OMNI API DOCUMENTATION V1.2

13rd Km, Athens-Lamia N.Road, Katsantoni & Olympias 2 Metamorfosi - Greece,
Postal Code: 14452, Tel.: +30 211 11 44 111, Fax: +30 210 68 12 386, Website: www.yuboto.com,
Email: support@yuboto.com

Omni API Documentation

Contents

INTRODUCTION

1. Getting Started with OMNI API

Before making use of our API, please read the following carefully to get you started.

The following document is addressed to programmers who want to incorporate the OMNI service to their systems.

Here, you will find information about reception via HTTP POST.

OMNI channel is messaging solution that enables you to communicate with your users through various messaging channels.

Leverage your communication options with OMNI messaging and engage your users over various channels:

- SMS
- VIBER

To start using the OMNI service, follow the steps below:

- Register at https://services.yuboto.com/yuniverse.
- Request an OMNI API Key from our support team at support@yuboto.com.
- Purchase the required credits to start sending messages.

If you need information about message charges or more information about Viber messages, please contact us at: sales@yuboto.com

If you need technical information, please contact us at: support@yuboto.com

You can also call us at +30 211 11 44 111 working days from 9.00 am to 6.00 pm

The use of Yuboto platform is subject to the terms of use and privacy statement you may find at https://services.yuboto.com/yuniverse

2. Base URL

Submit all requests to the base URL. All the requests are submitted through **HTTP POST** method.

Base URL: https://services.yuboto.com/

Service Endpoint: /omni/v1

Using the OMNI service, you can perform the following actions:

- **Send:** Send SMS or Viber messages (Paragraph 1)
- **Dlr:** Retrieve the status of previously sent messages (Paragraph 2)
- **Cost:** Request the cost of SMS and Viber messages (Paragraph 3 and Paragraph 4)
- **Balance:** Request your account's balance (Paragraph 5)
- **Cancel:** Cancel scheduled messages (Paragraph 6)

3. Content Type

The message request must be in JSON format. Because of that HTTP request must have HTTP header "Content-Type" that must have value "application/json; charset=utf-8".

4. Authentication

All API calls require authentication. This is essential for the API to identify which user is making the call so that appropriate results will be returned, as well as for security reasons.

For this purpose, API uses basic authentication. Authentication data are sent via HTTP header "Authorization".

Steps to construct authorization header:

1. Base64 encode the API Key.
2. Supply an "Authorization" header with content "Basic" followed by the encoded API Key. For example, the Authorization header will be:

```
Authorization: Basic apiKey
```

Request an API Key from our support team at support@yuboto.com. Your API Key depends on the type of sending you wish to make (e.g. an API Key for **only SMS messages**, an API Key for **only Viber messages** or an API Key for **OMNI (SMS and/or Viber) messages**). Please keep your API Key safe to prevent any unauthorized access. Once you obtain your API Key, you will have to use it in every API call you make. The API key must always be specified as a parameter in the query string of the requesting URL.

REFERENCE

1. Send Method (POST)

Description

This method allows you to send text messages to one or multiple recipients simultaneously. The maximum number of recipients you can send at one time is 1000.

URL to web service operation

https://services.yuboto.com/omni/v1/Send

1.1 Method Parameters, Request and Response

Request
The request body is of type SendRequest.

```
public class SendRequest
{
      public string[] phonenumbers { get; set; }
      public int? dateinToSend { get; set; }
      public int? timeinToSend { get; set; }
      public bool dlr { get; set; }
      public string callbackUrl { get; set; }
      public string option1 { get; set; }
      public string option2 { get; set; }
      public SmsObj sms { get; set; }
      public ViberObj viber { get; set; }
}
```

Parameters

The variables used to send text messages (SMS and/or Viber) are:

