZYKLADOWA_STALA
- API bramka SMS Cashbill nie korzysta z żadnych stałych globalnych.

7.2 Konwencje kodowania

API bramki Cashbill jest napisane według paradygmatu *fluid*. Metody przeprowadzające operacje na obiekcie zwracają ten sam obiekt, umożliwiając łączenie kolejnych wywołań w mniejszej liczbie linii kodu.

Dlatego kod:

```
$smsGateway->addReportHandler  ( new cReportLogger()   );
$smsGateway->addSMSHandler     ( new cSMSLogger()      );
$smsGateway->addSMSErrorHandler( new cSMSErrorLogger() );
```

Jest równoważny:

```
$smsGateway->addReportHandler  ( new cReportLogger()   )
           ->addSMSHandler     ( new cSMSLogger()      )
           ->addSMSErrorHandler( new cSMSErrorLogger() );
```

7.3 Konwencje stosowane w przykładach

We wszystkich przykładach zawartych w niniejszej dokumentacji zastosowanie mają następujące konwencje zapisu:

1. W nawiasach trójkątnych zawarto treści, które partner musi uzupełnić. Np. <klucz tajny>="">, <has ło>="">> itp. Należy pamiętać, by podawane ciągi znaków zawierać w cudzysłowach, np. w miejsce <login> wpisujemy "mójlogin"
2. W nawiasach kwadratowych zawarto parametry opcjonalne, których nie trzeba umieszczać. Zapis <login> [, <has ło>="">] powinien być zastąpiony przez "mójlogin" lub "mójlogin","mojehasło".

8 Gotowe Szablony

8.1 Informacje ogólne

Szablony usług SMS bezpośredniego to gotowe rozwiązania realizujące wiele najczęściej spotykanych wariantów usług SMS. W większości przypadków do uruchomienia usługi wymagana będzie jedynie niewielka modyfikacja jednego z prezentowanych szablonów.

8.1.1 Wymagania gotowych rozwiazań

Gotowe rozwiązania obsługi SMS bezpośrednich stworzone zostały z myślą o zastosowaniu na powszechnie dostępnych platformach serwerowych.

- Serwer WWW dostępny w publicznej sieci internetowej.
- Interpreter PHP 5 lub nowszy, zintegrowany z serwerem WWW.

8.1.2 Instalacja szablonu

1. Wybrany plik szablonu usługi należy skopiować do pliku index.php lub innego, dostępnego z poziomu protokołu HTTP.
2. Adres URL pod którym dostępny będzie szablon należy przekazać personelowi systemu Cashbill w celu konfiguracji usługi.
    - Gotowe szablony usług zapisują dane diagnostyczne do podkatalogu log/. Należy upewnić się, że aplikacja WWW ma możliwość zapisu do tego katalogu.
3. Treść szablonu należy uzupełnić w zaznaczonych miejscach.

Po zakończeniu prac nad rozwojem usługi, ze względów bezpieczeństwa, zalecamy usunięcie wszystkich plików template_$*$ z serwera.

8.2 Bezpośrednia odpowiedź na SMS

Ten szablon powinien być stosowany dla usług, w których klient po wysłaniu SMS bezzwłocznie otrzymuje określoną przez partnera odpowiedź. Partner otrzymuje wszystkie informacje o SMS wysłanym przez klienta, decyduje o treści odpowiedzi i bezzwłocznie odsyła ją do klienta.

Szablon zawarty jest w pliku template_100_directReply.php. Zalecamy skopiowanie treści szablonu do pliku index.php i wprowadzenie własnej obsługi wiadomości w wyznaczonym miejscu.

Wymagane ingerencje w treść zaznaczone są ciągami znaków todo.

8.3 Opóźniona odpowiedź na SMS

Ten szablon powinien być stosowany dla usług, w których klient wysyłając SMS rejestruje się w usłudze, która w późniejszym terminie wyśle do niego z góry określoną treść lub treści. Partner otrzymuje kompletną informację o SMS wysłanym przez klienta i na jej podstawie decyduje kiedy i jakie odpowiedzi zostaną wysłane.

Szablon powinien być stosowany do usług, w których można z góry określić jakie treści będą odsyłane klientowi.

Szablon zawarty jest w pliku template_200_delayedReply.php. Zalecamy skopiowanie treści szablonu do pliku index.php i wprowadzenie własnej obsługi wiadomości w wyznaczonym miejscu.

Wymagane ingerencje w treść zaznaczone są ciągami znaków todo.

9 Indeks przestrzeni nazw

9.1 Lista przestrzeni nazw

Tutaj znajdują się wszystkie udokumentowane przestrzenie nazw wraz z ich krótkimi opisami:

10 Indeks struktur danych

10.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

11 Indeks struktur danych

11.1 Struktury danych

Tutaj znajdują się struktury danych wraz z ich krótkimi opisami:

12 Dokumentacja przestrzeni nazw

12.1 Dokumentacja przestrzeni nazw cashbill

12.1.1 Opis szczegółowy

Autor

Mariusz Chwalba <koder@mediasystems.pl>

Copyright

2009 Media Systems

Interface obsługi bramki SMS cashbill.

2009-08-27 messageSqlBased.php

Autor

Mariusz Chwalba <koder@mediasystems.pl>

Copyright

2009 Media Systems

Klasy pomocnicze do obsługi wiadomości przez bazę MySql

2009-08-27 messageUnique.php

Autor

Mariusz Chwalba <koder@mediasystems.pl>

Copyright

2009 Media Systems

Obsługa unikalności identyfikatora.

