# 3rd Party SMS API

## Dialog Axiata PLC

# Contents

# 1. Revision History

| Version | Date | User | Description |
|---|---|---|---|
| 1.0 | - | - | Initial Version |
| 1.1 | 2021-07-05 | Sadisha | Source address added to campaign creation request |
| 1.2 | 2021-07-08 | Sadisha | New parameter called transaction_id added to sms api. Updated the requests and responses.Added a new enpoint to check status of a transaction. |
| 1.3 | 2021-07-21 | Sadisha | Added CURL Request for every API |
| 1.4 | 2021-07-22 | Sadisha | Token expiration time is set to 12 hours. |
| 1.5 | 2021-07-28 | Vishwa | Java, PHP plugin were created. |
| 1.6 | 2022-06-23 | Sadisha | Send SMS via GET request added |
| 1.7 | 2022-06-23 | Sadisha | Updated the responses for SMS via GET request |

## 3. Access Token generation – Please note that this API should be call only on initial request and on access token expiration only

### End point: https://e-sms.dialog.lk/api/v1/login

### Method: HTTP POST

### Type: application/json

### Request Parameters

| Parameter name | Description | Mandatory/ Optional | Data type |
|---|---|---|---|
| username | Username registered with Smart Messenger | Mandatory | String |
| password | Password registered with Smart Messenger | Mandatory | String |

### JSON Object sent via Post Method

```
{
  "username": "947xxxxxxxx",
  "password": "ABC"
}
```

### cURL Code snippet

```
curl --location --request POST 'https://e-sms.dialog.lk/api/v1/login' \
--header 'Content-Type: application/json' \
--data-raw '{ "username":"XXXXXX", "password":"XXXXX" }'
```

## Response

| Parameter name | Description | Mandatory/ Optional | Data type |
|---|---|---|---|
| status | Status of the request | Mandatory | String |
| comment | Comment to find the failure reason if failed | Mandatory | String |
| accessToken | Token to be use for future communications during sending SMS and refreshing token | Mandatory | String |
| expiration | Time after which token is expired (time in seconds). **Valid only for 12 hours** | Mandatory | int |
| remainingCount | Login Count (User has 5 attempts to enter the correct password. If exceeds, account will be locked) | Mandatory | Int |
| errCode | Corresponding Error Code (Check the error code definition section) | Mandatory | Int |
| userData | Data Object<br>This is empty for a failed response | Optional | Object |
| userData -> id | User ID | Optional | Number |
| userData -> fname | User First Name | Optional | String |
| userData -> lname | User Last Name | Optional | String |
| userData -> address | User Address | Optional | Object |
| userData -> mobile | User Mobile | Optional | Number |
| userData -> email | User Email | Optional | String |
| userData -> defaultMask | User Default Mask | Optional | String |
| userData -> additional_mask | User Additional Mask List if available | Optional | Object |
| userData -> additional_mask -> mask | Additional Mask | Optional | String |
| userData -> walletBalance | User Wallet Balane | Optional | Number |

If success


```
{
    "status": "success",
    "comment": "You have logged in",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6NDgxMiwidXNlcm5hbWUiOiJzYWRpc2hh",
    "remainingCount": null,
    "expiration": 43200,
    "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpX.JpZCI6NDgxMiwidXNlcm5hbWUiOiJzYWRpc2hh",
    "refreshExpiration": 604800,
    "userData": {
        "id": <user_id>,
        "fname": <fname>,
        "lname": <lname>,
        "address": <address>,
        "mobile": <user_mobile>,
        "email": <user_email>,
        "defaultMask": <user_deafault_mask>,
        "additional_mask": [
            {
                "mask": <mask_1>
            },
            {
                "mask": <mask_2>
            }
        ],
        "walletBalance": <user_account_balance>
    },
    "errCode": ""
}
```


Note: token received in the above success response should be used as the Authorization token.

EX: "Bearer<space>" + token

### If failed

```
{
    "status": "failed",
    "comment": "<reason>",
    "token": null,
    "remainingCount": <remaining_login_attempts>,
    "expiration": null,
    "refreshToken": null,
    "refreshExpiration": null,
    "userData": null,
    "errCode": <corresponding_error_code>
}
```

# 4. SMS API

End point: https://e-sms.dialog.lk/api/v1/sms

Method: HTTP POST

Type: application/json

## Request Parameters

| Header Parameter name | Description | Mandatory/ Optional<br>If optional value should be empty | Data type |
|---|---|---|---|
| Authorization | Token received during login<br>Format as follows<br>"Bearer<space>" + accessToken | Mandatory | String |

| Parameter name in body | Description | Mandatory/ Optional | Data type |
|---|---|---|---|
| msisdn | Number list array which has 9 digit mobile numbers in the following format<br><br>Ex:<br>[<br>{<br>mobile: ”7XXXXXXXX”<br>},<br>{<br>mobile: ” 7XXXXXXXX”<br>}<br>] | Mandatory | Array |
| message | Message that should be sent to customer | Mandatory | String |
| sourceAddress | Source address or the mask which is visible to customer. | Optional | String |

| | | | |
|---|---|---|---|
| | If source Address is not (since this is optional) defined. Default mask attached with the user account will be used. Maximum length: 11 | | |
| transaction_id | Every request should have a unique integer between 1 digit to maximum of 19 digits combined. | Mandatory | Int |

