Liveall.eu Docs

Release 12

Mike Nakos

Jun 19, 2023

HTTP API

You can find here all the reference docs about APIs and more, regarding Liveall.eu

CHAPTER

ONE

FORM API (X-WWW-FORM-URLENCODED API)

This type of Liveall.eu API is based on the form x-www-form-urlencoded type of web-requests (**POST**) as described here: POST HTTP - MDN Web Docs

The `Content-Type` on the request headers is set to `application/x-www-form-urlencoded`

Note: If you like to send multiple SMS with one request you are advised to use the *JSON API (application/json API)*, which offers more flexibility.

1.1 Send SMS

Contents

1.1.1 Endpoint URL

The end-point for sending SMS via HTTP (**POST**) calls is the following:

```
https://sms.liveall.eu/apiext/Sendout/SendSMS
```

1.1.2 curl example

```
curl --location --request POST 'https://sms.liveall.eu/apiext/Sendout/SendSMS' \
  --header 'Content-Type: application/x-www-form-urlencoded' \
  --data-urlencode 'apitoken=7ace3e49cae13ae4f5ccb8a6a8a0d6a8fe120aa82ae46ad6ee4c9d8' \
  --data-urlencode 'destination=306912345678' \
  --data-urlencode 'senderid=mySender' \
  --data-urlencode 'message=This is a test message from me!'
```

1.1.3 Variables

apitoken
: `string` a unique hash code for each account that authorizes each web request. That code you can find it on your account's page

destination
: `string` the cell's number (without leading zeros or + sign), for example for Greece: 306912345678. In case you need to send the same message **to more than one recipients**, then you may supply that variable with these numbers delimited by one of the following characters `;.^` **It is recommended** to send batches with a single request instead of making multiple requests, in case you want to send the same text to multiple destinations

senderid
: `string` the sender name of the SMS. There is a limit to 11 characters (latin characters). Allowed characters are: `[A-Za-z0-9\-\.\!\#\%\&\(\)\<\>]`

message
: `string` the SMS text

sendon
: `(optional) - unsigned integer` an optional scheduling parameter. You can define a future datetime a message to be sent. This variable is a type of unsigned integer - unix timestamp. You can find more reference on https://dev.mysql.com/doc/refman/5.5/en/date-and-time-functions.html#function_unix-timestamp That is, in case you want to send the message on 2016-07-06 12:17:45 you must provide the value 1467796665

pricecat
: `(optional) - unsigned integer` by setting that parameter you can choose between normal and low cost price category (where applicable). Set 1 in case you want to send the message with low cost, or ignore it or set the value to 0, in case you want to send with normal cost

Note: If you want to test the API we recommend to use the Postman.

1.1.4 Error Response

In case of error, the result could be like the following:

```
Error: <Error code> - <Error message>
```

where:

<Error code>	is the request's error code as shown below
<Error message>	is the error message, describing the problem with the request

1.1.5 Successful Response

In case of success, for a single destination number, we get the below result, where ID is the SMS id:

```
OK ID:123456789
```

(ID: is the submitted SMS id number) and in case of multiple destinations we get something like the below:

```
OK ID:123456787|OK ID:123456788|OK ID:123456789
```

1.2 Check the status of a submitted message

Contents

1.2.1 Endpoint URL

The end-point for sending SMS via HTTP (**POST**) calls is the following:

```
https://sms.liveall.eu/apiext/Sendout/GetSMSStatus
```

1.2.2 curl example

```
curl --location --request POST 'https://sms.liveall.eu/apiext/Sendout/GetSMSStatus' \
  --header 'Content-Type: application/x-www-form-urlencoded' \
  --data-urlencode 'apitoken=7ace3e49cae13ae4f5ccb8a6a8a0d6a8fe120aa82ae46ad6ee4c9d8' \
  --data-urlencode 'smsids=20817547,20818326'
```

1.2.3 Variables

apitoken
: a unique hash code for each account that authorizes each web request. That code you can find it on your account's page

smsids
: supply the SMS id(s) of the already submitted message(s). You may use one of the following delimiters, between SMS ids, in case you want to provide more than one message: `,.^`

1.2.4 Error Response

In case of error, the result could be like the following:

```
Error:  <Error code> - <Error message>
```

where:

<Error code>	is the request's error code as shown below
<Error message>	is the error message, describing the problem with the request

---

1.2.5 Successful Response

and in case of success, for one message, the result would be in the form of:

```
<SMSId>:<Submtited On>:<Last status datetime>:<Destination number>:<Status
number>:<Status text>:<Quantity of SMS>:<Charge amount>
```

For example:

```
20817547:1465021934:1465021977:306912456789:200000:Delivered:1:0.0379
```

For the case you want the status for more than one messages, you will be returned with the results delimiter with character |, for example:

```
20817547:1465021934:1465021977:306912456789:200000:Delivered:1:0.
0379|20818326:1467226402:0:306912345789:100007:Queued:1:0
```

1.2.6 Description of result fields

Field	Description
SMSId Integer	the sms id
Submited On (Integer)	the date and time of the message's submission
Last status datetime (Integer)	the datetime of the last status of message
Destination number (String)	the cell's number
Status number (Integer)	the numeric status code (*)
Quantity of SMS (Integer)	how many SMS are consumed for the message
Charge amount (Float)	the total charge of the message

1.3 Check the account balance

Contents

1.3.1 Endpoint URL

The end-point for sending SMS via HTTP (**POST**) calls is the following:

```
https://sms.liveall.eu/apiext/Sendout/GetAccountBalance
```

1.3.2 curl example

```
curl --location --request POST 'https://sms.liveall.eu/apiext/Sendout/GetAccountBalance'␣
↪\
  --header 'Content-Type: application/x-www-form-urlencoded' \
  --data-urlencode 'apitoken=7ace3e49cae13ae4f5ccb8a6a8a0d6a8fe120aa82ae46ad6ee4c9d8' \
  --data-urlencode 'countryprefix=30'
```

1.3.3 Variables

apitoken
the api token (as mentioned in previous end-points)

countryprefix (optional)
an optional country code. If you provide that, you will get the available SMS count, based on the price of the provided country (normal & low cost)

1.3.4 Successful Responses

- Case we don't provide country code

```
OK Balance:169.64|SmsRemainCount:-1|LCSmsRemainCount:-1
```

- Case that we provide the country code

```
OK Balance:169.64|SmsRemainCount:4475|LCSmsRemainCount:5317
```

Resp. Variable	Description
Balance	the account's balance in euros
SmsRemainCount	the remaining SMS (for the case we want to send with normal cost)
LCSmsRemainCount	the remaining SMS (for the case we want to send with low cost)

1.3.5 Error Response

```
Error:  <Error code> - <Error message>
```

where:

<Error code>	is the request's error code as shown below
<Error message>	is the error message, describing the problem with the request

1.4 Extract message log for a date

1.4.1 Description

This endpoint extract the messages log for a specific date in a string format (csv)

1.4.2 Endpoint URL

```
https://sms.liveall.eu/apiext/Sendout/GetSMSHistory
```

1.4.3 curl example

```
curl --location --request POST 'https://sms.liveall.eu/apiext/Sendout/GetSMSHistory' \
    --header 'Content-Type: application/x-www-form-urlencoded' \
    --data-urlencode 'apitoken=7ace3e49cae13ae4f5ccb8a6a8a0d6a8fe120aa82ae46ad6ee4c9d8' \
    --data-urlencode 'submit_date=20210524'
```

1.4.4 Form parameters

apitoken
: `string` a unique hash code for each account that authorizes each web request. That code you can find it on your account's page

submit_date

: `string` we define the date we want to extract the messages log. The date format must be in a form of: **yyyyMMdd**
where **yyyy** is the 4 digit year numeber
MM the 2 digit month number
dd the date of the month
For instance, if we need to get the log for 24 of May 2021, then string value would be: `20210524`

timezone_offset

: `(optional) - integer` this is your timezone value **based on UTC**. you have to specify it if you'are not in the same timezone as Athens/Greece.
Bear in mind that, `GMT is not UTC` - (GMT: Greenwich Mean Time, UTC: Coordinated Universal Time), because on winter time, GMT is +0 and on summer time is +1.
So for example if you're in Rome and your computer has winter time you must define 1 and on summer 2.
Accordingly, if you're in New York, on winter you enter -5 and on summer -4

senderid
: `(optional filter) - string` Filter results based on the sender_id

destination
: `(optional filter) - string` Filter results based on the destination(s) number

sms_id
: `(optional filter) - integer` Filter results based on the SMS id

batch_id
(optional filter) - integer Filter results based on the batch id

gt_sms_id
(optional filter) - integer Filter, gets rows with SMS id greater than the value specified. Suppose there are results with SMS id (1, 2, 3, 4, 5) and we define 2, then we will get the rows (3, 4, 5)