2009-08-24 template_directReply.php

Autor

Mariusz Chwalba <koder@mediasystems.pl>

Copyright

2009 Media Systems

13 Dokumentacja struktur danych

13.1 Dokumentacja klasy aMessage

Diagram dziedziczenia dla aMessage

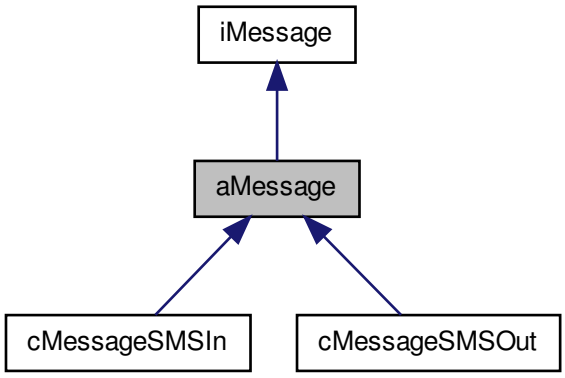
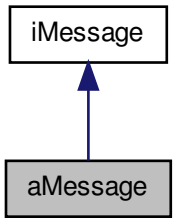


Diagram współpracy dla aMessage:



Metody publiczne

- setGateway (cGateway $gateway)
- **doSign** ()
- **verifySign** ()
- **computeSign** ()
- **getPrefix** ()

Pola danych

- $id
- $msisdn
- $la
- $text
- $prefix
- $time
- $sign

Atrybuty chronione

- $gateway

13.1.1 Dokumentacja funkcji składowych

13.1.1.1 setGateway (cGateway $ *gateway*)

Przypisanie bramki obsługującej wiadomość. Metoda wywoływana automatycznie przez bramkę

Parametry

c-Gateway	*$gateway*	

Zwraca

aMessage

13.1.2 Dokumentacja pól

13.1.2.1 $gateway [protected]

Bramka obsługująca wiadomość.

13.1.2.2 $id

Identyfikator wiadomości.

W przypadku wątpliwości co do statusu obsłużenia wiadomości zarówno partner jak i Cashbill może przesłać ponownie tą samą wiadomość. Jeżeli wiadomość została już wcześniej obsłużona poprawnie, ponowna transmisja powinna zostać zignorowana, z dowolnym raportem

13.1.2.3 $la

Numer bramki Large Account, przez którą wiadomość została odebrana bądź zostanie wysłana.

13.1.2.4 $msisdn

Adres, pod który ma zostać dostarczona wiadomość. W przypadku wiadomości SMS/-MMS oczekiwany jest numer telefonu z kodem kraju, np: 48xxxyyyzzz

13.1.2.5 $prefix

Identyfikator usługi partnera.

13.1.2.6 $sign

Sygnatura wiadomości. Podpis MD5 najważniejszy parametr wiadomości, umożliwiający zweryfikowanie nadawcy.

Automatycznie sprawdzany przez metodę verifySign() dla wiadomości przychodzących Automatycznie tworzony przez metodę doSign() dla wiadomości wychodzących

13.1.2.7 $text

Treść przesłanej wiadomości.

Długość możliwej do przesłania treści zależna jest od medium transmisji. Maksymalna długość tego pola to 900 znaków. W przypadku przekroczenia limitu wiadomości SMS system wyśle wiele wiadomości, które zostaną złożone w całość w telefonie klienta

13.1.2.8 $time

Czas dostarczenia wiadomości, w formacie unix, z mikrosekundami. W przypadku wiadomości wysyłanej może zostać pominięty - wtedy wiadomość zostanie wysłana od razu.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- cashbillSms.php

13.2 Dokumentacja klasy cGateway

Metody publiczne

- __construct ($privKey, $defaultPrefix=null, $mainUrl=null)
- handleInput ()
- addSMSHandler (iSMSHandler $handler)
- addSMSErrorHandler (iSMSErrorHandler $handler)
- addReportHandler (iReportHandler $handler)
- **soap** ()
- doGetMessageReports ($id)
- doSendMessageSMS (cMessageSMSOut $SMS)
- doSendMessagesSMS (cMessageSMSOutList $msgList)
- doSendMessageReport (cMessageReport $report)
- **doReceiveMessageSMS** (cMessageSMSIn $SMS)
- **doReceiveMessageReport** (cMessageReport $report)
- recieveSMS ()
- recieveReport ()

Pola danych

- $remoteWsdl = 'http://ssl.cashbill.pl/sms/gateway/smsBack/?wsdl'
- $localWsdl = 'template_PartnerGateway.wsdl'
- $mainUrl
- $privKey
- $defaultPrefix
- $wsdlCache = WSDL_CACHE_BOTH
- **$wsdlTrace** = false

Metody chronione

- **getMainUrl** ()
- **handleError** (cMessageSMSIn $SMS, $error)

Atrybuty chronione

- **$wsdlClassMap**
- **$soap**
- **$messages** = array()
- **$reports** = array()
- **$SMSHandlers** = array()
- **$SMSErrorHandlers** = array()
- **$reportHandlers** = array()

13.2.1 Opis szczegółowy

Bramka SMS/MMS systemu Cashbill

Przed zapoznaniem się ze szczegółową dokumentacją bramki zalecamy obejrzenie schematów gotowych rozwiązań, zawartych w plikach template_∗.php.

Skorzystanie z gotowego schematu zapewnia najszybsze uruchomienie usługi.

13.2.2 Dokumentacja konstruktora i destruktora

13.2.2.1 __construct ($ *privKey*, $ *defaultPrefix* = `null`, $ *mainUrl* = `null`)

Inicjalizacja obiektu bramki - odbiornika i nadajnika wiadomości SMS

Parametry

string	*$privKey*	- Klucz prywatny bramki (do odczytania w panelu systemu, po uruchomieniu usługi)
string	*$default-Prefix*	- Domyślny identyfikator usługi bramki, przypisywany wiadomościom wychodzącym gdy nie został określony indywidualny identyfikator dla wiadomości
string	*$mainUrl*	- Opcjonalny adres serwera SOAP odbierającego wiadomości. W przypadku pominięcia odczytywany jest z konfiguracji serwera.

13.2.3 Dokumentacja funkcji składowych

13.2.3.1 addReportHandler (iReportHandler $ *handler*)

Dodanie handlera do obsługi raportu.

