E-SMS API DOCUMENTATION

Table of 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
1.8	2022-07-27	Sadisha	• Add a new response status for SMS via GET request. • Added a new GET request to view account balance for users who are consuming SMS via GET request.

1.9	2022-08-19	Sadisha	New parameter to specify payment method added to the endpoint corresponding to sending SMS via GET request. New error codes added for section 9 of this document.
2.0	2022-08-22	Sadisha	Endpoints added to send SMS and check created campaign status for a transaction id (Version 2)
2.1	2022-10-07	Sadisha	New push notification parameter enabled to push delivery notifications for SMS APIs
2.2	2022-11-04	Roshan	SMS API v1 was depreciated. Error code 2012 was added related to get request SMS.
2.3	2023-05-15	Roshan	Update the length of E-SMS API transaction id.
2.4	2023-06-16	Sadisha	A new error codes were introduced in SMS API version 2 and send SMS via GET requests to indicate if too many requests were detected from the user.
2.5	2023-06-22	Roshan	E-SMS API request limitations were defined.
2.6	2023-08-09	Roshan	The token generation SMS API endpoint was updated.
2.7	2024-07-12	Savodya	Updated the document format and added the FAQ section.
2.8	2024-08-30	Savodya	Access token endpoint was updated.

2. Registration

Use the eSMS widget available in the Dialog Marketplace to consume the esms API. Visit https://e-sms.dialog.lk to get started.

3. SMS API

3.1. SMS Via POST Request

3.1.1. Access Token Generation

Please note that this API should be call only on initial request and on access token expiration only

End point: https://esms.dialog.lk/api/v2/user/login

Method: HTTP POST

Type: application/json

Request Parameters

Parameter name	Description	Mandatory/ Optional	Data type
username	Username registered with E-SMS.	Mandatory	String
password	Password registered with E-SMS.	Mandatory	String

JSON Object Sent Via Post Method

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

cURL Code Snippet

```
curl --location --request POST 'https://esms.dialog.lk/api/v2/user/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
token	Token to be use for future communications during sending SMS.	Mandatory	String
expiration	Time after which token is expired (time in seconds).	Mandatory	Integer
remainingCount	Login Count (User has 5 attempts to enter the correct password. If exceeds, account will be locked)	Mandatory	Integer
refreshToken	Plan for future use	Optional	
refreshExpiration	Time after which token is expired (time in seconds).	Optional	
userData	Data Object This is empty for a failed response	Optional	JASON Object
userData -> id	User ID	Optional	Integer
userData -> fname	User First Name	Optional	String
userData -> lname	User Last Name	Optional	String

userData -> address	User Address	Optional	JASON Object
userData -> mobile	User Mobile	Optional	Integer
userData -> email	User Email	Optional	String
userData -> defaultMask	User Default Mask	Optional	String
userData -> additional_mask	User Additional Mask List if available	Optional	JASON Array
userData -> additional_mask -> mask	Additional Mask	Optional	String
userData -> walletBalance	User Wallet Balance	Optional	Integer
accountType	Type of the account	Optional	Integer
accountLocked	Indicate the account lock status	Optional	Integer
accountStatus	Status whether active or not active	Optional	Integer
walletBalance	Available balance of the account.	Optional	Float
additional_addons	Addons that activated for the user account.	Optional	String
data	Future use	Optional	String
errCode	Corresponding Error Code (Check the error code definition section)	Mandatory	Integer

If Success


```
{
   "status": "success",
   "comment": "You have logged in",
   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6NDgxMiwidXNlcm5hbW
            UiOiJzYWRpc2hh",
   "remainingCount": null,
   "expiration": 43200,
   "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpX.JpZCI6NDgxMiwidXNlcm5hb
                  WUiOiJzYWRpc2hh",
   "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>
         }
      ],
          "accountType": <account type>,
      "agreeToTerms": <agreement>,
```