1.4.5 Error Response

There are 2 cases of error:

Validation issue, which service will reply with

```
Error: <Error code> - <Error message>
```

where:

<Error code>	is the request's error code as shown below
<Error message>	is the error message, describing the problem with the request

Internal error

In that case, service will return http status: 500 as descibed here: https://en.wikipedia.org/wiki/List_of_HTTP_status_codes#5xx_Server_errors

---

1.4.6 Successful Response

In case of success, we will have a string response with so many lines as the rows found. If no row found, then we will get an empty string. Lines are delimited with character \n. Below is an example of a successful response with 2 rows found:

```
47680777|8350040|terracom|306912345678|1585742558|Delivered|1|0.0379|UNDEFINED
47680768|8350041|Liveall.eu|306912345679|1585742462|Delivered|1|0.0379|UNDEFINED
```

Below are the descriptions of each field of the above example response:

1. sms id
2. batch id
3. sender id
4. recipient's phone number
5. Last status datetime, in Unixtime - You can check the value here: Online epoch converter
6. SMS Status string value - Possible values are here:

1.5 Operation result codes

```
enum DLR_CODES {
   // Extended error codes
   EXT_DLR_UNDEFINED = -1,
   EXT_DLR_NO_ERROR = 0,
   EXT_DLR_ABSENT_SUBSCR = 1,
   EXT_DLR_HNDSET_FULL_MEM = 2,
   EXT_DLR_EQ_PROTOCOL_ERROR = 3,
   EXT_DLR_NO_SMS_ABILITY = 4,
   EXT_DLR_UNKNOWN_SUBSCR = 5,
   EXT_DLR_ILLEGAL_SUBSCR = 6,
   EXT_DLR_IMEI_ILLEGAL = 8,
   EXT_DLR_CALL_BARRED = 9,
   EXT_DLR_BUSY_SUBSCR = 11,
   EXT_DLR_SYSTEM_FAILURE = 12,
   EXT_DLR_UNEXPECTE_DT_VAL = 14,
   EXT_DLR_UNIDENTIFIED_SUBSCR = 15,
   EXT_DLR_ABS_SUBSCR_1 = 16,
   EXT_DLR_ABS_SUBSCR_2 = 17,
   EXT_DLR_ABS_SUBSCR_ROAM_RESTR = 18,
   EXT_DLR_WINDOW_EXCIDED = 50,
   EXT_DLR_EXPIRED = 52,
   EXT_DLR_INVALID_DEST = 54,
   EXT_DLR_NO_ROUTE_FOR_SMS = 55,
   EXT_DLR_BUFFERED = 56,
   EXT_DLR_ABS_SUBSCR_3 = 59,
   EXT_DLR_PORTABILITY_ERR = 60,
   EXT_DLR_UNKNOWN_ERR1 = 255,
   EXT_DLR_SENT_NO_DLR_RECEIVED = 713,
   EXT_DLR_FLOOD_PROTECT = 716,
   EXT_DLR_INVALID_DEST_ADRR = 718,
   EXT_DLR_PERM_OPER_ERR = 920,
   EXT_DLR_OPER_NET_FAIL = 925,
   EXT_DLR_PHONE_RELATE_ERR = 926,
   EXT_DLR_PERM_PHONE__REL_ERR = 927,
   EXT_DLR_ANTI_SPAM = 928,
   EXT_DLR_NO_ROUTE_FOR_SMS_2 = 1027,
   EXT_DLR_UNKNOWN_OPER = 1205,
   EXT_DLR_REJECT_BY_SMSC = 2256,
   EXT_DLR_SERV_CNTR_CONG = 3001,
   EXT_DLR_TELSRVCE_NOT_PROV = 3002,
   EXT_DLR_ILLEGAL_EQUIPMENT = 3003,
   EXT_DLR_DATA_MISSING = 3004,
   EXT_DLR_RESRC_LIMITATION = 3005,
   EXT_DLR_ILLEGAL_ERR_CODE = 3006,
   EXT_DLR_NETWORK_TIMEOUT = 3007,
   EXT_DLR_OPER_BARRED_DEST_OPERTR = 3008,
   EXT_DLR_DELIVERY_FAILED = 3009,
   EXT_DLR_HLR_FAILURE = 3010,
   EXT_DLR_VLR_FAILURE = 3011,
   EXT_DLR_CTRL_MSC_FAILURE = 3012,
```

(continues on next page)

(continued from previous page)

```
    EXT_DLR_VISITED_MSC_FAILURE = 3013,
    EXT_DLR_MSG_STORE_BUSY = 3014,
    EXT_DLR_SME_INTERFACE_BUSY = 3015,
    EXT_DLR_CLSD_USR_GRP_REJECT = 3016,
    EXT_DLR_DEFERRED_DELIVERY = 3017,
    EXT_DLR_REJECTED_DUE_BLCK_ISSUE = 3018,
    EXT_DLR_ERROR_IN_SMSC = 3019,
    EXT_DLR_REJ_OPER_DUE_VAL_PER_EXPR = 3020,
    EXT_DLR_RETRIED_TO_DELIVER = 3021,
    EXT_DLR_NO_DLR_FROM_OPERATOR = 3022,
    EXT_DLR_ADULT_PARENT_LOCK = 3023,
    EXT_DLR_PORT_COMB_FAILURE = 3024,
    EXT_DLR_AORT_FROM_HLR = 3025,
    EXT_DLR_SRC_ADDR_BLACKLST_UNSUPP = 3026,
    EXT_DLR_OPERATION_BARRED = 3027,
    EXT_DLR_REJ_DUE_TO_DUPLICATE = 3028,
    EXT_DLR_INFO_NOT_AVAILABLE = 3029,
    EXT_DLR_Msg_WAIT_LIST_FULL = 3030,
    EXT_DLR_OTHER_ERROR = 65500,
    EXT_DLR_TEM_SYS_ERROR = 65501,
    //EXT_DLR_SEEN = 65502,

    // Basic error codes
    BSC_STATUS_SENT = 100001,
    BSC_STATUS_QUEUED_SMSC = 100002,
    BSC_STATUS_UNDELIVRD = 100003,
    BSC_STATUS_NONDLVRD_SMSC = 100004,
    BSC_STATUS_ERROR = 100005,
    BSC_STATUS_AWAIT_HLR = 100006,
    BSC_STATUS_QUEUED = 100007,
    BSC_STATUS_TMP_UNAVLBL = 100008,
    BSC_STATUS_HLR_SENT = 100009,
    BSC_STATUS_HLR_COMPLETED = 100010,
    BSC_STATUS_DELIVERED = 200000,

    // IM error codes
    IM_UNDEFINED = 300000,
    IM_ACCEPTD = 300001,
    IM_DELIVRD = 300002,
    IM_EXPIRED = 300003,
    IM_DELETED = 300004,
    IM_UNDELIV = 300005,
    IM_REJECTD = 300006,
    IM_UNKNOWN = 300007,
    IM_SEEN = 300008,
    IM_SENT = 300009,
    IM_NYSENT = 300010,
    IM_OPEN = 300011,
}
```

CHAPTER

TWO

JSON API (APPLICATION/JSON API)

This type of Liveall.eu API offers more flexibilty when using it, since you can easily and straight-forward send messages to many recipients with a different text for each one. With advanced API type you can send SMS and Viber messages.

The `Content-Type` on the request headers is set to `application/json` and all the data structure is a text, representing a **JSON object**. Likewise, the response is not a delimited string but a JSON object

2.1 Send SMS

Contents

2.1.1 Description

Using that type of API gives you the flexibility to send to a single or multiple destinations SMS by calling the web-service once. Also you are dealing with a JSON object (as a payload), which is much more straight-forward to a programer.