Parametry

iReport-Handler	*$handler*	- obiekt który będzie obsługiwał raporty o przetwarzaniu SMS

Zwraca

cGateway

13.2.3.2 addSMSErrorHandler (iSMSErrorHandler $ *handler*)

Dodanie handlera do obsługi błędów transmisji wiadomości SMS.

Parametry

iSMS-Error-Handler	*$handler*	- obiekt który będzie obsługiwał błąd wiadomości SMS

Zwraca

cGateway

13.2.3.3 addSMSHandler (iSMSHandler $ *handler*)

Dodanie handlera do obsługi wiadomości SMS.

Parametry

iSMS-Handler	*$handler*	- klasa która będzie obsługiwała przychodzącą wiadomość SMS

Zwraca

cGateway

13.2.3.4 doGetMessageReports ($ *id*)

Zwraca wszystkie raporty związane z wiadomością o podanym ID.

Parametry

strign	*$id*	- identyfikator wiadomości do sprawdzenia.

Zwraca

cMessageReport[] - tablica wszystkich raportów dotyczących wiadomości

13.2.3.5 doSendMessageReport (cMessageReport $ *report*)

Wysyła do systemu Cashbill raport o statusie przetwarzania wiadomości. System - Cashbill wymaga jedynie raportu "ACCEPTED". Wszystkie inne raporty posłużą jedynie w celach diagnostycznych.

Parametry

cMessageReport	*$report*	- raport do wysłania

13.2.3.6 doSendMessageSMS (cMessageSMSOut $ *SMS*)

Wysłanie wiadomości SMS poprzez bramkę Cashbill.

Jedną z metod wysłania nowej wiadomości SMS do klienta jest przygotowanie struktury cMessageSMSOut i przekazanie jej do metody doSendMessageSMS bramki.

Parametry

cMessageSMSOut	*$SMS*	- wiadomość do wysłania

Zwraca

cMessageReport - raport o dostarczeniu wiadomości

13.2.3.7 doSendMessagesSMS (cMessageSMSOutList $ *msgList*)

Wysyła do systemu listę wiadomości do wysłania. W przypadku rozsyłania dużej liczby wiadomości jest to zalecana metoda komunikacji z systemem Cashbill. Indywidualne wiadomości mogą mieć różne czasy dostarczenia.

Faktyczna przepustowość rozsyłania wiadomości jest zależna od obciążenia systemu GSM operatora. W większości przypadków

Parametry

cMessageSMSOutList	*$msgList*	- lista wiadomości do rozesłania

Zwraca

cReportList - lista raportów o przyjęciu wiadomości

13.2.3.8 handleInput ( )

Obsłużenie requestu SOAP. Metodę należy wywoływać w serwerze SOAP odbierającym wiadomości z systemu Cashbill, którego URL został podany przez partnera w trakcie konfigurowania usługi. Metoda przechwytuje całą komunikację SOAP.

13.2.3.9 recieveReport ( )

Zwraca jeden odebrany raport

Alternatywną do handlerów metodą obsługi raportów wiadomości jest skorzystanie z metody recieveReport(), która zwraca kolejne odebrane raporty o dostarczanych wiadomościach. Ze względu na brak możliwości raportowania błędów zalecamy jednak by korzystać z handlerów.

Zwraca

cMessageReport

13.2.3.10 recieveSMS ( )

Zwraca pojedynczą odebraną wiadomość.

Alternatywną do handlerów metodą obsługi wiadomości jest skorzystanie z metody recieveSMS(), która zwraca kolejne wiadomości wysłane przez użytkowników. Ze względu na brak możliwości raportowania błędów zalecamy jednak by korzystać z handlerów.

Zwraca

cMessageSMSIn

13.2.4 Dokumentacja pól

13.2.4.1 $defaultPrefix

Domyślny identyfikator usługi, którym posługuje się bramka gdy nie został zdefiniowany indywidualny identyfikator usługi dla wychodzącej wiadomości SMS.

13.2.4.2 $localWsdl = 'template_PartnerGateway.wsdl'

Adres WSDL serwera lokalnego, do którego Cashbill wysyła wiadomości

13.2.4.3 $mainUrl

URL na którym umieszczony jest serwer SOAP odbierający wiadomości

13.2.4.4 $privKey

Klucz prywatny bramki

13.2.4.5 $remoteWsdl = 'http://ssl.cashbill.pl/sms/gateway/smsBack/?wsdl'

Adres WSDL serwera Cashbill służącego do wysyłania wiadomości.

13.2.4.6 $wsdlCache = WSDL_CACHE_BOTH

Poziom cache WSDL klienta SOAP.

WSDL_CACHE_NONE WSDL_CACHE_DISK WSDL_CACHE_MEMORY WSDL_CACHE_BOTH

W czasie rozwoju aplikacji zalecamy stosowanie WSDL_CACHE_NONE. Po zakończeniu prac, by przyśpieszyć funkcjonowanie obsługi SMS, zalecane jest stosowanie WSDL_CACHE_MEMORY lub WSDL_CACHE_BOTH.

13.2.4.7 $wsdlClassMap [protected]

Wartość początkowa:

```
array(

                                'cMessageReport'     => 'cMessageReport',
                                'cMessageSMSIn'      => 'cMessageSMSIn',
                                'cMessageSMSOut'     => 'cMessageSMSOut',
                                'cMessageReportList' => 'cMessageReportList',
                                'cMessageSMSOutList' => 'cMessageSMSOutList',

                        )
```

Dokumentacja dla tej klasy została wygenerowana z pliku:

- cashbillSms.php

13.3 Dokumentacja klasy cGatewayException

Pola danych

- const IGNORE_MESSAGE = 100
- const RETRY_LATER = 200

13.3.1 Opis szczegółowy

Wyjątek bramki SMS

13.3.2 Dokumentacja pól

13.3.2.1 const IGNORE_MESSAGE = 100

Zignoruj wiadomość. Ten wyjątek nie powoduje zgłoszenia błędu, a jedynie zignorowanie wiadomości. Powinien być stosowany w handlerach obsługi unikalności identyfikatorów wiadomości.