## Sample JSON Object

```
{
  "msisdn":
   [
      { "mobile": "714551682" },
      { "mobile": "763625800" },
      { "mobile": "763625800" }
   ],
  "sourceAddress": "Adeona",
  "message": "This is a test message",
  "transaction_id": "<unique number for each transaction>",
}
```

## cURL Code snippet

```
curl --location --request POST 'https://e-sms.dialog.lk/api/v1/sms' \
--header 'Authorization: Bearer XXXXXXXXXXXXXX' \
--header 'Content-Type: application/json' \
--data-raw '{
    "sourceAddress": "x 514",
    "message": "test message from the sms api by X",
    "transaction_id":"125",
    "msisdn": [
```

```
        {
            "mobile": "799999999"
        },
         {
            "mobile": "799999998"
        }
    ]
}'
```

## Response

| Parameter name | Description | Mandatory/ Optional | Data type |
|---|---|---|---|
| status | Status of the request | Mandatory | String |
| comment | Comment to find the failure reason if failed | Mandatory | String |
| data | Data Object<br>This is empty for a failed response | Optional | Object |
| data -> campaignId | Created campaign id | Optional | Number |
| data -> campaignCost | Created campaign cost | Optional | Number |
| data -> walletBalance | User Wallet Balance | Optional | Number |
| data -> userMobile | User Mobile | Optional | Number |
| data -> userId | User ID | Optional | Number |
| data -> duplicatesRemoved | Duplicate numbers removed from the msisdn list | Optional | Number |
| data -> invalidNumbers | Invalid numbers removed from the msisdn list | Optional | Number |
| errCode | Error Code | Optional | String |

### If success

```
{
    "status": "success",
    "comment": "Campaign Created, Campaign ID <campaign_id>, Camp
aign payment of (LKR) <campaign_cost> was successful",
    "data": {
```

```
            "campaignId": <campaign_id>,
            "campaignCost": <campaign_cost>,
            "walletBalance": <available_wallet_balance>,
            "userMobile": <user_mobile>,
            "userId": <user_id>,
            "duplicatesRemoved": <duplicates_removed_from_the_msisdn_
    list>,
            "invalidNumbers": <invalid_numbers_removed_from_the_msisd
    n_list>
        },
     "errCode": ""

    }
```


### If failed


```
{
    "status": "failed",
    "comment": "<reason>",
    "data": "",
    "errCode": <relevant_eror_code>
}
```

# 5. Check created campaign status for a transaction id

End point: https://e-sms.dialog.lk/api/v1/sms/check-transaction

Method: HTTP POST

Type: application/json

## Request Parameters

| Header Parameter name | Description | Mandatory/ Optional<br>If optional value should be empty | Data type |
|---|---|---|---|
| Authorization | Token received during login<br>Format as follows<br>"Bearer<space>" + accessToken | Mandatory | String |

| Parameter name in body | Description | Mandatory/ Optional | Data type |
|---|---|---|---|
| transaction_id | Every request should have a unique integer between 1 digit to maximum of 19 digits combined. | Mandatory | Int |

## Sample JSON Object

```
{
  "transaction_id": "<transaction_id>",
}
```

## cURL Code snippet

```
curl --location --request POST 'https://e-
sms.dialog.lk/api/v1/sms/check-transaction' \
--header 'Authorization: Bearer XXXXXXXXX' \
--header 'Content-Type: application/json' \
--data-raw '{
    "transaction_id":"126"
}'
```

## Response

| Parameter name | Description | Mandatory/ Optional | Data type |
|---|---|---|---|
| status | Status of the request | Mandatory | String |
| comment | Comment to find the failure reason if failed | Mandatory | String |
| data | Data Object<br>This is empty for a failed response | Optional | Object |
| data -> campaign status | Status of the created campaign | Optional | String |
| errCode | Corresponding Error Code | Optional | String |
| transaction_id | Transaction ID sent by the user | Mandatory | Int |