Note: you can send a different text to each destination as you may have noticed below

2.1.2 Endpoint URL

```
https://sms.liveall.eu/apiext/Sendout/SendJSMS
```

2.1.3 curl example

```
curl --location --request POST 'https://sms.liveall.eu/apiext/Sendout/SendJSMS' \
--header 'Content-Type: application/json' \
--data-raw '  {
      "apitoken": "7ace3e49cae13ae4f5ccb8a6a8a0d6a8fe120aa82ae46ad6ee4c9d8",
      "senderid": "mySender",
      "messages": [
          {
              "destination": "306912345678",
              "message": "Test message A"
          },
          {
              "destination": "306912345677",
              "message": "Test message B"
          }
      ]
  }
'
```

2.1.4 JSON object example

```
{
    "apitoken": "7ace3e49cae13ae4f5ccb8a6a8a0d6a8fe120aa82ae46ad6ee4c9d8",
    "senderid": "mySender",
    "messages": [
        {
            "destination": "306912345678",
            "message": "Test message A"
        },
        {
            "destination": "306912345677",
            "message": "Test message B"
        }
    ]
}
```

2.1.5 JSON Object variables

apitoken
`string` a unique hash code for each account that authorizes each web request. That code you can find it on your account's page

senderid
`string` the sender name of the SMS. There is a limit to 11 characters (latin characters). Allowed characters are: `[A-Za-z0-9\-\.\!\#\%\&\(\)\<\>]`

messages
`object` is an array of objects that holds the data of the message, as shown in the above example. Object consists of 2 properties: **destination** (the cell's number (without leading zeros or + sign), for example for Greece: 306912345678), and **message** (the message's text)

sendon
`(optional) - unsigned integer` an optional scheduling parameter. You can define a future datetime a message to be sent. This variable is a type of unsigned integer - unix timestamp. You can find more reference on https://dev.mysql.com/doc/refman/5.5/en/date-and-time-functions.html#function_unix-timestamp That is, in case you want to send the message on 2016-07-06 12:17:45 you must provide the value 1467796665

pricecat
`(optional) - unsigned integer` by setting that parameter you can choose between normal and low cost price category (where applicable). Set 1 in case you want to send the message with low cost, or ignore it or set the value to 0, in case you want to send with normal cost

---

2.1.6 Error Response

In case of error, we get something like the below:

```
{
    "success": false,
    "OperationErrors": [
        {
            "errorCode": 13,
            "errorMessage": "Invalid destination number",
            "SMSErrorType": 3,
            "valueOfError": "3069"
        }
    ],
    "SubmissionID": 0,
    "data": null
}
```

[success] will be false and you'll find the object **[OperationErrors]** with error details

For more details see the *APPENDIX*

2.1.7 Successful Response

```
{
    "success": true,
    "OperationErrors": null,
    "SubmissionID": 0,
    "data": [
        {
            "destination": "306912345678",
            "smsid": 20818588
        },
        {
            "destination": "306912345677",
            "smsid": 20818589
        },
        {
            "destination": "306912345676",
            "smsid": 20818590
        }
    ]
}
```

[success] is true and the **[data]** property contains the **[smsid]** for each SMS

2.1.8 APPENDIX

Response properties

Name	Description
success	when false, then no message sent and the whole request is considered failed
OperationErrors	when success is false, we get an array of objects with errors. Each object has 4 properties: **errorCode**: the error code (integer) of the error, **errorMessage**: the descriptive text of the error and **SMSErrorType**: this indicates the source of the problem (please see below) **valueOfError**: the value that caused the error (for debugging or troubleshooting purposes)
data	in case of success, web-service is returning an array ob objects - one for each destination, having 2 properties: **destination**: the cell's number and **smsid**: the unique id of the SMS

OperationErrors

This is an array with objects having the properties `errorCode`, `errorMessage`, `SMSErrorType`, `valueOfError`. In case of success this object is null

errorCode

tabid
errCD

```
public enum SMS_SERVICE_ERROR_CODES
{
    NO_ERROR                            = 0,
    EMPTY_SENDERID                      = 1,
    INVALID_SENDERID                    = 2,
    UNAUTHORIZED_NUM_SENDER_ID          = 3,
    ALPHA_SENDERID_TOO_LONG             = 4,
    NUM_SENDERID_TOO_LONG               = 5,
    INTERR_NO_SMS_TYPE_PROV             = 6,
    INTERR_NO_SMS_TEXT                  = 7,
    INTERNAL_ERROR                      = 8,
    ILLEGAL_SENDERID                    = 9,
    SMS_TEXT_EMPTY                      = 10,
    SMS_TEXT_LEN_TOO_LONG               = 11,
```

(continues on next page)

(continued from previous page)

```
    NO_DESTINATION_NUMBERS_PROVIDED     = 12,
    INVALID_DESTINATION_NUMBER          = 13,
    INVALID_GREEK_DEST_NUM              = 14,
    INVALID_CYPR_DEST_NUM               = 15,
    INVALID_ITALIAN_DEST_NUM            = 16,
    NOTFOUND_BUFFERED_BATCH_HEAD        = 17,
    INSUFFICIENT_USER_BALANCE           = 18,
    INTERR_COULDNT_FOUND_BUFFBATCH      = 19,
    INVALID_BATCHID_GIVEN               = 20,
    ERROR_CREATING_SMSLOGFILE           = 21,
    ERROR_WHEN_TRYING_TO_BLACKLIST      = 22,
    ERROR_ON_GETTING_CONTACTS           = 23,
    ERROR_NO_CONTACT_TO_DELETE          = 24,
    RECORD_ALREADY_EXISTS               = 25,
    RECORD_DOES_NOT_EXISTS              = 26,
    RECORD_CHANGE_FROM_DIFF_SESSION     = 27,
    PBOOK_CONTACT_CELL_EMPTY            = 28,
    PBOOK_CONTACT_NAME_EMPTY            = 29,
    PBOOK_INVLD_CELL                    = 30,
    PBOOKGRP_NO_GROUP_PRVD_TO_DEL       = 31,
    ACCSETT_EMPTY_SETTINGS              = 32,
    INVALID_IMPORT_FILE                 = 33,
    INSUFFICIENT_INVLD_PARAMETER_DATA   = 34,
    ERROR_IMPORTING_CONTACTS            = 35,
    INS_UPD_DUPLICATE_CELL_FOUND        = 36,
    NOT_ENOUGH_CREDITS_FOR_HLR_QUERY    = 37,
    ERROR_WHEN_TRYING_SUBMIT_USERHLR    = 38,
    API_TOKEN_NOT_PROVIDED              = 39,
    API_TOKEN_MISMATCH                  = 40,
    INVALID_SCHEDULED_SENDOUT_DATE      = 41,
    SMSIDS_PARAMETER_INVALID            = 42,
    NO_SUBMITTED_SMS_FOUND              = 43,
    INVALID_API_TOKEN                   = 44,
    VOUCHER_FROM_DIFFERENT_DOMAIN       = 45,
    VOUCHER_NOT_FOUND_OR_NON_FREE       = 46,
    VOUCHER_AMOUNT_CREDIT_FAILED        = 47,
    ERROR_UPDATING_CHARGED_VOUCHER      = 48,
    ERROR_DATA_NOT_FOUND                = 49,
    APITOKEN_USR_BELONGS_OTHER_MASTER   = 50,
    SUBACCOUNT_ALREADY_ASSIGNED         = 51,
    SENDERID_TOO_SHORT                  = 52,
    ERROR_CREATING_FILE                 = 53,
    IM_TEXT_EMPTY                       = 54,
    IM_TEXT_LONGER_THAN_EXPECTED        = 55,
    IM_SENDERID_NOT_APPROVED            = 56,
    IM_IMAGE_INVALID                    = 57,
    IM_ACTION_INVALID                   = 58,
    EMPTY_OR_INVALID_PARAMETERS         = 59,
    DATA_VERIFICATION_ERROR             = 60,
    SENDERID_INJ_NUMERIC_DETECTED       = 61,
    SMSFORM_NO_VALUETOKEN_FOUND         = 62,
    SMSFORM_NO_FORM_DATA_FOUND          = 63,
```

(continues on next page)

(continued from previous page)

```
}
```

SMSErrorType

tabid
SMSErrType

```
public enum SMS_INGRENTIENT_TYPES
{
    SENDERID        = 1,
    TEXT            = 2,
    DESTINATION_NUM = 3,
    OTHER           = 4,
}
```

2.2 Send SMS with a form link

Contents

2.2.1 Description - How it works

That document describes the same API as the *Send SMS* but it covers the case when we send form links to the message recipients. The usage is simple enough and goes like this:

1. provide an additional property [forms] as described here: *[forms] object*
2. SMS body must have a token: `{%form_link_token}`. That token will be replaced with the final URL that message recipient will tap on his phone
3. object for each destination must also have a property `value_tokens` that is a key/value holder with the values of each parameter on the URL, as described on *[value_tokens] object* section
4. the above key/value holder replaces the values of the corresponding param for each destination

Note: See also the *Step-by-step demonstration*

2.2.2 Endpoint URL

The end-point to use for send-out is:

```
https://sms.liveall.eu/apiext/Sendout/SendJSMS
```

2.2.3 curl example

```
curl --location --request POST 'https://sms.liveall.eu/apiext/Sendout/SendJSMS' \
--header 'Content-Type: application/json' \
--data-raw '{
    "apitoken": "7ace3e49cae13ae4f5ccb8a6a8a0d6a8fe120aa82ae46ad6ee4c9d8",
    "senderid": "mySender",
    "messages": [
        {
            "destination": "306912345678",
            "message": "Test message A"
        },
        {
            "destination": "306912345677",
            "message": "Test message B. In order to consent, tap on the following link {
↪%form_link_token}",
            "value_tokens": {
              "SOMEID_VAL_TOKEN": "306912345677-j22Jd4p4c6"
            }
        },
        {
            "destination": "306912345676",
            "message": "Test message C. In order to consent, tap on the following link {
↪%form_link_token}",
            "value_tokens": {
```