13.3.2.2 const RETRY_LATER = 200

Prześlij tą wiadomość ponownie później.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- cashbillSms.php

13.4 Dokumentacja klasy cLoggerException

Pola danych

- const CANNOT_WRITE_LOG = 100

13.4.1 Dokumentacja pól

13.4.1.1 const CANNOT_WRITE_LOG = 100

Brak możliwości zapisu do pliku logowania.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- messageLogger.php

13.5 Dokumentacja klasy cMessageReport

Pola danych

- const ACCEPTED = 'ACCEPTED'
- const NOT_ACCEPTED = 'NOT_ACCEPTED'
- const SENDING = 'SENDING'
- const SENT = 'SENT'
- const SEND_ERROR = 'SEND_ERROR'
- const SEND_FAILED = 'SEND_FAILED'
- const DELIVERING = 'DELIVERING'
- const DELIVERED = 'DELIVERED'
- const DELIVERY_ERROR = 'DELIVERY_ERROR'
- const DELIVERY_FAILED = 'DELIVERY_FAILED'
- $id
- $status
- $errorstr
- $time
- $numberOfParts
- $partNumber

13.5.1 Opis szczegółowy

Raport z przetwarzania wiadomości

Struktura zawiera informacje o procesie przetwarzania wiadomości w systemie cashbill lub u partnera. Raport składa się ze statusu (jednej ze stałych zadeklarowanych w ciele klasy), momentu wystąpienia i treści diagnostycznej.

Raporty mają głównie znaczenie diagnostyczne - pozwalają ustalić co się dzieje z wiadomością po odebraniu przez infrastrukturę Cashbill, aż do dostarczenia do klienta końcowego.

13.5.2 Dokumentacja pól

13.5.2.1 $errorstr

Opis diagnostyczny statusu, dostarczony przez podmiot przetwarzający wiadomość.

13.5.2.2 $id

Identyfikator wiadomości, której dotyczy raport

13.5.2.3 $numberOfParts

Liczba części, z których składa się przetwarzana wiadomość

13.5.2.4 $partNumber

Numer części, której dotyczy raport

13.5.2.5 $status

Status wiadomości. Stałe statusów zadeklarowane są w cMessageReport

13.5.2.6 $time

Moment wystąpienia statusu, w formacie unix timestamp, z mikrosekundami

13.5.2.7 const ACCEPTED = 'ACCEPTED'

STATUS: Wiadomość zaakceptowana przez bramkę. Status wymagany. Wiadomości które nie miały tego statusu będą ponownie przesyłane przez system.

13.5.2.8 const DELIVERED = 'DELIVERED'

STATUS: Wiadomość została dostarczona do klienta końcowego. Status finalny.

13.5.2.9 const DELIVERING = 'DELIVERING'

STATUS: Wiadomość w trakcie transmisji w sieci GSM.

13.5.2.10 const DELIVERY_ERROR = 'DELIVERY_ERROR'

STATUS: Błąd dostarczania wiadomości. Szczegóły w polu diagnostycznym. - Wiadomość będzie automatycznie dostarczana ponownie.

13.5.2.11 const DELIVERY_FAILED = 'DELIVERY_FAILED'

STATUS: Wiadomość nie może zostać dostarczona. Status finalny.

13.5.2.12 const NOT_ACCEPTED = 'NOT_ACCEPTED'

STATUS: Wiadomość nie została przyjęta. Szczegóły w polu diagnostycznym.

13.5.2.13 const SEND_ERROR = 'SEND_ERROR'

STATUS: Wystąpił błąd przy wysyłaniu wiadomości. Będzie przeprowadzona kolejna próba wysłania.

13.5.2.14 const SEND_FAILED = 'SEND_FAILED'

STATUS: Wiadomość nie może być wysłana. Szczegóły błędu w polu diagnostycznym. Status finalny.

13.5.2.15 const SENDING = 'SENDING'

STATUS: Wiadomość w czasie transmitowania

13.5.2.16 const SENT = 'SENT'

STATUS: Wiadomość wysłana

Dokumentacja dla tej klasy została wygenerowana z pliku:

- cashbillSms.php

13.6 Dokumentacja klasy cMessageReportList

Pola danych

- $report

13.6.1 Opis szczegółowy

Lista raportów.

Zawiera w sobie listę raportów dotyczących wiadomości. Jest przekazywany z systemu Cashbill, gdy konieczne jest przekazanie więcej niż jednego raportu.

Raporty mogą dotyczyć jednej lub wielu wiadomości.

13.6.2 Dokumentacja pól

13.6.2.1 $report

Tablica raportów

Dokumentacja dla tej klasy została wygenerowana z pliku:

- cashbillSms.php

13.7 Dokumentacja klasy cMessageSMSIn

Diagram dziedziczenia dla cMessageSMSIn

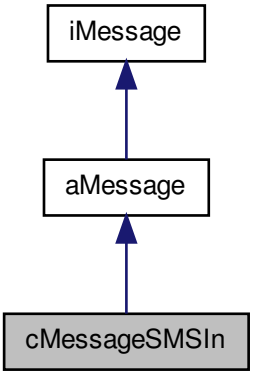
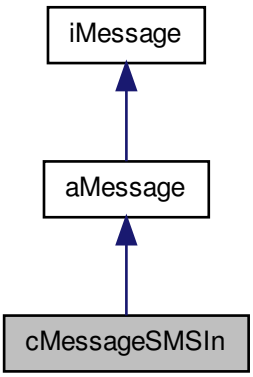


Diagram współpracy dla cMessageSMSIn:



Metody publiczne

- reply ($text, $nr=1, $time=0)
- makeReport ($status, $detail= ”, $time=null)
- isCompelte ()

Pola danych

- $net
- $numberOfParts
- $recivedParts

13.7.1 Opis szczegółowy

Wiadomość SMS przychodząca (wysłana przez klienta).