```
    "accountLocked": <account lock status>,
    "accountStatus": <account status active/inactive>,
    "campaignTwoPhaseLock": 0,
    "walletBalance": <user_account_balance>,
    "additional_addons": <additional addons >
  },
   "data": "",
  "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,
   "data": "",
   "errCode": <corresponding error code>
}
```


3.1.2. Send SMS

End point: https://e-sms.dialog.lk/api/v2/sms

Method: HTTP POST

Type: application/json

Request Parameters

Header Parameter name	Description	Mandatory/ Optional If optional value should be empty	Data type
Authorization	Token received during login Format as follows "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
	
	Ex:
	[
	{
	mobile: ”7XXXXXXXX”
	},
	{
	mobile: ” 7XXXXXXXX”
	}
	]
		Mandatory	JASON Array

message	Message that should be sent to customer	Mandatory	String
sourceAddress	Source address or the mask which is visible to customer. If source Address is not (since this is optional) defined. Default mask attached with the user account will be used. Maximum length: 11	Optional	String
transaction_id	Every request should have a unique integer between 1 digit to maximum of 18 digits combined.	Mandatory	Integer

payment_method	0-pay using my esms wallet, 4-pay using my esms API package (Only available for package users) If you are a new user,set this value to 0. This parameter is an optional parameter where is case not found, we assume it is 0. 0 is for a wallet payment	Optional	Integer (0 Or 4)
push_notification_url	Endpoint to which delivery notifications are pushed. (Delivery reports are only pushed if this parameter is defined)	Optional	String

Sample JSON Object

```
{
 "msisdn":
  [
    { "mobile": "714551682" },
    { "mobile": "763625800" },
    { "mobile": "763625800" }
  ],
 "sourceAddress": "<Mask>",
 "message": "<SMS Content>",
 "transaction_id": <unique number for each transaction>,
  "payment_method": <payment_method>,
  "push_notification_url": "https://xxx/xx?",
                }
```


cURL Code

snippet

```
curl --location --request POST ' https://e-sms.dialog.lk /api/v2/sms' \
--header 'Authorization: Bearer XXXXXXXXXXXXXX' \
--header 'Content-Type: application/json' \
--data-raw '{
   "sourceAddress": "test_mask",
   "message": "test message",
   "transaction_id": 1001,
   "payment_method": 4,
   "msisdn": [
    {
        "mobile": "799999999"
    },
        {
        "mobile": "799999998"
    }

   ],
   "push_notification_url": " https://xxx/xx?",
   }'
```


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 This is empty for a failed response	Optional	JASON Object
data -> campaignId	Created campaign id	Optional	Integer
data -> campaignCost	Created campaign cost	Optional	Integer
data -> walletBalance	User Wallet Balance	Optional	Integer
data -> userMobile	User Mobile	Optional	Integer
data -> userId	User ID	Optional	Integer
data -> duplicatesRemoved	Duplicate numbers removed from the msisdn list	Optional	Integer
data -> invalidNumbers	Invalid numbers removed from the msisdn list	Optional	Integer
data -> mask_blocked_numbers	Mobile numbers who blocked messages form the specified mask	Optional	Integer
errCode	Error Code	Mandatory	String

If success


```
{
   "status": "success",
    "comment": "Campaign Created, Campaign ID <campaign_id>, Campaign 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>,
      "mask_blocked_numbers":<mobiles_who_blocked_the_mask>,"userId": <user_id>,
    },
          "errCode": ""

}
```


If failed


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


Note: SMS API has demonstrated its reliability and efficiency within the tested limit of 1000 recipient numbers. While the API is designed to handle high message volumes, surpassing this limit may have adverse effects on its performance for the particular user. We strongly advise against exceeding the recommended limit to prevent potential disruptions in service, delayed message delivery, or degraded performance.

3.1.3. Check Created Campaign Status for a Transaction Id

Version 2 is the updated version for checking campaign status using transaction id. This version is optimized compared to its predecessor.

End point: https://e-sms.dialog.lk/api/v2/sms/check-transaction

Method: HTTP POST

Type: application/json

Request Parameters

Header Parameter name	Description	Mandatory/ Optional If optional value should be empty	Data type
Authorization	Token received during login Format as follows "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 18 digits combined.	Mandatory	Integer

Sample JSON Object

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

cURL Code Snippet

```
curl --location --request POST 'https://e-sms.dialog.lk/api/v2/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 This is empty for a failed response	Optional	JASON Object
data -> campaign status	Status of the created campaign pending/completed/running	Optional	String
errCode	Corresponding Error Code	Mandatory	String
transaction_id	Transaction ID sent by the user	Mandatory	Integer