### If success

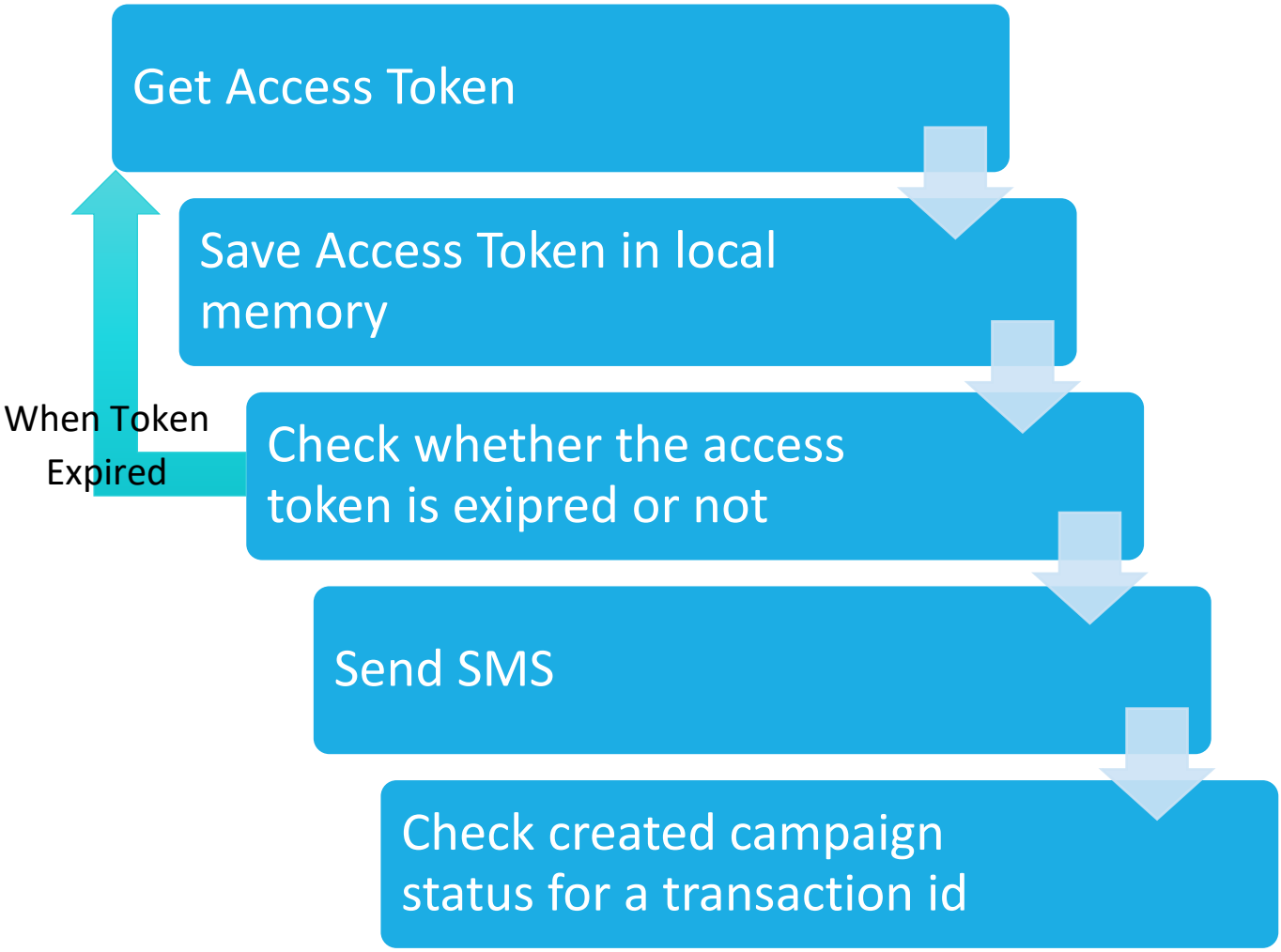
```
{
    "status": "success",
    "comment": "campaign found for the transaction id:<id> ",
    "data": {
        "campaign status": "completed"
    },
    "errCode": "",
    "transaction_id": "<transaction_id>"

}
```

### If failed

```
{
    "status": "failed",
    "comment": "<reason>",
    "data": "",
    "errCode": "<error_code>",
     "transaction_id": "<transaction_id>"
}
```

# 6. Plugins

## 6.1. Working Scenario



## 6.2. Java

### 6.2.1. Supported Projects

#### 6.2.1.1. Maven projects

#### 6.2.1.2. Spring Boot maven projects

### 6.2.2. Installation

Find sample maven project based on Java Jdk 8 to guide how to communicate with SMS API library from https://github.com/adeonatech/Java-SMS-API-lib-maven-sample and find latest version of library in Maven Central Repository when click on the maven badge in README.md. Then add the dependency to your project.