Przychodzące od klientów wiadomości są wysyłane do handlerów obsługi wiadomości, dodanych do bramki przez addSMSHandler.

Handler wiadomości to szkieletowa klasa obejmująca jedynie metodę handle, do której przekazywana jest otrzymana wiadomość.

Uruchomienie obsługi SMS przychodzących sprowadza się więc do utworzenia własnego handlera i wpisania w metodzie handle odpowiedniej obsługi wiadomości.

Do bramki może zostać przypisane wiele handlerów. Każdy z nich otrzyma przetwarzaną wiadomość. Taka konstrukcja umożliwia proste i szybkie dołączenie do istniejącego oprogramowania klienta gotowych rozwiązań Cashbill, jak na przykład logowanie diagnostyczne.

13.7.2 Dokumentacja funkcji składowych

13.7.2.1 isCompelte ( )

Sprawdzenie, czy wiadomość jest kompletna - czy operator wysłał wszystkie części wiadomości (dla wiadomości powyżej 160 znaków).

Jeżeli wiadomość jest niekompletna, oznacza to, że pole text nie zawiera wszystkich znaków które wysłał użytkownik.

Zwraca

boolean

13.7.2.2 makeReport ($ *status*, $ *detail* = `''`, $ *time* = `null`)

Przygotowanie raportu

Parametry

string	*$status*	- status, jaki chcemy raportować
string	*$detail*	- tekst diagnostyczny (opcjonalne)
float	*$time*	- unix timestamp momentu, w którym wystąpił status (opcjonalny)

Zwraca

cMessageReport

13.7.2.3 reply ($ *text*, $ *nr* = 1, $ *time* = 0)

Wygeneruj odpowiedź na tego SMSa

Parametry

string	*$text*	- treść odpowiedzi
int	*$nr*	- numer odpowiedzi (opcjonalnie)
int	*$time*	- unix timestamp momentu, w którym powinna zostać dostarczona wiadomość (opcjonalnie)

Zwraca

cMessageSMSOut

13.7.3 Dokumentacja pól

13.7.3.1 $net

Operator telefonii GSM, z usług którego korzysta klient

13.7.3.2 $numberOfParts

Liczba części, z których składa się wiadomość. Wiadomości powyżej 160 znaków wysyłane są przez telefony komórkowe w częściach.

13.7.3.3 $recivedParts

Liczba odebranych części wiadomości. W przypadku problemów w komunikacji GSM wiadomość może dotrzeć niekompletna. To pole pozwala na wykrycie tego problemu.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- cashbillSms.php

13.8 Dokumentacja klasy cMessageSMSOut

Diagram dziedziczenia dla cMessageSMSOut

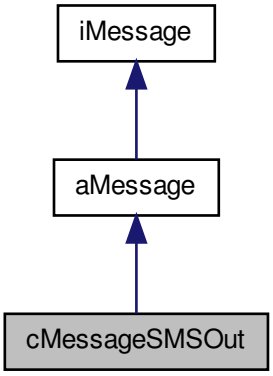
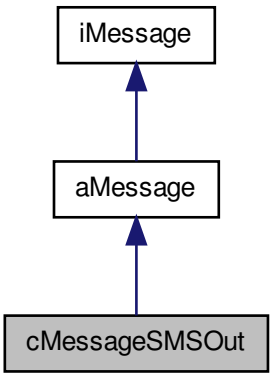


Diagram współpracy dla cMessageSMSOut:



Metody publiczne

- __construct ($msisdn, $la, $text, $time=null, $prefix=null, $id=null)
- send ()

Pola danych

- $laOverride
- $replyId

13.8.1 Opis szczegółowy

Wiadomość SMS wychodząca (odpowiedź do klienta)

Struktura zawiera kompletną informację o wiadomości wysyłanej do klienta GSM.

Aby wysłać wiadomość do klienta należy utworzyć nową strukturę przez new cMessageSMSOut(<parametry>) i przekazać ją do bramki.

Dostępne opcje to:

1. `$sms->setGateway( $gateway )->send();`
2. `$gateway->doSendMessageSMS( $sms );`

gdzie:

$sms - to nowo utworzona wiadomość cMessageSMSOut $gateway - to zainicjowany obiekt cGateway

13.8.2 Dokumentacja konstruktora i destruktora

13.8.2.1 __construct ($ *msisdn*, $ *la*, $ *text*, $ *time* = `null`, $ *prefix* = `null`, $ *id* = `null`)

Utworzenie nowej wiadomości. W przypadku odpowiadania na istniejącą wiadomość, należy skorzystać z metody reply() wiadomości źródłowej. Ustawia ona wszystkie konieczne pola, pozwalając na dodanie samej treści wiadomości zwrotnej

Parametry

string	*$msisdn*	- numer telefonu adresata
string	*$la*	- numer bramki. usługa musi być uruchomiona na tej bramce
string	*$text*	- treść wiadomości do wysłania do klienta
float	*$time*	- moment wysłania wiadomości (unix timestamp). pominięcie spowoduje wysłanie natychmiastowe
string	*$prefix*	- identyfikator (prefiks) usługi. Pominięcie spowoduje ustalenie domyślnego dla bramki.
string	*$id*	- unikalny identyfikator wiadomości. Pominięcie spowoduje wygenerowanie unikalnego identyfikatora.

13.8.3 Dokumentacja funkcji składowych

13.8.3.1 send ( )

Wysyła tą wiadomość.

Zwraca

cMessageSMSOut

13.8.4 Dokumentacja pól

13.8.4.1 $laOverride

Pole pozwalające ustawić inny niż domyślne źródło SMS. Dostępne tylko w przypadku wiadomości marketingowych (płatnych od każdej odpowiedzi)

W przypadku odpowiedzi na SMS Premium Rate pole ignorowane.

13.8.4.2 $replyId

Identyfikator wiadomości, na którą odpowiedzią jest niniejsza.