(continues on next page)

(continued from previous page)

```
            "SOMEID_VAL_TOKEN": "ZTq9Jzj8LH"
          }
        }
    ],
    "forms": {
      "actual_url": "https://forms.onlineformsservice.example/myforms/get/?mysoid=
↪{SOMEID_VAL_TOKEN}"
    }
   }'
```

2.2.4 JSON object example

The following JSON shows a possible payload for SMS send-out, that send a different text to each destination with a single request:

```
{
 "apitoken": "7ace3e49cae13ae4f5ccb8a6a8a0d6a8fe120aa82ae46ad6ee4c9d8",
 "senderid": "mySender",
 "messages": [
     {
         "destination": "306912345678",
         "message": "Test message A"
     },
     {
         "destination": "306912345677",
         "message": "Test message B. In order to consent, tap on the following link {
↪%form_link_token}",
         "value_tokens": {
           "SOMEID_VAL_TOKEN": "306912345677-j22Jd4p4c6"
         }
     },
     {
         "destination": "306912345676",
         "message": "Test message C. In order to consent, tap on the following link {
↪%form_link_token}",
         "value_tokens": {
           "SOMEID_VAL_TOKEN": "ZTq9Jzj8LH"
         }
     }
 ],
 "forms": {
   "actual_url": "https://forms.onlineformsservice.example/myforms/get/?mysoid={SOMEID_
↪VAL_TOKEN}"
 }
}
```

2.2.5 JSON Object variables

apitoken
`string` a unique hash code for each account that authorizes each web request. That code you can find it on your account's page

senderid
`string` the sender name of the SMS. There is a limit to 11 characters (latin characters). Allowed characters are: `[A-Za-z0-9\-\.\!\#\%\&\(\)\<\>]`

messages
`object` is an array ob objects that holds the data of the message. Object consists of 3 properties: **[destination]** (the cell's number (without leading zeros or + sign), for example for Greece: 306912345678), **[message]** (the message's text) and the *[value_tokens] object*

sendon
`(optional) - unsigned integer` an optional scheduling parameter. You can define a future datetime a message to be sent. This variable is a type of unsigned integer - unix timestamp. You can find more reference on https://dev.mysql.com/doc/refman/5.5/en/date-and-time-functions.html#function_unix-timestamp That is, in case you want to send the message on 2016-07-06 12:17:45 you must provide the value 1467796665

pricecat
`(optional) - unsigned integer` by setting that parameter you can choose between normal and low cost price category (where applicable). Set 1 in case you want to send the message with low cost, or ignore it or set the value to 0, in case you want to send with normal cost

forms
`object` an object that has form data. Please read *[forms] object*

2.2.6 [value_tokens] object

[value_tokens] object contains **key/values** with URL parameter name and its value to set for each SMS sent to the recipient. For example, for the below URL:

`?name=<USERNAME_VALUE_TOKEN>&enabled=<ISENABLED_VALUE_TOKEN>&campaign_source=email`

for a **specific** destination, we would have the following JSON object:

```
{
   "destination": "3069XXXXXXXX",
   "message": "Test message A. In order to consent, tap on the following link {%form_
↪link_token}",
   "value_tokens": {
      "USERNAME_VALUE_TOKEN": "Mike",
      "ISENABLED_VALUE_TOKEN": "true"
   }
}
```

meaning that, [**value_tokens**] object will have as many entries as the parameters that must have different value for each destination. In the above example the URI query has 3 parameters, but we need to have different values only on 2 of them, since the 1 is static

---

2.2.7 [forms] object

[forms] object currently contains a property called [actual_url]. Its data type is a `string` and it holds the actual URL of the form. This URL will be shortened by our **internal shortener system** - will shorten the URL part that **does not** contain the URL parameters.

Listing 1: How is the long URL with its parameters being shortened

```
1. Long URL:
https://forms.onlineformsservice.example/myforms/get/?mysoid={SOMEID_VAL_TOKEN}#23

2. Part of the URL to be shortened
https://forms.onlineformsservice.example/myforms/get/

3. Short URL with all the parameters
https://lval.eu/XXX?mysoid={SOMEID_VAL_TOKEN}#23
```

Listing 2: How the tokens are replaced

```
For a destination for example with SOMEID_VAL_TOKEN=ZTq9Jzj8LH, the final URL would be:
https://lval.eu/1?mysoid=ZTq9Jzj8LH#23
```

As you can see in the *JSON object example*, there is a token: `{SOMEID_VAL_TOKEN}` which that will be replaced by the `SOMEID_VAL_TOKEN` value of the `value_tokens` key/value object and have a different value for each destination.

2.2.8 Error Response

In case of error, we get something like the below:

```
{
    "success": false,
    "OperationErrors": [
        {
            "errorCode": 13,
            "errorMessage": "Invalid destination number",
            "SMSErrorType": 3,
            "valueOfError": "3069"
        }
    ],
    "SubmissionID": 0,
    "data": null
}
```

[success] will be false and you'll find the object **[OperationErrors]** with error details

For more details see the *APPENDIX*

2.2.9 Successful Response

```
{
    "success": true,
    "OperationErrors": null,
    "SubmissionID": 0,
    "data": [
        {
            "destination": "306912345678",
            "smsid": 20818588
        },
        {
            "destination": "306912345677",
            "smsid": 20818589
        },
        {
            "destination": "306912345676",
            "smsid": 20818590
        }
    ]
}
```

[success] is true and the **[data]** property contains the **[smsid]** for each SMS

2.2.10 Step-by-step demonstration

Step 1

tabid
first-jsonPl

Json payload

```
{
    "apitoken": "7ace3e49cae13ae4f5ccb8a6a8a0d6a8fe120aa82ae46ad6ee4c9d8",
    "senderid": "mySender",
    "messages": [
        {
            "destination": "306912345676",
            "message": "Test message C. In order to consent, tap on the following link {
↪%form_link_token}",
            "value_tokens": {
                "SOMEID_VAL_TOKEN": "ZTq9Jzj8LH"
            }
        }
    ],
    "forms": {
    "actual_url": "https://forms.onlineformsservice.example/myforms/get/?mysoid={SOMEID_
↪VAL_TOKEN}"
    }
}
```

Step 2

tabid
2nd-shortenURL

Long URL is shortened

`https://forms.onlineformsservice.example/myforms/get/?mysoid={SOMEID_VAL_TOKEN}` is splitted like this

First part: `https://forms.onlineformsservice.example/myforms/get/`
Second part: `?mysoid={SOMEID_VAL_TOKEN}`

First part is shortened, for example to this: `https://lval.eu/1`
So the final URL turns into the: `https://lval.eu/1?mysoid={SOMEID_VAL_TOKEN}`

Step 3

tabid
3rd-val-replace

Values replacement

Since `value_tokens` object has a key with a name `SOMEID_VAL_TOKEN`, its value will replace the `{SOMEID_VAL_TOKEN}` part on the URI

```
https://lval.eu/1?mysoid={SOMEID_VAL_TOKEN}
turns into the
https://lval.eu/1?mysoid=ZTq9Jzj8LH
```

Step 4

tabid
4th-final-form

Final formation of the SMS text

The final SMS text will become:

```
Test message C. In order to consent, tap on the following link https://lval.eu/1?
↪mysoid=ZTq9Jzj8LH
```

2.2.11 APPENDIX

Response properties

Name	Description
success	when false, then no message sent and the whole request is considered failed
OperationErrors	when success is false, we get an array of objects with errors. Each object has 3 properties: **errorCode**: the error code (integer) of the error, **errorMessage**: the descriptive text of the error and **SMSErrorType**: this indicates the source of the problem (please see below) **valueOfError**: the value that caused the error (for debugging or troubleshooting purposes)
data	in case of success, web-service is returning an array ob objects - one for each destination, having 2 properties: **destination**: the cell's number and **smsid**: the unique id of the SMS

OperationErrors

This is an array with objects having the properties `errorCode`, `errorMessage`, `SMSErrorType`, `valueOfError`. In case of success this object is null

errorCode

tabid
errCD

```
public enum SMS_SERVICE_ERROR_CODES
{
    NO_ERROR                            = 0,
    EMPTY_SENDERID                      = 1,
    INVALID_SENDERID                    = 2,
    UNAUTHORIZED_NUM_SENDER_ID          = 3,
    ALPHA_SENDERID_TOO_LONG             = 4,
    NUM_SENDERID_TOO_LONG               = 5,
    INTERR_NO_SMS_TYPE_PROV             = 6,
    INTERR_NO_SMS_TEXT                  = 7,
    INTERNAL_ERROR                      = 8,
    ILLEGAL_SENDERID                    = 9,
    SMS_TEXT_EMPTY                      = 10,
    SMS_TEXT_LEN_TOO_LONG               = 11,
    NO_DESTINATION_NUMBERS_PROVIDED     = 12,
    INVALID_DESTINATION_NUMBER          = 13,
```