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>
}
```


Note: Maximum campaign status check API requests count per minute is 120.

3.1.4. Delivery Report API

If 3rd party passes “push_notification_url” in the API request, then eSMS platform will send delivery notifications as GET request.

Following is the format that eSMS system is calling 3rd party platforms. Therefore, it is mandatory to implement delivery report end point as per below.

Method: HTTP GET

Request format: <end point passed in “push_notification_url”>campaignId=<campaign ID returned in the response of campaign creation API>&msisdn=<MSISDN/ mobile number>&status=<status> Sample request

https://adeonatech.net/api/v1/delivery-report-listner?campaignId=25&msisdn=94777888665&status=1

Possible values for status

Status	Description
1	Successfully submitted to SMSC
2	SMS submission failed due to number not valid Or connectivity failure
3	Successfully delivered
4	Delivery failed

For delivery success scenario eSMS will send status code 1 and 3. Order of the status code submission is not guaranteed. Therefore, its 3rd parties responsibility to develop the application processing accordingly.

3.1.5. Error Codes – POST Request

Below error codes are not applicable to section 3.2(Send SMS via GET Request) and section 3.2.2(Check account balance via GET Request) of this document.

Error Code	Description
100	Invalid Token (Token Expired)
101	Invalid Request Parameters
102	User account not found or not a valid account
103	Unable to find a campaign for the specified transaction ID.
104	Transaction ID is already used
105	Invalid Token Signature
106	Token not found in the header (Or token is not attached as a bearer token)
107	One or more mandatory parameters in the request in either missing or invalid

108	User don’t have such active mask eligible to send messages
109	There is no single valid mobile number after removing invalids, duplicates and mask blocked numbers from MSISDN list.
110	Not eligible to consume packaging
111	Package payments can only be used for campaigns scheduled for this month.
112	Number of messages left in the package is less than the campaign messages.
113	Package Maintenance Downtime.
114	Not enough wallet balance to run the campaign.
115	Username or password invalid
116	Account locked
117	Too many requests
999	Internal Server Error

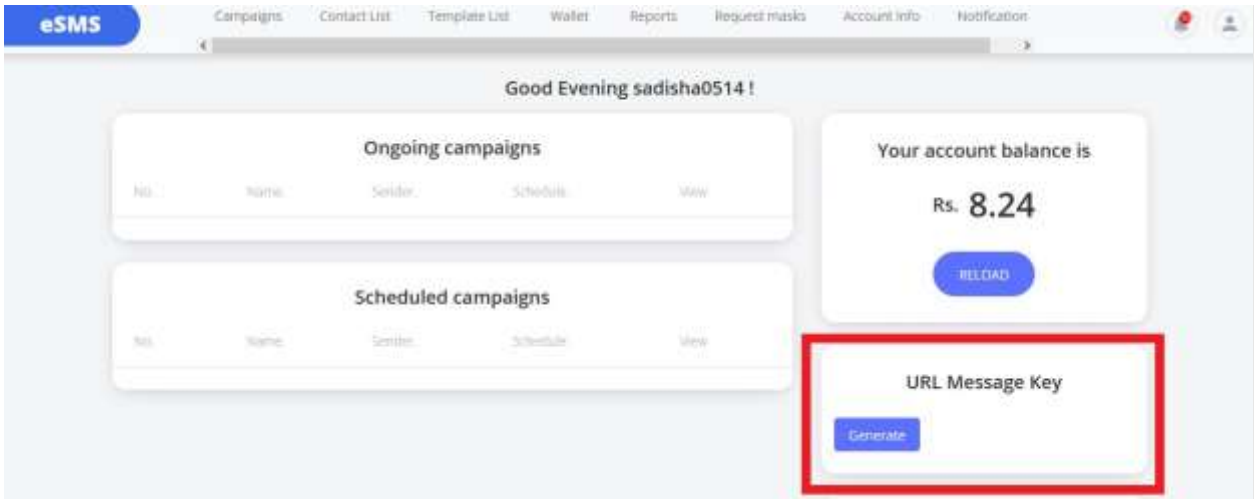
3.2. SMS via GET Request

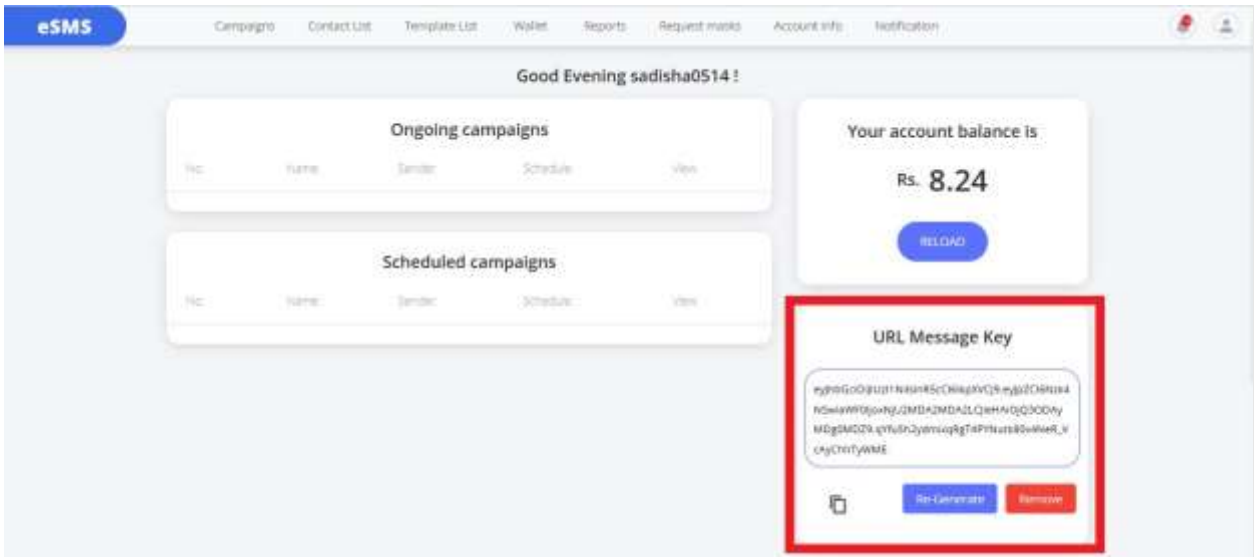
3.2.1. Send SMS

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





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 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. i.799999999 (9 digits) ii.0799999999(10 digits) iii. 94799999999 (11digits)	Mandatory	String
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

paymentType	Payment method for the campaign. As for now, only via package (4) and via wallet (0) are the available payment methods. Note: 0: payment via wallet 4: payment via package If paymentType is not defined or empty, it is considered as a payment made via wallet.	Optional	Integer
push_notification_url	Endpoint to which delivery notifications are pushed. (Delivery reports are only pushed if this parameter is defined)	Optional	String

Sample Request