Opcjonalność tego pola jest zależna od podpisanej umowy. W przypadku usług o podwyższonym ryzyku system wymaga, by każda wysyłana odpowiedź zawierała identyfikator wiadomości, której dotyczy.

Nawet w przypadku gdy ta informacja jest opcjonalna zalecamy wypełnianie tego pola ze względu na ułatwienie i przyśpieszenie procesu reklamacyjnego.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- cashbillSms.php

13.9 Dokumentacja klasy cMessageSMSOutList

Metody publiczne

- __construct (cGateway $gateway)
- add (cMessageSMSOut $msg)
- makeBulk ($text, $msisdns, $la, $prefix=null, $time_from=0, $time_to=0)
- send ()

Pola danych

- **$message**

Atrybuty chronione

- **$gateway**

13.9.1 Opis szczegółowy

Lista wiadomości wychodzących.

Zawiera wiele wiadomości wychodzących, które mogą zostać wysłane jednocześnie w procesie rozsyłki masowej. Dostarcza również gotowych metod, dzięki którym można łatwo przygotować masową rozsyłkę.

13.9.2 Dokumentacja konstruktora i destruktora

13.9.2.1 __construct (cGateway $ *gateway*)

Przygotowuje listę wiadomości. Ponieważ wszystkie wiadomości muszą być rozsyłane tą samą bramką, należy tą bramkę podać jako parametr konstruktora.

Parametry

cGateway	*$gateway*	- bramka, przez którą będą rozsyłane wiadomości.

13.9.3 Dokumentacja funkcji składowych

13.9.3.1 add (cMessageSMSOut $ *msg*)

Dodaje wiadomość do rozsyłki masowej.

Parametry

cMessageSMSOut	*$msg*	- wiadomość do rozesłania

Zwraca

cMessageSMSOutList

13.9.3.2 makeBulk ($ *text*, $ *msisdns*, $ *la*, $ *prefix* = `null`, $ *time_from* = `0`, $ *time_to* = `0`)

Przygotowuje wysyłkę wiadomości masowej (bulk).

Parametry

$text	- treść wiadomości
$msisdns	- tablica numerów telefonicznych odbiorców, w formacie (48xxxyyyzzz)
$la	- bramka, z której ma być prowadzona rozsyłka
$prefix	- identyfikator usługi, której dotyczy rozsyłka
$time_from	- czas rozpoczęcia rozsyłki (unix timestamp)
$time_to	- czas zakończenia rozsyłki (unix timestamp)

Zwraca

cMessageSMSOutList

13.9.3.3 send ( )

Wysyła wszystkie wiadomości przez zadeklarowaną bramkę.

Zwraca

cMessageReportList

Dokumentacja dla tej klasy została wygenerowana z pliku:

- cashbillSms.php

13.10 Dokumentacja klasy cMySqlConnection

Metody publiczne

- __construct ($host, $login, $password, $database)

Pola danych

- **$host**
- **$login**
- **$password**
- **$database**

13.10.1 Dokumentacja konstruktora i destruktora

13.10.1.1 __construct ($ *host*, $ *login*, $ *password*, $ *database*)

Połączenie do bazy danych MySql

Parametry

$host	- serwer bazy danyh
$login	- login do serwera
$password	- hasło do serwera
$database	- baza danych do użycia

Dokumentacja dla tej klasy została wygenerowana z pliku:

- messageSqlBased.php

13.11 Dokumentacja klasy cReportLogger

Diagram dziedziczenia dla cReportLogger

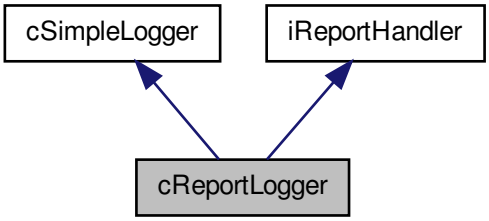
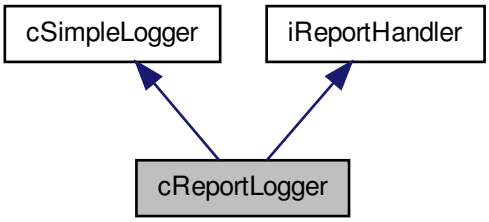


Diagram współpracy dla cReportLogger:



Metody publiczne

- **handle** (cMessageReport $report)

13.11.1 Opis szczegółowy

Logger plikowy do raportów SMS

Zapisuje wszystkie szczegóły raportów do pliku

Po dodaniu klasy do bramki cGateway (metodą addSMSReportHandler) wszystkie raporty o dostarczeniu wiadomości zostaną zapisane we wskazanym pliku. Pełny opis

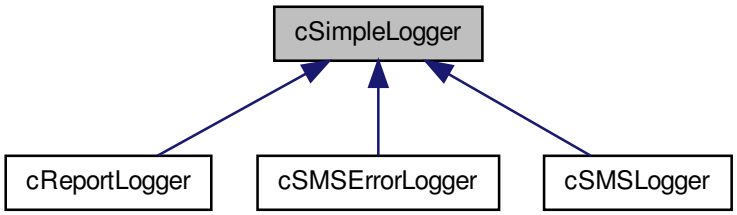
znaczenia raportów znajduje się w załączniku "graf przejść stanów wiadomości", oraz w dokumentacji klasy cMessageReport.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- messageLogger.php

13.12 Dokumentacja klasy cSimpleLogger

Diagram dziedziczenia dla cSimpleLogger



Metody publiczne

- __construct ($logFile=null)

Pola danych

- **$logFile**

Metody chronione

- **microDate** ($format, $date)
- **log** ($message, $timeStamp=null)
- **write** ($msg)

Atrybuty chronione

- **$fh**

13.12.1 Dokumentacja konstruktora i destruktora

13.12.1.1 __construct ($ *logFile* = `null`)

Uruchomienie loggera

Parametry

string	*$logFile*	- nazwa pliku zawierająca log. W przypadku braku będzie zapisywać do log/sms.log