(continues on next page)

(continued from previous page)

```
    INVALID_GREEK_DEST_NUM              = 14,
    INVALID_CYPR_DEST_NUM               = 15,
    INVALID_ITALIAN_DEST_NUM            = 16,
    NOTFOUND_BUFFERED_BATCH_HEAD        = 17,
    INSUFFICIENT_USER_BALANCE           = 18,
    INTERR_COULDNT_FOUND_BUFFBATCH      = 19,
    INVALID_BATCHID_GIVEN               = 20,
    ERROR_CREATING_SMSLOGFILE           = 21,
    ERROR_WHEN_TRYING_TO_BLACKLIST      = 22,
    ERROR_ON_GETTING_CONTACTS           = 23,
    ERROR_NO_CONTACT_TO_DELETE          = 24,
    RECORD_ALREADY_EXISTS               = 25,
    RECORD_DOES_NOT_EXISTS              = 26,
    RECORD_CHANGE_FROM_DIFF_SESSION     = 27,
    PBOOK_CONTACT_CELL_EMPTY            = 28,
    PBOOK_CONTACT_NAME_EMPTY            = 29,
    PBOOK_INVLD_CELL                    = 30,
    PBOOKGRP_NO_GROUP_PRVD_TO_DEL       = 31,
    ACCSETT_EMPTY_SETTINGS              = 32,
    INVALID_IMPORT_FILE                 = 33,
    INSUFFICIENT_INVLD_PARAMETER_DATA   = 34,
    ERROR_IMPORTING_CONTACTS            = 35,
    INS_UPD_DUPLICATE_CELL_FOUND        = 36,
    NOT_ENOUGH_CREDITS_FOR_HLR_QUERY    = 37,
    ERROR_WHEN_TRYING_SUBMIT_USERHLR    = 38,
    API_TOKEN_NOT_PROVIDED              = 39,
    API_TOKEN_MISMATCH                  = 40,
    INVALID_SCHEDULED_SENDOUT_DATE      = 41,
    SMSIDS_PARAMETER_INVALID            = 42,
    NO_SUBMITTED_SMS_FOUND              = 43,
    INVALID_API_TOKEN                   = 44,
    VOUCHER_FROM_DIFFERENT_DOMAIN       = 45,
    VOUCHER_NOT_FOUND_OR_NON_FREE       = 46,
    VOUCHER_AMOUNT_CREDIT_FAILED        = 47,
    ERROR_UPDATING_CHARGED_VOUCHER      = 48,
    ERROR_DATA_NOT_FOUND                = 49,
    APITOKEN_USR_BELONGS_OTHER_MASTER   = 50,
    SUBACCOUNT_ALREADY_ASSIGNED         = 51,
    SENDERID_TOO_SHORT                  = 52,
    ERROR_CREATING_FILE                 = 53,
    IM_TEXT_EMPTY                       = 54,
    IM_TEXT_LONGER_THAN_EXPECTED        = 55,
    IM_SENDERID_NOT_APPROVED            = 56,
    IM_IMAGE_INVALID                    = 57,
    IM_ACTION_INVALID                   = 58,
    EMPTY_OR_INVALID_PARAMETERS         = 59,
    DATA_VERIFICATION_ERROR             = 60,
    SENDERID_INJ_NUMERIC_DETECTED       = 61,
    SMSFORM_NO_VALUETOKEN_FOUND         = 62,
    SMSFORM_NO_FORM_DATA_FOUND          = 63,
}
```

SMSErrorType

tabid
SMSErrType

```
public enum SMS_INGRENTIENT_TYPES
{
    SENDERID        = 1,
    TEXT            = 2,
    DESTINATION_NUM = 3,
    OTHER           = 4,
}
```

2.3 Send Viber messages

Contents

2.3.1 Description

Viber endpoint makes the IM sendout convenient offering bulk messages transmission with a SMS fallback capability

2.3.2 Endpoint URL

```
https://sms.liveall.eu/apiext/Sendout/SendIM
```

2.3.3 curl example

```
curl --location --request POST 'https://sms.liveall.eu/apiext/Sendout/SendIM' \
--header 'Content-Type: application/json' \
--data-raw '{
  "apitoken": "7ace3e49cae13ae4f5ccb8a6a8a0d6a8fe120aa82ae46ad6ee4c9d8",
  "senderid": "Liveall.eu",
  "sms_text": "my SMS text 2",
  "pricecat": 1,
  "im_type": "IM_SMSFB",
  "im_sender_name": "my viber sender name",
  "im_text": "1546 - This is a test Viber msg",
  "im_image_urls": [ "https://www.liveall.eu/sites/liveall.eu/files/lall-logo-by3.png" ],
  "im_actions": [{ "caption": "Visit Us", "url": "https://www.liveall.eu" }],
  "destinations": [
    "306912345678",
    "306923456789"
  ]
}
'
```

2.3.4 JSON Object variables

apitoken
: `string` a unique hash code for each account that authorizes each web request. That code you can find it on your account's page

im_type
: `string` there are 2 options. **IM** for excplicit Viber sendout and **IM_SMSFB** in case you want to send a Viber message and if that number is invalid, then an SMS will be sent with an alternative body text (fallback option)

im_sender_name
: `string` the sender name of the Viber message (this will be defined and be enabled by us)

im_text
: `string` The Viber message (max 1000 characters)

destinations
: `array of string` an array of string that holds the cell numbers

im_image_urls
: `(optional) - array of string` an array of strings that holds the images to be included with Viber message

(WARNING!) - only one image is permited

im_actions

(optional) - `array of object` an array of objects with actions to be included on message. Every object has 2 properties.

i) caption: which is the text that will be displayed in action's button and

ii) url: that holds the url of the action (when the recipient taps the action's button he will be redirected to this ur - opened in phone's browser).

(WARNING!) - Only one action is also permited for Viber

senderid

(optional) - `string` if you defined **IM_SMSFB** as im_type, then you must set this option for SMS sender name

pricecat

(optional) - `integer` (same as above) it is the price category for SMS. 0 or nothing for normal and 1 for low cost

sms_text

(optional) - `string` (same as above) the text of the fallback SMS

2.3.5 JSON object example (No SMS fallback)

```
{
  "apitoken": "7ace3e49cae13ae4f5ccb8a6a8a0d6a8fe120aa82ae46ad6ee4c9d8",
  "im_type": "IM",
  "im_sender_name": "my viber sender name",
  "im_text": "1546 - This is a test Viber msg",
  "im_image_urls": [ "https://www.liveall.eu/sites/liveall.eu/files/lall-logo-by3.png" ],
  "im_actions": [{ "caption": "Visit Us", "url": "https://www.liveall.eu" }],
  "destinations": [
    "306912345678",
    "306923456789"
  ]
}
```

2.3.6 JSON object example (With SMS fallback)

```
{
  "apitoken": "7ace3e49cae13ae4f5ccb8a6a8a0d6a8fe120aa82ae46ad6ee4c9d8",
  "senderid": "Liveall.eu",
  "sms_text": "my SMS text 2",
  "pricecat": 1,
  "im_type": "IM_SMSFB",
```

(continues on next page)

(continued from previous page)

```
  "im_sender_name": "my viber sender name",
  "im_text": "1546 - This is a test Viber msg",
  "im_image_urls": [ "https://www.liveall.eu/sites/liveall.eu/files/lall-logo-by3.png" ],
  "im_actions": [{ "caption": "Visit Us", "url": "https://www.liveall.eu" }],
  "destinations": [
    "306912345678",
    "306923456789"
  ]
}
```

2.3.7 Error Response

In case of error, we got something like the following:

```
{
    "success": false,
    "OperationErrors": [
        {
            "errorCode": 56,
            "errorMessage": "Sender id for IM is not approved",
            "SMSErrorType": 4,
            "valueOfError": ""
        }
    ],
    "SubmissionID": 0,
    "data": null
}
```

2.3.8 Successful Response

```
{
    "success": true,
    "OperationErrors": null,
    "SubmissionID": 0,
    "data": [
        11271180,
        11271181
    ]
}
```

2.3.9 Response properties

Name	Description
success	when false, then no message sent and the whole request is considered failed
OperationErrors	when success is false, we get an array of objects with errors. Each object has 3 properties: **errorCode**: the error code (integer) of the error, **errorMessage**: the descriptive text of the error and **valueOfError**: the value that caused the error (for debugging purposes)
data	in case of success, web-service is returning an array ob objects - one for each destination, having 2 properties: **destination**: the cell's number and **smsid**: the unique id of the SMS

CHAPTER

THREE

OLD API (OBSOLETE)

Contents

3.1 1. Send SMS

3.1.1 1.1 Description

We maintain the old http api for backwards compatibility (there are no improvements on this api), because Liveall.eu has an ecosystem of 3rd party applications-sites that may not switched to our new API yet.