#### Apache Maven

```
<dependency>
  <groupId>net.adeonatech</groupId>
  <artifactId>SmsAPI</artifactId>
  <version>1.0.4</version>
</dependency>
```

#### Gradle Groovy DSL

```
implementation 'net.adeonatech:SmsAPI:1.0.4'
```

## 6.2.3. Quick Start

### 6.2.3.1. Get Access token

After sign up in https://esms.dialog.lk you must need to get access token using username, password:

1. **Create the object of "TokenBody" for set data to get token:**

```
TokenBody tokenBody = new TokenBody();
```

2. **Set username through object:**

```
tokenBody.setUsername(<YOUR USERNAME>);
```

3. **Set password through object:**

```
tokenBody.setPassword(<YOUR PASSWORD>);
```

4. **Create the object of "SendSMSImpl" to use method to get token :**

```
SendSMSImpl sendSMS = new SendSMSImpl();
```

5. **Get token using "getToken()" method through object:**

```
sendSMS.getToken(<new TokenBody()>){
      return <new TokenResponse()>
}
```

6. **Get "TokenResponse" type return values:**
   a. **Get token**

```
sendSMS.getToken(<new TokenBody()>).getToken();
```

   b. **Get comment**

```
sendSMS.getToken(<new TokenBody()>).getComment();
```

   c. **Get status**

```
sendSMS.getToken(<new TokenBody()>).getStatus();
```

   d. **Get remaining count**

```
sendSMS.getToken(<new TokenBody()>).getRemainingCount();
```

#### e. Get expiration

```
sendSMS.getToken(<new TokenBody()>).getExpiration();
```

#### f. Get refresh token

```
sendSMS.getToken(<new TokenBody()>).getRefreshToken();
```

#### g. Get refresh expiration

```
sendSMS.getToken(<new TokenBody()>).getRefreshExpiration();
```

#### h. Get error code

```
sendSMS.getToken(<new TokenBody()>).getErrCode();
```

#### i. Get user data

##### i. Get wallet balance

```
sendSMS.getToken(<new TokenBody()>).getUderData().getWalletBalance();
```

##### ii. Get default mask

```
sendSMS.getToken(<new TokenBody()>).getUderData().getDefaultMask();
```

##### iii. Get email

```
sendSMS.getToken(<new TokenBody()>).getUderData().getEmail();
```

##### iv. Get mobile

```
sendSMS.getToken(<new TokenBody()>).getUderData().getMobile();
```

##### v. Get address

```
sendSMS.getToken(<new TokenBody()>).getUderData().getAddress();
```

##### vi. Get last name

```
sendSMS.getToken(<new TokenBody()>).getUderData().getLname();
```

##### vii. Get first name

```
sendSMS.getToken(<new TokenBody()>).getUderData().getFname();
```

##### viii. Get id

```
sendSMS.getToken(<new TokenBody()>).getUderData().getId();
```

##### ix. Get additional masks

```
sendSMS.getToken(<new
TokenBody()>).getUderData().getAdditional_mask().get(0).getMask();
```

example:

```
TokenBody tokenBody = new TokenBody();

tokenBody.setUsername("nimal");
tokenBody.setPassword("Admin#67!");

SendSMSImpl sendSMS = new SendSMSImpl();

String token = sendSMS.getToken(tokenBody).getToken();
```

### 6.2.4. Save Access token in local memory

After getting the access token you should save it in the local memory. When call for every other methods you should set the saved access token as a parameter to get success response.

### 6.2.5. Check whether the access token expired or not

When access token expired the response by **sendSMS.getToken(<new TokenBody()>).getComment()** will be "**Authentication Token Expired**". Then again get new token and save it to the local memory and continue your process.

### 6.2.6. Send SMS

1. **Create the object of "SendTextBody" for set data to get token:**

```
SendTextBody sendTextBody = new SendTextBody();
```

2. **Set source address through object:**

```
sendTextBody.setSourceAddress(<YOUR SOURCE ADDRESS>);
```

3. **Set message through object:**

```
sendTextBody.setMessage(<YOUR MESSAGE>);
```

4. **Set transaction id through object:**

```
sendTextBody.setTransaction_id(<YOUR TRANSACTION ID>);
```

5. **Set msisdns through creating the object of "SendSMSImpl":**

```
SendSMSImpl sendSMS = new SendSMSImpl();
sendTextBody.setMsisdn(sendSMS.setMsisdns(new String[]{"<MOBILE 1>","< MOBILE 2>"}));
```

6. **Send SMS using "sendText()" method through created object of "SendSMSImpl":**

```
sendSMS.sendText(<new SendTextBody()>, <String token>){
      return <new SendTextResponse()>
}
```

7. **Get "SendTextResponse" type return values:**
    a. **Get comment**

```
sendSMS.sendText(<new SendTextBody()>, <String token>).getComment();
```

    b. **Get status**

```
sendSMS.sendText(<new SendTextBody()>, <String token>).getStatus();
```

    c. **Get error code**

```
sendSMS.sendText(<new SendTextBody()>, <String token>).getErrCode();
```

    d. **Get data**
        i. **Get invalid numbers**

```
sendSMS.sendText(<new SendTextBody()>, <String token>).getData().getInvalidNumbers();
```

        ii. **Get default mask**

```
sendSMS.sendText(<new SendTextBody()>, <String
token>).getData().getInvalidNumbers();
```

### iii. Get duplicates removed

```
sendSMS.sendText(<new SendTextBody()>, <String
token>).getData().getDuplicatesRemoved();
```