Dokumentacja dla tej klasy została wygenerowana z pliku:

- messageLogger.php

13.13 Dokumentacja klasy cSMSDelayedReply

Metody publiczne

- **handle** (cMessageSMSIn $sms)

Dokumentacja dla tej klasy została wygenerowana z pliku:

- template_200_delayedReply.php

13.14 Dokumentacja klasy cSMSDirectReply

Metody publiczne

- **handle** (cMessageSMSIn $sms)

Dokumentacja dla tej klasy została wygenerowana z pliku:

- template_100_directReply.php

13.15 Dokumentacja klasy cSMSErrorLogger

Diagram dziedziczenia dla cSMSErrorLogger

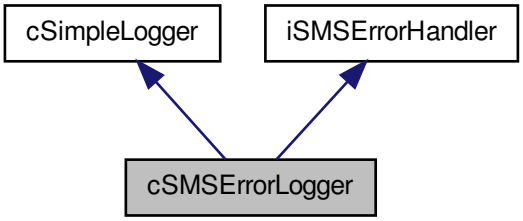
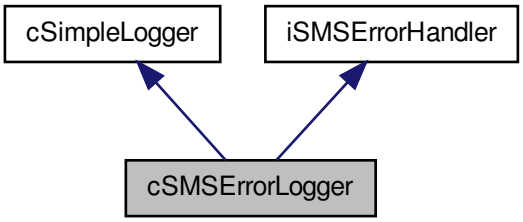


Diagram współpracy dla cSMSErrorLogger:



Metody publiczne

- **handle** (cMessageSMSIn $sms, $error)

13.15.1 Opis szczegółowy

Logger plikowy do błędów wiadomości SMS.

Zapisuje wszystkie szczegóły o błędach otrzymanych wiadomości do pliku.

Po dodaniu klasy do bramki cGateway (metodą addSMSErrorHandler) wszystkie błędnie odebrane wiadomości przez bramkę zostawią odpowiednie wpisy we wskazanym

logu.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- messageLogger.php

13.16 Dokumentacja klasy cSMSLogger

Diagram dziedziczenia dla cSMSLogger

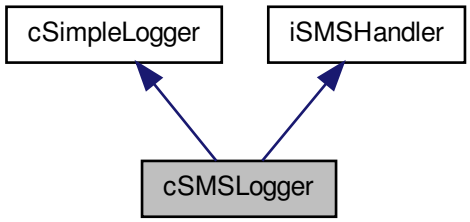
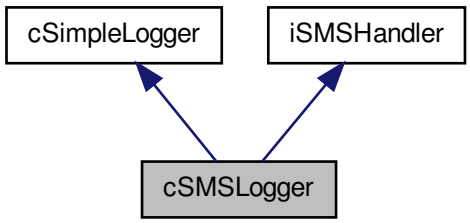


Diagram współpracy dla cSMSLogger:



Metody publiczne

- **handle** (cMessageSMSIn $sms)

13.16.1 Opis szczegółowy

Logger plikowy do wiadomości SMS.

Gotowy handler obsługi wiadomości, zapisujący szczegóły każdej otrzymanej wiadomości do pliku.

Dodłączenie do istniejącej obsługi sprowadza się do dodania polecenia: $gateway->addSMSHandler(new cSMSLogger([opcjonalna nazwa pliku]));

Zapisuje wszystkie szczegóły otrzymanych wiadomości do pliku. Klasa wykorzystywana w celach diagnostycznych.

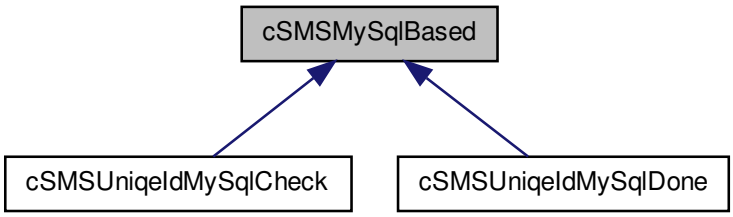
Po dodaniu klasy do bramki cGateway (metodą addSMSHandler) wszystkie wiadomości odebrane przez bramkę zostawią odpowiednie wpisy we wskazanym logu.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- messageLogger.php

13.17 Dokumentacja klasy cSMSMySqlBased

Diagram dziedziczenia dla cSMSMySqlBased



Metody publiczne

- __construct (cMySqlConnection $con, $table='sms_unique')
- **createTables** ()

Pola danych

- **$connection**

Atrybuty chronione

- **$table**

13.17.1 Dokumentacja konstruktora i destruktora

13.17.1.1 __construct (cMySqlConnection $ *con*, $ *table* = 'sms_unique')

Uruchomienie handlera z obsługą bazy danych

Parametry

cMySql-Connection	*$con*	- połączenie do bazy danych
	$table	- nazwa tabeli

Dokumentacja dla tej klasy została wygenerowana z pliku:

- messageSqlBased.php

13.18 Dokumentacja klasy cSMSUniqeIdMySqlCheck

Diagram dziedziczenia dla cSMSUniqeIdMySqlCheck

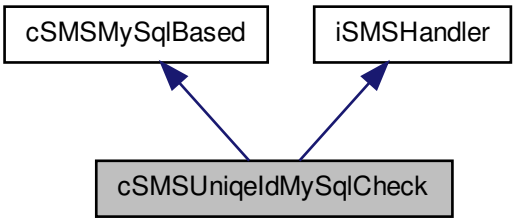
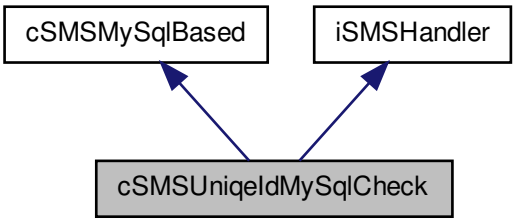


Diagram współpracy dla cSMSUniqeIdMySqlCheck:



Metody publiczne

- **handle** (cMessageSMSIn $sms)