Note: All newcomers to the service cannot use that API. They **must** use newer versions of API

3.1.2 1.2 Endpoint URL

```
https://www.liveall.eu/webservice/sms/sendSMSHTTP.php

OR

https://www.liveall.eu/webservice/sms/sendSMSHTTPUTF8.php
```

3.1.3 1.3 curl example

```
curl --location --request GET 'https://www.liveall.eu/webservice/sms/sendSMSHTTP.php' \
  --header 'Content-Type: application/x-www-form-urlencoded' \
  --data-urlencode 'username=myusername' \
  --data-urlencode 'password=mypass' \
  --data-urlencode 'destination=306912345678' \
  --data-urlencode 'sender=mySender' \
  --data-urlencode 'message=This is a test message from me!'
```

3.1.4 1.4 Variables

username
: `string` the username

password
: `string` the password

destination
: `string` the destination cell number (must **not contain** leading characters + or `00`). If you want to send to multiple destinations, then separate numbers with the character `.`

sender
: `string` the sender name of the SMS. There is a limit to 11 characters (latin characters).

message
: `string` the SMS text

pricecat
: `(optional) - unsigned integer` by defining this parameter you can send with low-cost (if you set 1 to it) instead of sending with the default normal cost (where applicable)

3.1.5 1.5 Error Response

In case of error, we get one of the following responses:

```
Error: 1001 - No username given.
Error: 1002 - No password given.
Error: 1003 - No destination number given.
Error: 1004 - Unknown destination error.
Error: 1005 - Invalid destination number.
Error: 1006 - Alphanumeric sender address is longer than accepted.
Error: 1007 - Numeric sender address is longer than accepted.
```

(continues on next page)

(continued from previous page)

```
Error: 1008 - No sender name or number given.
Error: 1009 - Message contains invalid character.
Error: 1010 - Error sending SMS - Gateway call error.
Error: 1011 - Greek numbers must have 12 digits (including country code).
Error: 1011 - There is no SMS text given.
Error: 1012 - User authendication failure. Username and/or password mismatch.
Error: 1013 - Not available credits for user.
Error: 1014 - Given user id mismatch.
Error: 1015 - Unknown error with sender.
Error: 1016 - Not enough alphanumeric characters on sender address. Required
Error: 1017 - Destination number(s) not supported.
Error: 1018 - Not enough credits when sending sms to that destination.
Error: 1019 - Error inserting row on transactions table. Aborting
Error: 1020 - Error With sms charge on this destination.
Error: 1021 - Invalid characters on sender address.
Status: 1022 - Awaiting cost confirmation. SMS did't sent yet.
Status: 1023 - Not enough credits left for SMS postage.
Status: 1024 - Error validating Sms.
Status: 1025 - Error while checking balance of user.
Status: 1026 - Batch of SMS queued and is about to be transmited.
Error: 1027 - No SMS ID given.
Error: 1028 - No valid parameters given.
Error: 1029 - The allowed length of sms is no more than 612 characters (4 SMS).
Error: 1031 - The SMS Service is temporary unavailable. Try again in a few minutes.
Error: 1032 - Error submiting SMS. Please try again a bit more later.
Error: 1033 - Not enough balance on your account to make HLR lookup. Please buy credits.
Error: 1036 - The sender name you have provided is not allowed.
Error: 1038 - User is disabled
Error: 1039 - HLR, Invalid number(s) provided
Error: 1040 - No HLR ID given.
Error: 1041 - DBID for HLR query not found
Error: 1042 - Api token not provided
Error: 1043 - Internal error
```

3.1.6 1.6 Successful Response

On a successful SMS submit, you get the following result:

```
OK ID:<SMS_HTTP_request_ID>
```

where `SMS_HTTP_request_ID` is the SMS id of your SMS web-request

Example results

`OK ID:1234` when sending to a single cell number, or `OK ID:1234|OK ID:1235|OK ID:1236` when sending to more than one destination.

3.2 2. Check the status of SMS

3.2.1 2.1 Description

By calling the above end-point you can check the status of SMS previously sent by our platform

3.2.2 2.2 Endpoint URL

```
https://www.liveall.eu/webservice/sms/getSMSStatus.php
```

3.2.3 2.3 curl example

```
curl --location --request GET 'https://www.liveall.eu/webservice/sms/getSMSStatus.php' \
  --header 'Content-Type: application/x-www-form-urlencoded' \
  --data-urlencode 'username=myusername' \
  --data-urlencode 'password=mypass' \
  --data-urlencode 'SMSId=1111'
```

3.2.4 2.4 Variables

username
: `string` the username

password
: `string` the password

SMSId
: `unsigned integer` the SMS id we have been returned by web-service, on message's submission

3.2.5 2.5 Error Response

Possible results for above end-point are, in case of error:

```
Error: 1012 - User authendication failure. Username and/or password mismatch.
Error: 1027 - No SMS ID given.
Error: 1028 - No valid parameters given.
```

3.2.6 2.6 Successful Response

On a successful response we get the following result:

```
<SMSId>:<Submtited On>:<Destination number>:<Delivered On>:<Status number>:<Quantity of␣
↪SMS>:<Charge amount>
```

Example result

```
2345:20101212152514:306912345678:20101212152519:2048:1:0.057
```

3.2.7 2.7 Response properties

Name	Description
SMSId [Integer]	the SMS ID
Submited On [String]	datetime of SMS submit. Date format is: YYYYMMDDHHmmSS [YYYY:year, MM:month 00~12, DD:day of month 00~31, HH:hour 00~24, mm:minutes 00~59, SS:seconds 00~59]
Destination number [String]	the phone number
Delivered On [String]	datetime of last SMS status. Date format is: YYYYMMDDHHmmSS [YYYY:year, MM:month 00~12, DD:day of month 00~31, HH:hour 00~24, mm:minutes 00~59, SS:seconds 00~59]
Status number [Integer]	the status code. *2.8 Possible SMS statuses*
Quantity of SMS [Integer]	the quantity of SMS needed to send the message
Charge amount [float]	the charged amount

3.2.8 2.8 Possible SMS statuses

```
1: Queued
2: Queued on SMSC
4: Waiting Validation
8: Unknown subscriber
16: Temporary unavailable
32: Pending
64: Undelivered
128: Expired
256: Non-Delivered to SMSC
512: Error
1024: Unknown error
2048: Delivered
16384: HLR Sent
32768: HLR Completed
```

3.3 3. Check the current balance of messaging account

3.3.1 3.1 Description

This endpoint fetches the current balance of the account

3.3.2 3.2 Endpoint URL

```
https://www.liveall.eu/webservice/sms/getAccountBalance.php
```

3.3.3 3.3 curl example

```
curl --location --request GET 'https://www.liveall.eu/webservice/sms/getAccountBalance.
↪php' \
  --header 'Content-Type: application/x-www-form-urlencoded' \
  --data-urlencode 'username=myusername' \
  --data-urlencode 'password=mypass' \
  --data-urlencode 'countryprefix=30'
```

3.3.4 3.4 Variables

username
: `string` the username

password
: `string` the password

countryprefix
: `(optional) string` an optional country code. If you provide that, you will get the available SMS count, based on the price of the provided country (normal & low cost)

3.3.5 3.5 Error Response

Possible results for above end-point are, in case of error:

```
Error: 1001 - No username given.
Error: 1002 - No password given.
```

3.3.6 3.6 Successful Response

On a successful response we get the following result:

```
Status: 1000 - Balance:BalanceInEuro|SmsRemainCount:RemainingSMSCount
```

where:

BalanceInEuro	is the request's error code as shown below
RemainingSMSCount	is the error message, describing the problem with the request

CHAPTER

FOUR

SMS FROM .NET LIBRARY

Contents

The target framework of the library is the .Net Framework 4.8.