### iv. Get user id

```
sendSMS.sendText(<new SendTextBody()>, <String token>).getData().getUserId();
```

### v. Get user mobile

```
sendSMS.sendText(<new SendTextBody()>, <String token>).getData().getUserMobile();
```

### vi. Get wallet balance

```
sendSMS.sendText(<new SendTextBody()>, <String token>).getData().getWalletBalance();
```

### vii. Get campaign cost

```
sendSMS.sendText(<new SendTextBody()>, <String token>).getData().getCampaignCost();
```

### viii. Get campaign id

```
sendSMS.sendText(<new SendTextBody()>, <String token>).getData().getCampaignId();
```

example:

```
SendSMSImpl sendSMS = new SendSMSImpl();

SendTextBody sendTextBody = new SendTextBody();

sendTextBody.setMsisdn(sendSMS.setMsisdns(new String[]{"71XXXXXXX",
"71XXXXXXX"}));
sendTextBody.setSourceAddress("sourse1");
sendTextBody.setMessage("Hi! this is test from JAVA lib");
sendTextBody.setTransaction_id("144");

TransactionBody transactionBody = new TransactionBody();
transactionBody.setTransaction_id("144");

String response = sendSMS.sendText(sendTextBody,
sendSMS.getToken(tokenBody).getToken()).getStatus();
```

### 6.2.7. Check created campaign status for a transaction id

1. **Create the object of "TransactionBody" for set data to get token:**

```
TransactionBody transactionBody = new TransactionBody();
```

2. **Set transaction id through object:**

```
transactionBody.setTransaction_id(<YOUR TRANSACTION ID>);
```

3. **Create the object of "SendSMSImpl":**

```
SendSMSImpl sendSMS = new SendSMSImpl();
```

4. **Get campaign status for the transaction id using "getTransactionIDStatus()" method through created object of "SendSMSImpl":**

```
sendSMS.getTransactionIDStatus(<new TransactionBody()>, <String token>){
      return <new TransactionResponse()>
}
```

5. **Get "TransactionResponse" type return values:**
    a. **Get comment**

```
sendSMS.getTransactionIDStatus(<new TransactionBody()>, <String token>).getComment();
```

    b. **Get status**

```
sendSMS.getTransactionIDStatus(<new TransactionBody()>, <String token>).getStatus();
```

    c. **Get error code**

```
sendSMS.getTransactionIDStatus(<new TransactionBody()>, <String token>).getErrCode();
```

    d. **Get transaction id**

```
sendSMS.getTransactionIDStatus(<new TransactionBody()>, <String
token>).getTransaction_id();
```

    e. **Get data**
        i. **Get campaign status**

```
sendSMS.getTransactionIDStatus(<new TransactionBody()>, <String
token>).getDataTransaction().getCampaign_status();
```

example:

```
SendSMSImpl sendSMS = new SendSMSImpl();

TransactionBody transactionBody = new TransactionBody();
transactionBody.setTransaction_id("144");

String campaignStatus =
sendSMS.getTransactionIDStatus(transactionBody,sendSMS.getToken(tokenBody).ge
tToken()).getDataTransaction().getCampaign_status();
```

## 6.3. PHP

### 6.3.1. Installation

Find library source code from https://github.com/adeonatech/PHP-SMS-API-lib and download it to your selected path. Then call

```
<?php

    require('<YOUR LIBRARY PATH>/PHP-SMS-API-lib-main/send_sms_impl.php');

?>
```

inside your PHP code.

## 6.3.2. Quick Start

### 6.3.2.1. Get Access token

After sign up in https://esms.dialog.lk you must need to get access token using username, password:

1. **Create the object of "SendSMSImpl" to use method to get token:**

```
$sendSmsImpl = new SendSMSImpl();
```

2. **Set username through object:**

```
$tokenBody->setUsername(<YOUR USERNAME>);
```

3. **Set password through object:**

```
$tokenBody->setPassword(<YOUR PASSWORD>);
```

4. **Create the object of "TokenBody" for set data to get token:**

```
$tokenBody = new TokenBody();
```

5. **Get token using "getToken()" method through object:**

```
$sendSmsImpl->getToken(<new TokenBody()>){
      return <new TokenResponse()>
}
```

6. **Get "TokenResponse" type return values:**
   a. **Get token**

```
$sendSmsImpl->getToken((<new TokenBody()>)->getToken();
```

   b. **Get comment**

```
$sendSmsImpl->getToken((<new TokenBody()>)->getComment();
```

   c. **Get status**

```
$sendSmsImpl->getToken((<new TokenBody()>)->getStatus();
```

   d. **Get remaining count**

```
$sendSmsImpl->getToken((<new TokenBody()>)->getRemainingCount();
```

   e. **Get expiration**

```
$sendSmsImpl->getToken((<new TokenBody()>)->getExpiration();
```