13.18.1 Opis szczegółowy

Handler wiadomości SMS zapewniający poprawną obsługę błędów. Sprawdza czy wiadomość została wcześniej przysłana, czy proces obsługi został zakończony poprawnie oraz czy nadal trwa. Zapewnia zgodną z protokołem obsługę wielokrotnie przesyłanych wiadomości.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- messageUnique.php

13.19 Dokumentacja klasy cSMSUniqeIdMySqlDone

Diagram dziedziczenia dla cSMSUniqeIdMySqlDone

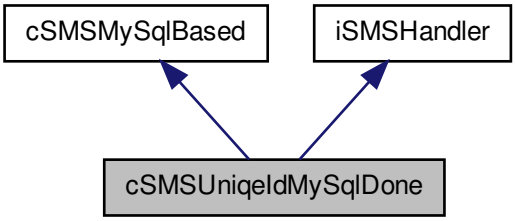
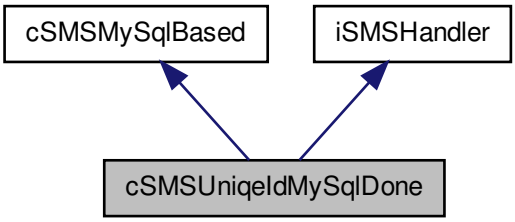


Diagram współpracy dla cSMSUniqeIdMySqlDone:



Metody publiczne

- **handle** (cMessageSMSIn $sms)

13.19.1 Opis szczegółowy

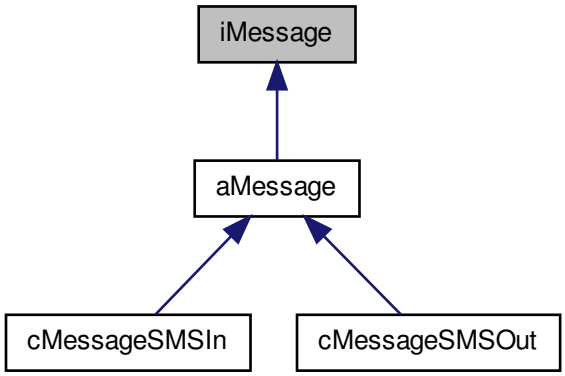
Handler wiadomości SMS obsługi błędów, zaznaczający zakończenie prac nad wiadomością. Powinien być dodany do bramki jako ostatni. Jego wywołanie zaznaczy wiadomość jako obsłużoną.

Dokumentacja dla tej klasy została wygenerowana z pliku:

- messageUnique.php

13.20 Dokumentacja interfejsu iMessage

Diagram dziedziczenia dla iMessage

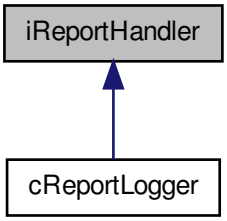


Dokumentacja dla tego interfejsu została wygenerowana z pliku:

- cashbillSms.php

13.21 Dokumentacja interfejsu iReportHandler

Diagram dziedziczenia dla iReportHandler



Metody publiczne

- **handle** (cMessageReport $message)

13.21.1 Opis szczegółowy

Inteface obsłużenia raportu dostarczenia wiadomości SMS.

Jeżeli usługa została skonfigurowania tak, by przesyłać szczątkowe raporty o dostarczaniu wiadomości, to do serwera SOAP klienta zostaną wysyłane informacje o statusie przetwarzania wiadomości.

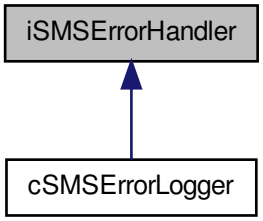
Dowolna liczba obiektów implementujących iReportHandler może zostać dodana do cGateway przez metodę addReportHandler.

Dokumentacja dla tego interfejsu została wygenerowana z pliku:

- cashbillSms.php

13.22 Dokumentacja interfejsu iSMSErrorHandler

Diagram dziedziczenia dla iSMSErrorHandler



Metody publiczne

- **handle** (cMessageSMSIn $message, $error)

13.22.1 Opis szczegółowy

Interface partnera do obsługi błędów w transmisji wiadomości SMS.

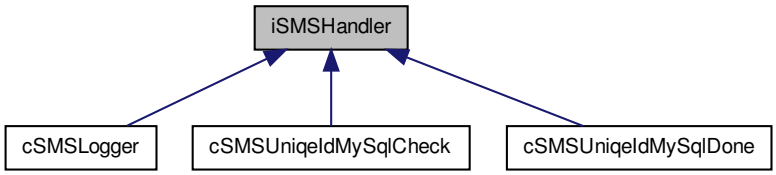
Dowolna liczba obiektów implementujących iSMSErrorHandler może zostać dodana do cGateway przez metodę addSMSErrorHandler(). Dla każdej odebranej wiadomości zostanie wywołana metoda handle.

Dokumentacja dla tego interfejsu została wygenerowana z pliku:

- cashbillSms.php

13.23 Dokumentacja interfejsu iSMSHandler

Diagram dziedziczenia dla iSMSHandler



Metody publiczne

- **handle** (cMessageSMSIn $message)

13.23.1 Opis szczegółowy

Interface partnera do obsługi wiadomości SMS.

Dowolna liczba obiektów implementujących iSMSHandler może zostać dodana do cGateway przez metodę addSMSHandler(). Dla każdej odebranej wiadomości zostanie wywołana metoda handle.

W przypadku błędu obsługi wiadomości oczekiwane jest przerwanie wykonywania handlera przez wyjątek. Komunikat wyjątku zostanie przekazany do systemu Cashbill.

Wiadomość będzie transmitowana ponownie.

W przypadku potrzeby zignorowania wiadomości (np. powtórnego przetworzenia wiadomości o tym samym ID) metoda powinna zakończyć się normalnie.

Dokumentacja dla tego interfejsu została wygenerowana z pliku:

- cashbillSms.php