Note: Download the library from the following link: Terracom.Liveall.ConnectorLib

[Terracom.Liveall.ConnectorLib.Connector] class is used to send your messages by calling the endpoint of the HTTP API, as described here: *Send SMS*

In order to use it, after downloading it, extract the file contents in your project and then add the 2 DLLs (Terracom.Liveall.ConnectorLib.dll & Newtonsoft.Json.dll) as references

See also:

To start with the class first you have to create an object with the new operator and invoke the Init() method by supplying it with the API access token and the sender name:

```
Connector sms_connector = new Connector();
sms_connector.Init("YOUR_ACCOUNT_API_TOKEN", "SenderName");
```

Then we use either **SendSms()** or **SendSmsAsync()** to send in a synchronous or asynchronous way accordingly. There are also other helper methods we introduce below

4.1 Connector.SendSmsAsync()

```
//-------------------------------------------------------------------------------------
/// FUNCTION PROTOTYPE
public async Task<Models.OP_RESULT> SendSmsAsync(Models.HttpApiJSMSParts[] messages, int␣
↪price_category = 0, DateTime? send_on = null);
//-------------------------------------------------------------------------------------
/// USAGE EXAMPLE
#region ~~~~~~~~~~~~~~~~~~~~~~~~ TEST SMS SENDOUT ASYNC
Models.OP_RESULT res = null;
Task.Run(async () =>
{
res = await sms_connector.SendSmsAsync(
   new Models.HttpApiJSMSParts[] {
      new Models.HttpApiJSMSParts {
      destination = "306901234567",
      message = "Test message (async) - English.    ()"
   }
});
}).Wait();

if (res.ErrorCode != SMS_SERVICE_ERROR_CODES.NO_ERROR)
   Console.WriteLine("Problem sending SMS. Error message: {0}", res.ErrorMessage);
else
   Console.WriteLine("SMS sent sucessfully! SMSId for 1st sent message: {0}", res.
↪SubmitInfo.FirstOrDefault().smsid);

#endregion
//-------------------------------------------------------------------------------------
```

This is the async version of the SMS sendout method. Below are the parameters and the result type

Parameters

messages
`Array` of *Models.HttpApiJSMSParts* that contains every destination/message pair to be sent

price_category
`(optional) - int` it is the price category to use. Default value: 0. 1 for low-cost where applicable

send_on
`(optional) - DateTime?` It is the DateTime for the SMS batch to be sent

Return value

Data type: *Models.OP_RESULT*

4.2 Connector.SendSms()

```
//---------------------------------------------------------------------------------------
/// FUNCTION PROTOTYPE
public Models.OP_RESULT SendSms(Models.HttpApiJSMSParts[] messages, int price_category =␣
↪0, DateTime? send_on = null);
//---------------------------------------------------------------------------------------
/// USAGE EXAMPLE
#region ~~~~~~~~~~~~~~~~~~~~~~~~ SYNC SMS SENDOUT
Models.OP_RESULT res1 = sms_connector.SendSms(
   new Models.HttpApiJSMSParts[] {
      new Models.HttpApiJSMSParts {
      destination = "306901234567",
      message = "Test message (sync) - English.    ()"
   }
});

if (res1.ErrorCode != SMS_SERVICE_ERROR_CODES.NO_ERROR)
   Console.WriteLine("Problem sending SMS. Error message: {0}", res1.ErrorMessage);
else
   Console.WriteLine("SMS sent sucessfully! SMSId for 1st sent message: {0}", res1.
↪SubmitInfo.FirstOrDefault().smsid);

#endregion
//---------------------------------------------------------------------------------------
```

This is the sync version of the SMS sendout method. Below are the parameters and the result type

Parameters

messages
`Array` of *Models.HttpApiJSMSParts* that contains every destination/message pair to be sent

price_category
`(optional) - int` it is the price category to use. Default value: 0. 1 for low-cost where applicable

send_on
`(optional) - DateTime?` It is the DateTime for the SMS batch to be sent

Return value

Data type: *Models.OP_RESULT*

4.3 Connector.GetSMSHistoryAsync()

```
//-------------------------------------------------------------------------------------------
/// FUNCTION PROTOTYPE
public async Task<Models.OP_RESULT_SMSLOG> GetSMSHistoryAsync(Models.SMSLogParameters␣
↪parameters);
//-------------------------------------------------------------------------------------------
/// USAGE EXAMPLE
#region ~~~~~~~~~~~~~~~~~~~~~~~~ GET SMS HISTORY OF A SPECIFIC DATE
Models.OP_RESULT_SMSLOG res = null;
Task.Run(async () => {
   res = await sms_connector.GetSMSHistoryAsync(new Models.SMSLogParameters()
   {
      submit_date = "20200402",
      //timezone_offset = 2,
      //sms_id = 47680777,
   });
}).Wait();

if (res.ErrorCode == SMS_SERVICE_ERROR_CODES.NO_ERROR)
{
   Console.WriteLine("Results\r\n\r\n" +
      "Sms id\t\tBatch id\tSender id\tDestination\tStatus DT\tStatus\t\tQty\t\tMsg␣
↪charge\tIM Status\r\n" +

↪"=============================================================================================================
↪");

   foreach (var line in res.SMSLogRows)
   {
      Console.WriteLine($"{line.SMS_ID}\t{line.BatchID}\t\t{line.Sender_ID}\t{line.
↪Destination}\t{line.LastStatusUnixDatetime}\t" +
         $"{line.StatusStr}\t{line.SMS_Qty}\t\t{line.MessageCharge}\t\t{line.
↪InstantMessageStatusStr}");
   }
}
else
{
   Console.WriteLine($"Problem while trying to fetch SMS log data: {res.ErrorCode} -
↪{res.ErrorMessage}");
}

#endregion
//-------------------------------------------------------------------------------------------
```

Fetches the SMS history of a specific date. For more info see at *Extract message log for a date*

Parameters

parameters
Models.SMSLogParameters it contains all the available properties as parameters

Return value

Data type: *Models.OP_RESULT_SMSLOG*

4.4 Connector.GetSMSStatusAsync()

```
//-----------------------------------------------------------------------------------------
/// FUNCTION PROTOTYPE
public async Task<Models.OP_RESULT_SMSSTAT> GetSMSStatusAsync(uint[] sms_ids);
//-----------------------------------------------------------------------------------------
/// USAGE EXAMPLE
#region ~~~~~~~~~~~~~~~~~~~~~~~~ GET SENT SMS STATUS BY ID
Models.OP_RESULT_SMSSTAT res = null;
Task.Run(async () => {
   res = await sms_connector.GetSMSStatusAsync(new uint[] { 99999998, 99999999 });
}).Wait();

if (res.ErrorCode == SMS_SERVICE_ERROR_CODES.NO_ERROR)
{
   foreach (var sms_stat in res.StatusInfo)
   {
      Console.WriteLine($"sms_id: {sms_stat.sms_id}, recipient: {sms_stat.recipient}, " +
         $"last_status_time: {sms_stat.last_status_time}, status_code: {sms_stat.status_
↪code}, " +
         $"ststus_txt: {sms_stat.status_txt}");
   }
}
else
{
   Console.WriteLine($"Failed to get SMS status: {res.ErrorCode} - {res.ErrorMessage}");
}

#endregion
//-----------------------------------------------------------------------------------------
```

Gets the status of sent message(s) providing their sms_id(s) in array

Parameters

sms_ids
`Array on uint` an array of the SMS IDs to be looked-up

Return value

Data type: *Models.OP_RESULT_SMSSTAT*

4.5 Connector.GetAccountBalanceAsync()

```
//-------------------------------------------------------------------------------------
/// FUNCTION PROTOTYPE
public async Task<Models.OP_RESULT_ACCOUNT_INFO> GetAccountBalanceAsync(string␣
↪countryprefix = null);
//-------------------------------------------------------------------------------------
/// USAGE EXAMPLE
#region ~~~~~~~~~~~~~~~~~~~~~~ GET ACCOUNT BALANCE
Models.OP_RESULT_ACCOUNT_INFO res = null;
Task.Run(async () => {
   res = await sms_connector.GetAccountBalanceAsync("30");
}).Wait();

if (res.ErrorCode == SMS_SERVICE_ERROR_CODES.NO_ERROR)
   Console.WriteLine($"Balance: {res.AccountBalance.Balance}, SMS balance: {res.
↪AccountBalance.SmsRemainCount}");
else
   Console.WriteLine($"Problem when trying to get account info: {res.ErrorCode} - {res.
↪ErrorMessage}");
#endregion
//-------------------------------------------------------------------------------------
```

Gets various info about the account - currently this returns the current balance and the remaining SMS count
GetAccountBalanceAsync() has an optional parameter (countryprefix). When this is provided, it can calculate the remaining SMS count - for the provided country, otherwise only the balance is returned.
For example if you provide 30, it will return the remaining SMS for Greece

Parameters

countryprefix
: `string` this is the country prefix to calculate the remaining SMS for the specified country