#### f. Get refresh token

```
$sendSmsImpl->getToken((<new TokenBody()>)->getRefreshToken();
```

#### g. Get refresh expiration

```
$sendSmsImpl->getToken((<new TokenBody()>)->getRefreshExpiration();
```

#### h. Get error code

```
$sendSmsImpl->getToken((<new TokenBody()>)->getErrCode();
```

#### i. Get user data

##### i. Get wallet balance

```
$sendSmsImpl->getToken((<new TokenBody()>)->getUderData()->getWalletBalance();
```

##### ii. Get default mask

```
$sendSmsImpl->getToken((<new TokenBody()>)->getUderData()->getDefaultMask();
```

##### iii. Get email

```
$sendSmsImpl->getToken((<new TokenBody()>)->getUderData()->getEmail();
```

##### iv. Get mobile

```
$sendSmsImpl->getToken((<new TokenBody()>)->getUderData()->getMobile();
```

##### v. Get address

```
$sendSmsImpl->getToken((<new TokenBody()>)->getUderData()->getAddress();
```

##### vi. Get last name

```
$sendSmsImpl->getToken((<new TokenBody()>)->getUderData()->getLname();
```

##### vii. Get first name

```
$sendSmsImpl->getToken((<new TokenBody()>)->getUderData()->getFname();
```

##### viii. Get id

```
$sendSmsImpl->getToken((<new TokenBody()>)->getUderData()->getId();
```

##### ix. Get additional masks

```
$sendSmsImpl->getToken((<new TokenBody()>)->getUserData()-
>getAdditionalMask()[0]['mask']
```

example:

```
$sendSmsImpl = new SendSMSImpl();

$tokenBody = new TokenBody();

$tokenBody->setUsername("nimal”);
$tokenBody->setPassword("Admin#67!");

$sendTextBody = new SendTextBody();

$token = $sendSmsImpl->getToken($tokenBody)->getToken();
```

### 6.3.3. Save Access token in local memory

After getting the access token you should save it in the local memory. When call for every other methods you should set the saved access token as a parameter to get success response.

### 6.3.4. Check whether the access token expired or not

When access token expired the response by

**$sendSmsImpl->getToken((<new TokenBody()>)->getComment()**

will be “**Authentication Token Expired**”. Then again get new token and save it to the local memory and continue your process.

### 6.3.5. Send SMS

1. **Create the object of "SensSMSImpl" to send SMS:**

```
$sendSmsImpl = new SendSMSImpl();
```

2. **Create the object of "SendTextBody" for set data to get token:**

```
$sendTextBody = new SendTextBody();
```

3. **Set source address through object:**

```
$sendTextBody->setSourceAddress(<YOUR SOURCE ADDRESS>);
```

4. **Set message through object:**

```
$sendTextBody->setMessage(<YOUR MESSAGE>);
```

5. **Set transaction id through object:**

```
$sendTextBody->setTransactionId(<YOUR TRANSACTION ID>);
```

6. **Set msisdns through the created object of "SendSMSImpl":**

```
$sendTextBody->setMsisdn($sendSmsImpl->setMsisdns(array("<MOBILE 1>","<MOBILE 2>")));
```

7. **Send SMS using "sendText()" method through created object of "SendSMSImpl":**

```
$sendSmsImpl->sendText(<new SendTextBody>,$token){
        return <new SendTextResponse>
}
```

8. **Get "SendTextResponse" type return values:**

   a. **Get comment**

```
$sendSmsImpl->sendText(<new SendTextBody>,$token)->getComment();
```

   b. **Get status**

```
$sendSmsImpl->sendText(<new SendTextBody>,$token)->getStatus();
```

   c. **Get error code**

```
$sendSmsImpl->sendText(<new SendTextBody>,$token)->getErrCode();
```

### d. Get data

#### i. Get invalid numbers

```
$sendSmsImpl->sendText(<new SendTextBody>,$token)->getData()->getInvalidNumbers();
```

#### ii. Get default mask

```
$sendSmsImpl->sendText(<new SendTextBody>,$token)->getData()->getInvalidNumbers();
```

#### iii. Get duplicates removed

```
$sendSmsImpl->sendText(<new SendTextBody>,$token)->getData()->getDuplicatesRemoved();
```

#### iv. Get user id

```
$sendSmsImpl->sendText(<new SendTextBody>,$token)->getData()->getUserId();
```

#### v. Get user mobile

```
$sendSmsImpl->sendText(<new SendTextBody>,$token)->getData()->getUserMobile();
```

#### vi. Get wallet balance

```
$sendSmsImpl->sendText(<new SendTextBody>,$token)->getData()->getWalletBalance();
```

#### vii. Get campaign cost

```
$sendSmsImpl->sendText(<new SendTextBody>,$token)->getData()->getCampaignCost();
```

#### viii. Get campaign id

```
$sendSmsImpl->sendText(<new SendTextBody>,$token)->getData()->getCampaignId();
```

example:

```
$sendSmsImpl = new SendSMSImpl();

$sendTextBody = new SendTextBody();

$sendTextBody->setSourceAddress("source1");
$sendTextBody->setMsisdn($sendSmsImpl-
>setMsisdns(array("71XXXXXXX","71XXXXXXX")));
$sendTextBody->setTransactionId("146");
$sendTextBody->setMessage("Hi this is test from PHP");

$transactionBody = new TransactionBody();
$transactionBody->setTransactionId("146");

$response = $sendSmsImpl->sendText($sendTextBody, $sendSmsImpl-
>getToken($tokenBody)->getToken())->getData()->getUserId();
```