https://e-sms.dialog.lk/api/v1/message-via-url/create/url-campaign?esmsqk=eyJhbGciOiJIUzI1NiIsInR5c&list=0799999999,

799999999,94799999999&source_address=test&message=Welcome&push_notification_url=https://xx/xx'

cURL Code snippet

curl --location --request GET 'https://e-sms.dialog.lk/api/v1/message-via-url/create/url-campaign?esmsqk=syGciOiJIUzI1MiIsInR7nCI6IkpXVCJ9.eyJpZCI6MTUsImlhdCI6MTY1NTM1 MDkyNCwiZXhwIjo0Nzc5MferMzI0fQ.EqylAmt1_CWOEpcrQA--kRiW-

qFHNktMTz6Y1YDA1f4&list=0717881242,763625800,94777584536&source_address=adeona&message=Welcome&push_notification_url=https://xx/xx" \

--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>

Note: Get Request SMS API has demonstrated its reliability and efficiency within the tested limit of 100 recipient numbers. While the API is designed to handle high message volumes, surpassing this limit may have adverse effects on its performance for the particular user. We strongly advise against exceeding the recommended limit to prevent potential disruptions in service, delayed message delivery, or degraded performance.

3.2.2. Check Account Balance Via GET Request

Prerequisite: Only available for users who are eligible for sending SMS via GET requests. (Section 3.2)

End point: https://e-sms.dialog.lk/api/v1/message-via-url/check/balance?esmsqk=<key>

Method: HTTP GET

Request Params

Parameter name		Description	Mandatory/ Optional 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

Sample Request

https://e-sms.dialog.lk/api/v1/message-viaurl/check/balance?esmsqk=eyJhbGciOiJIUzI1NiIsInR5c

If success

1|<balance>

Pipe mark separate the status and the balance. In the above,1 is the success and it is followed by "|" and then the account balance

If failed

<error_id>|0

Pipe mark separates the status and the balance. In the above, <error_id> is for a failure (check table below) and it is followed by "|" and the account balance will be 0 for all failed requests.

Response id	Description
1|<balance>	Success
2001|0	Error occurred during campaign creation
2002|0	Bad request
2006|0	Not eligible to send messages via get requests (Admin haven't provided the access level)
2007|0	Invalid key (esmsqk parameter is invalid)

Above response id(s) only valid for send SMS via get requests (section 3.2).

3.2.3. Error Codes – GET Request

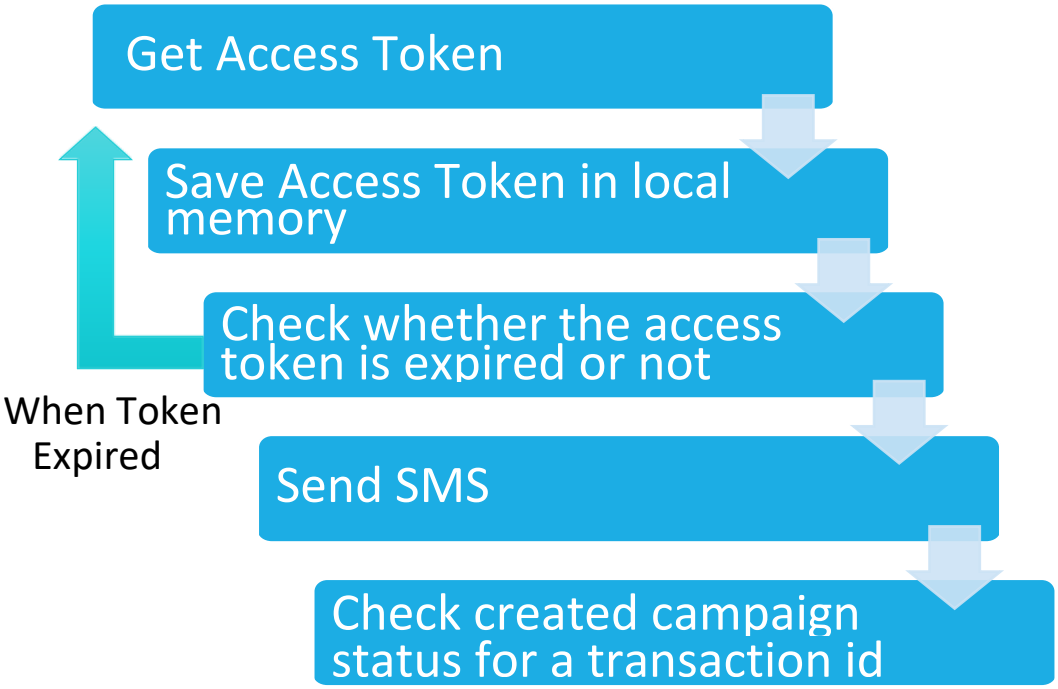
Response id	Description
1	Success
2001	Error occurred during campaign creation
2002	Bad request
2003	Empty number list
2004	Empty message body
2005	Invalid number list format
2006	Not eligible to send messages via get requests (Admin haven't provided the access level)
2007	Invalid key (esmsqk parameter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

2008	Not enough money in the user's wallet or not enough messages left in the package for the user. (When consuming package payments)
2009	No valid numbers found after the removal of mask blocked numbers.
2010	Not eligible to consume packaging
2011	Transactional error
2012	Doesn't have the access for mask.
2020	Too many requests

Above response id(s) only valid for send SMS via get requests (section 3.2).

4. Plugins

4.1. Working Scenario



4.2. Java

4.2.1. Supported Projects

4.2.1.1. Maven projects

4.2.1.2. Spring Boot maven projects

4.2.2. Installation

Then add the dependency to your project.

Apache Maven