Return value

Data type: *Models.OP_RESULT_ACCOUNT_INFO*

4.6 Send SMS with VB.NET

Library can also be used by VB.NET. There is an example below that demonstrates the usage

```
Imports Terracom.Liveall.ConnectorLib

Module Module1
   Sub Main()
         Dim sms_connector As New Connector()
         Dim res As Models.OP_RESULT
         Dim submit_info(1) As Models.HttpApiJSMSParts
```

(continues on next page)

(continued from previous page)

```
        Dim si As New Models.HttpApiJSMSParts

        si.destination = "306912345678"
        si.message = "Test message - English.    ()"
        submit_info(0) = si

        sms_connector.Init("MY_VERY_SECRET_TOKEN", "mySenderID")
        res = sms_connector.SendSms(submit_info, 0)
        If res.ErrorCode <> SMS_SERVICE_ERROR_CODES.NO_ERROR Then
           Console.WriteLine("Problem sending SMS. Error message: {0}", res.
↪ErrorMessage)
        Else
           Console.WriteLine("SMS sent sucessfully! SMSId for 1st sent message: {0}",␣
↪res.SubmitInfo.FirstOrDefault().smsid)
        End If
   End Sub
End Module
```

4.7 APPENDIX - Classes

4.7.1 Models.OP_RESULT_BASE

```
public class OP_RESULT_BASE
{
   public OP_RESULT_BASE()
   {
      ErrorCode = SMS_SERVICE_ERROR_CODES.NO_ERROR;
      ErrorMessage = null;
   }

   public SMS_SERVICE_ERROR_CODES ErrorCode { get; set; }
   public string ErrorMessage { get; set; }
}
```

4.7.2 Models.OP_RESULT

```
public class OP_RESULT : OP_RESULT_BASE
{
   public OP_RESULT()
      :base()
   {
      SubmitInfo = null;
   }

   public SubmissionInfo[] SubmitInfo { get; set; }
}
```

Models.SubmissionInfo

```
public class SubmissionInfo
{
   public string destination { get; set; }
   public uint smsid { get; set; }
}
```

4.7.3 Models.OP_RESULT_SMSLOG

```
public class OP_RESULT_SMSLOG : OP_RESULT_BASE
{
   public OP_RESULT_SMSLOG()
      : base()
   {
   }

   public Models.SMSLogRow[] SMSLogRows { get; set; }
}
```

Models.SMSLogRow

```
public class SMSLogRow
{
   public uint SMS_ID { get; set; }
   public uint BatchID { get; set; }
   public string Sender_ID { get; set; }
   public string Destination { get; set; }
   public uint LastStatusUnixDatetime { get; set; }
   public DateTime LastStatusDatetime { get; set; }
   public string StatusStr { get; set; }
   public int SMS_Qty { get; set; }
   public double MessageCharge { get; set; }
   public string InstantMessageStatusStr { get; set; }
}
```

4.7.4 Models.OP_RESULT_SMSSTAT

```
public class OP_RESULT_SMSSTAT : OP_RESULT_BASE
{
   public OP_RESULT_SMSSTAT()
      :base()
   {
   }

   public SmsStatusInfo[] StatusInfo { get; set; }
}
```

Models.SmsStatusInfo

```
public class SmsStatusInfo
{
   public uint sms_id { get; set; }
   public DateTime submitted_on { get; set; }
   public DateTime last_status_time { get; set; }
   public string recipient { get; set; }
   public DLR_CODES status_code { get; set; }
   public string status_txt { get; set; }
   public int sms_qty { get; set; }
   public double charge_amount { get; set; }
}
```

4.7.5 Models.OP_RESULT_ACCOUNT_INFO

```
public class OP_RESULT_ACCOUNT_INFO : OP_RESULT_BASE
{
   public OP_RESULT_ACCOUNT_INFO()
      :base()
   {
   }

   public AccountBalanceInfo AccountBalance { get; set; }
}
```

Models.AccountBalanceInfo

```
public class AccountBalanceInfo
{
   public double Balance { get; set; }
   public int SmsRemainCount { get; set; }
   public int LCSmsRemainCount { get; set; }
}
```

4.7.6 Models.HttpApiJSMSParts

```
public class HttpApiJSMSParts
{
   public string destination { get; set; }
   public string message { get; set; }
}
```

4.7.7 Models.SMSLogParameters

```
public class SMSLogParameters
{
   public string submit_date { get; set; }
   public int? timezone_offset { get; set; }
   public string senderid { get; set; }
   public string destination { get; set; }
   public uint sms_id { get; set; }
   public uint batch_id { get; set; }
   public uint gt_sms_id { get; set; }
}
```

4.7.8 Error Codes

```
public enum SMS_SERVICE_ERROR_CODES
{
   NO_ERROR                          = 0,
   EMPTY_SENDERID                    = 1,
   INVALID_SENDERID                  = 2,
   UNAUTHORIZED_NUM_SENDER_ID        = 3,
   ALPHA_SENDERID_TOO_LONG           = 4,
   NUM_SENDERID_TOO_LONG             = 5,
   INTERR_NO_SMS_TYPE_PROV           = 6,
   INTERR_NO_SMS_TEXT                = 7,
   INTERNAL_ERROR                    = 8,
   ILLEGAL_SENDERID                  = 9,
   SMS_TEXT_EMPTY                    = 10,
   SMS_TEXT_LEN_TOO_LONG             = 11,
   NO_DESTINATION_NUMBERS_PROVIDED   = 12,
   INVALID_DESTINATION_NUMBER        = 13,
   INVALID_GREEK_DEST_NUM            = 14,
   INVALID_CYPR_DEST_NUM             = 15,
   INVALID_ITALIAN_DEST_NUM          = 16,
   NOTFOUND_BUFFERED_BATCH_HEAD      = 17,
   INSUFFICIENT_USER_BALANCE         = 18,
   INTERR_COULDNT_FOUND_BUFFBATCH    = 19,
   INVALID_BATCHID_GIVEN             = 20,
   ERROR_CREATING_SMSLOGFILE         = 21,
   ERROR_WHEN_TRYING_TO_BLACKLIST    = 22,
   ERROR_ON_GETTING_CONTACTS         = 23,
   ERROR_NO_CONTACT_TO_DELETE        = 24,
   RECORD_ALREADY_EXISTS             = 25,
   RECORD_DOES_NOT_EXISTS            = 26,
   RECORD_CHANGE_FROM_DIFF_SESSION   = 27,
   PBOOK_CONTACT_CELL_EMPTY          = 28,
   PBOOK_CONTACT_NAME_EMPTY          = 29,
   PBOOK_INVLD_CELL                  = 30,
   PBOOKGRP_NO_GROUP_PRVD_TO_DEL     = 31,
   ACCSETT_EMPTY_SETTINGS            = 32,
   INVALID_IMPORT_FILE               = 33,
```

(continues on next page)

(continued from previous page)

```
    INSUFFICIENT_INVLD_PARAMETER_DATA   = 34,
    ERROR_IMPORTING_CONTACTS            = 35,
    INS_UPD_DUPLICATE_CELL_FOUND        = 36,
    NOT_ENOUGH_CREDITS_FOR_HLR_QUERY    = 37,
    ERROR_WHEN_TRYING_SUBMIT_USERHLR    = 38,
    API_TOKEN_NOT_PROVIDED              = 39,
    API_TOKEN_MISMATCH                  = 40,
    INVALID_SCHEDULED_SENDOUT_DATE      = 41,
    SMSIDS_PARAMETER_INVALID            = 42,
    NO_SUBMITTED_SMS_FOUND              = 43,
    INVALID_API_TOKEN                   = 44,
    VOUCHER_FROM_DIFFERENT_DOMAIN       = 45,
    VOUCHER_NOT_FOUND_OR_NON_FREE       = 46,
    VOUCHER_AMOUNT_CREDIT_FAILED        = 47,
    ERROR_UPDATING_CHARGED_VOUCHER      = 48,
    ERROR_DATA_NOT_FOUND                = 49,
    APITOKEN_USR_BELONGS_OTHER_MASTER   = 50,
    SUBACCOUNT_ALREADY_ASSIGNED         = 51,
    SENDERID_TOO_SHORT                  = 52,
    ERROR_CREATING_FILE                 = 53,
    IM_TEXT_EMPTY                       = 54,
    IM_TEXT_LONGER_THAN_EXPECTED        = 55,
    IM_SENDERID_NOT_APPROVED            = 56,
    IM_IMAGE_INVALID                    = 57,
    IM_ACTION_INVALID                   = 58,
    EMPTY_OR_INVALID_PARAMETERS         = 59,
    DATA_VERIFICATION_ERROR             = 60,
    SENDERID_INJ_NUMERIC_DETECTED       = 61,
    SMSFORM_NO_VALUETOKEN_FOUND         = 62,
    SMSFORM_NO_FORM_DATA_FOUND          = 63,
}
```