### 6.3.6. Check created campaign status for a transaction id

1. **Create the object of "SendSMSImpl":**

```
$sendSmsImpl = new SendSMSImpl();
```

2. **Create the object of "TransactionBody" for set data to get token**

```
$transactionBody = new TransactionBody();
```

3. **Set transaction id through object:**

```
$transactionBody->setTransactionId (<YOUR TRANSACTION ID>);
```

4. **Get campaign status for the transaction id using "getTransactionIDStatus()" method through created object of "SendSMSImpl":**

```
$sendSmsImpl->getTransactionIDStatus (<new TransactionBody>, <$token>){
      return <new TransactionResponse>
}
```

5. **Get "TransactionResponse" type return values:**
    a. **Get comment**

```
$sendSmsImpl->getTransactionIDStatus (<new TransactionBody>, <$token>)->getComment();
```

    b. **Get status**

```
$sendSmsImpl->getTransactionIDStatus (<new TransactionBody>, <$token>)->getStatus();
```

    c. **Get error code**

```
$sendSmsImpl->getTransactionIDStatus (<new TransactionBody>, <$token>)->getErrCode();
```

    d. **Get transaction id**

```
$sendSmsImpl->getTransactionIDStatus (<new TransactionBody>, <$token>)-
>getTransactionId();
```

    e. **Get data**
        i. **Get campaign status**

```
$sendSmsImpl->getTransactionIDStatus (<new TransactionBody>, <$token>)->getData()-
>getCampaignStatus();
```

example:

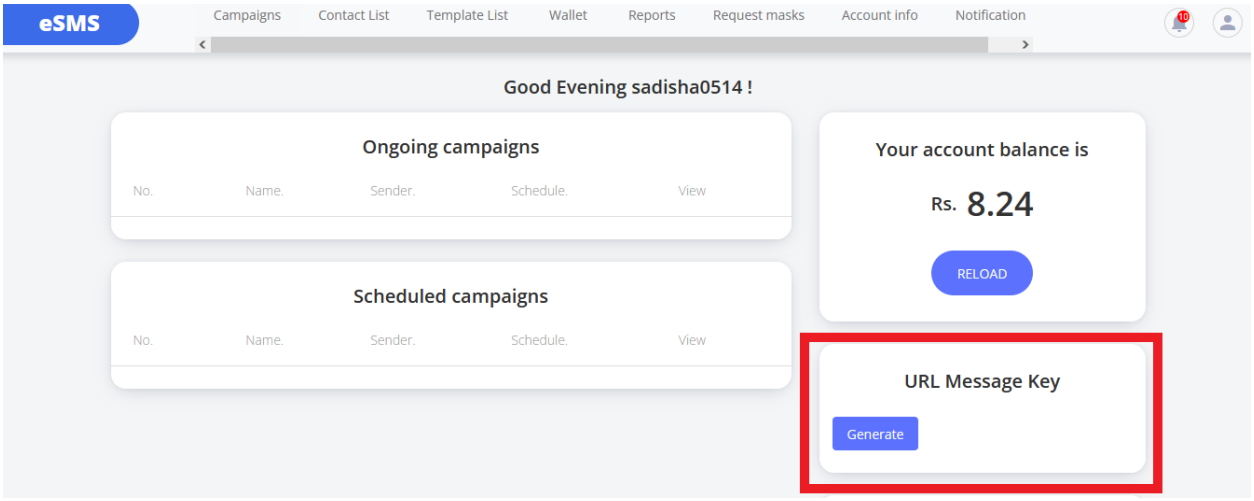
```
$sendSmsImpl = new SendSMSImpl();

$transactionBody = new TransactionBody();
$transactionBody->setTransactionId("146");

$campaignStatus = $sendSmsImpl-
>getTransactionIDStatus($transactionBody, $sendSmsImpl-
>getToken($tokenBody)->getToken())->getData()->getCampaignStatus();
```