```
<dependency>
    <groupId>net.adeonatech</groupId  >
    <artifactId>SmsAPI</artifactId>
    <version>1.0.5</version>
</dependency>
```

Gradle Groovy DSL

```
implementation  'net.adeonatech:SmsAPI:1.0.5'
```

4.2.3. Quick Start

4.2.3.1. 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();
```

4.2.4. Save Access Token in Local Memory

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

4.2.5. 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.

4.2.6. 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();
```

4.2.7. 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).getToken()
).getDataTransaction
().getCampaign_status();
```

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.

4.3. PHP

4.3.1. Installation

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

```
<?php

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

?>
```

4.3.2. Quick Start

4.3.2.1. 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();
```


4.3.3. Save Access Token in Local Memory

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

4.3.4. 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.

4.3.5. 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();
```


4.3.6. 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();
```

5. API Utilization Related Restrictions

Each account will receive a specific throughput (TPS) to

- send SMS (TPS: 20),
- consuming the API (TPS: 30),
- campaign status check (TPS: 2).

6. FAQ

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. Please request administrator assistance to enable the feature.

I'm getting error 104 when I retry sending a failed request. What does this mean?

- The error indicates that you are using a previously used transaction ID. The system requires transaction IDs to be unique to prevent possible duplicates caused by retransmissions.

- This means that your request has already been registered by the system, regardless of whether it was successful or failed. Therefore, when retrying, you should use a different transaction ID to ensure the message is sent according to your application's logic.