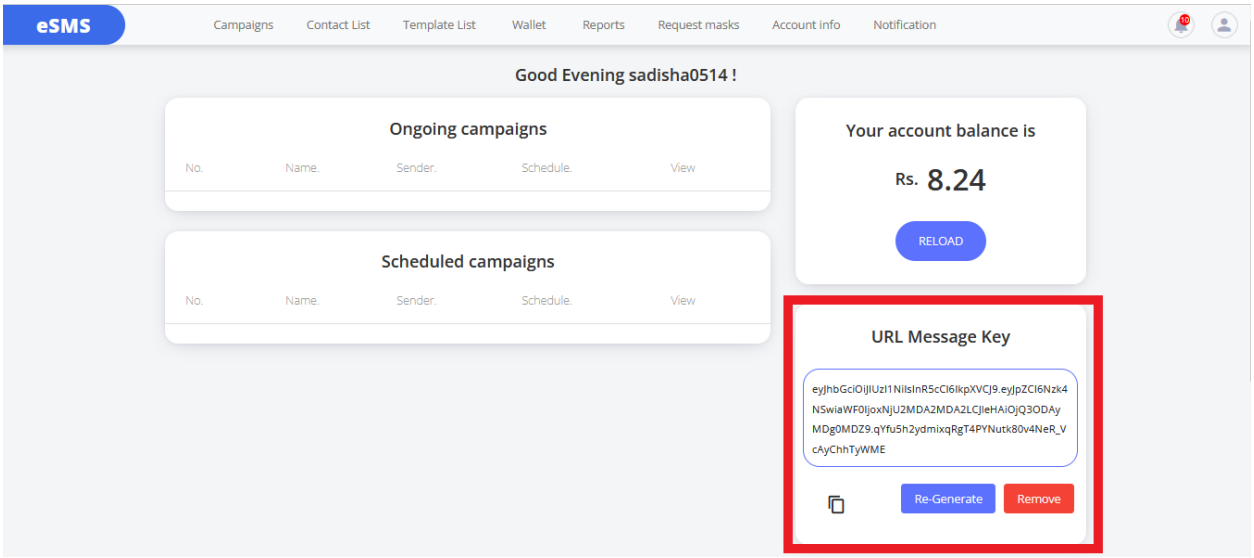
# 7. Send SMS via GET Request

**Prerequisite: ESMS admin has to provide the ability to use this functionality. By default, it is not available.**

## How to generate esmsqk ? (Key to send SMS via GET)

**Once you log into the eSMS portal, you can see a section which is named as URL Message Key**



## Why can't I see URL Message Key section on my dashboard?

- Try clearing your browser cache and login again.
- If the issue still exists, you don't have the necessary access to consume this resource.

End point: https://e-sms.dialog.lk/api/v1/message-via-url/create/url-campaign?esmsqk=<key> &list=<number_list>&source_address=<mask> &message=<message>

Method: HTTP GET

## Request Params

| Parameter name | Description | Mandatory/ Optional<br>If optional value should be empty | Data type |
|---|---|---|---|
| esmsqk | Client key to send messages via GET requests. This can be generated via eSMS user portal. eSMS user account can only have one key. (If a new key is generated, the old key will be invalid) | Mandatory | String |
| list | Comma separated mobile number list. Each mobile number has to be either of the following formats. | Mandatory | String |

| | | | |
|---|---|---|---|
| | i.799999999 (9 digits)<br>ii.0799999999(10 digits)<br>iii. 94799999999<br><br>(11digits) | | |
| source_address | Source address or the mask which is visible to customer. | Optional (If not defined, default mask of the user will be used) | String |
| message | Message that should be sent to customer | Mandatory | String |

## Sample Request

https://e-sms.dialog.lk/api/v1message-via-url/create/url-campaign?esmsqk=eyJhbGciOiJIUzI1NiIsInR5c&list=0799999999,799999999,94799999999&source_address=test&message=Welcome

## cURL Code snippet

```
curl --location --request GET 'http://localhost:3000/api/v1/message-via-url/create/url-campaign?esmsqk=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTUsImlhdCI6MTY1NTM1MDkyNCwiZXhwIjo0Nzc5NTUzMzI0fQ.EqylAmt1_CWOEpcrQA--kRiW-qFHNktMTz6Y1YDA1f4&list=0714551682,763625800,94777337045&source_address=adeona&message=Welcome' \
--data-raw ''
```

## Response

| Parameter name | Description | Mandatory/ Optional | Data type |
|---|---|---|---|
| <integer_value> | Integer value (Check the response id table) | Mandatory | Integer |

### If success

```
1
```

### If failed

```
<error_id>
```

| Response id | Description |
|---|---|
| 1 | Success |
| 2001 | Error occurred during campaign creation |
| 2002 | Bad request |
| 2003 | Empty number list |
| 2004 | Empty message body |
| 2005 | Invalid number list format |
| 2006 | Not eligible to send messages via get requests (Admin haven't provided the access level) |
| 2007 | Invalid key (esmsqk parameter is invalid) |
| 2008 | Not enough money in the user wallet |

Above response id(s) only valid for send SMS via get requests (section 7).

## 8. Error Codes

Below error codes are not applicable to section 7(Send SMS via GET Request) of this document

| Error Code | Description |
|---|---|
| 100 | Invalid Token |
| 101 | Invalid Request Parameters |
| 102 | User account not found or not a valid account |
| 103 | Logical Error |
| 104 | Transaction ID is already used |
| 999 | Internal Server Error |

Powered by Dialog Axiata PLC
For more information,
please contact
Kasun Eranda – 0777 03 62 62