Variables	Description	Permitted Values	Required
phonenumbers	Refers to the phone number of the recipient or recipients of the text message (use array for multiple recipients).	String/Array. Use country code without + or 00.	Yes
dateinToSend	Indicates the date you wish to send the message. If this is omitted, the message is sent instantly.	Integer. YYYYMMDD YYYY refers to the year MM refers to the month DD refers to the day	No
timeinToSend	Indicates the time you wish to send your message. If this is omitted, the message is sent instantly.	Integer. HHMM HH refers to the hour MM refers to minutes	No

dlr	The flag indicates if delivery receipt request must be sent to customer's application. (Default: false)	Bool	No
callbackUrl	When the message reaches its final state, a call to this url will be performed by Yuboto's system with the message's delivery info. See paragraph 2.3.	String	No***
option1	User defined value that will be included in the call to the provided callback_url.	String	No**
option2	User defined value that will be included in the call to the provided callback_url.	String	No**
sms	This object is required if list of channels contains SMS channel.	SmsObj Object	No*
viber	This object is required if a list of channels contains VIBER channel. Parameters text, buttonCaption + buttonAction and image make Viber Service Message content. There are 4 possible combinations of Viber Service Message content: • text only, • image only, • text + button, • text + button + image	ViberObj Object	No*

** One of two these parameters must always exists.*
***Option1 & Option2 Parameters will be available for retrieve only if you pass dlr:true and a callbackUrl parameter*
****You can add a persistent callback url in your account without sending a callbackURL parameter each time you call the API. Contact with support@yuboto.com in order to set your persistent callbackURL under your account.*

The definition of the SmsObj is the following.

```
public class SmsObj
{
      public string sender { get; set; }
      public string text { get; set; }
      public int validity { get; set; }
      public string typesms { get; set; }
      public bool longsms { get; set; }
      public int priority { get; set; }
```

```
        public TwoFaObj TwoFa { get; set; }
}
```

Variables	Description	Permitted Values	Compulsory
sender	SMS originator ("sender") that will be displayed on mobile device's screen. • Alphanumeric origin, max. 11 characters • Numeric origin, max. 20 characters	String	Yes
Text*	The text of the message. **If two factor authentication is activated TwoFa, is mandatory that '{pin_code}' is included in this string. This placeholder will then be replaced with the generated pin.**	String	Yes
validity	If the SMS is not delivered directly, this variable indicates the amount of seconds for which the message will remain active, before being rejected by the SMSC.	Integer. Min Value: 30 Max Value: 4320 (default)	No
typesms	Indicates the type of message you wish to send.	String. 1. sms (default) 2. flash 3. unicode	No
longsms	Indicates if the message can be over 160 characters. It applies only to standard type SMS.	Bool. 1. false (default) 2. true	No
priority	Indicates which channel has priority when it comes to omni messaging (default value is: 0)	Integer	No
TwoFa	If Two Factor Authentication is needed then provide this object along with other values.	Object	No

The definition of the ViberObj is the following.

```
public class ViberObj
{
      public string sender { get; set; }
```

```
        public string text { get; set; }
        public int validity { get; set; }
        public string expiryText { get; set; }
        public string buttonCaption { get; set; }
        public string buttonAction { get; set; }
        public string image { get; set; }
        public int priority { get; set; }
        public TwoFaObj TwoFa { get; set; }
}
```

Variables	Description	Permitted Values	Compulsory
sender	Viber message originator ("sender") that will be displayed on mobile device's screen. • Alphanumeric origin, max. 20 characters	String	Yes
Text*	The Viber Service Message text. Text length can be up to 1000 characters. VIBER text can be sent alone, without button or image. **If two factor authentication is activated TwoFa, is mandatory that '{pin_code}' is included in this string. This placeholder will then be replaced with the generated pin.**	String	No
validity	If the Viber message is not delivered directly, this variable indicates the amount of seconds for which the message will remain active, before being rejected.	Integer. Min Value: 15 Max Value: 86.400 (default)	No
expiryText	Relevant for iOS version of Viber application (iPhone users only). This is the text that will be displayed if Viber Service Message expires.	String	Yes
buttonCaption*	A textual writing on the button. Maximum length is 30 characters. The VIBER button can be sent only if Viber Service Message contains text.	String	No
buttonAction*	The link of button action.	String	No
Image*	The URL address of image sent to end user. The VIBER image can be sent only alone or together with text and button.	String	No
priority	Indicates which channel has priority when it comes to omni messaging (default value is: 0).	Integer	No

TwoFa	If Two Factor Authentication is needed then provide this object along with other values.	Object	No

***** Parameters text, buttonCaption + buttonAction and image make Viber Service Message content. There are 4 possible combinations of Viber Service Message content:

- text only,
- image only,
- text + button,
- text + button + image

The definition of the TwoFaObj is the following.

```
public class TwoFaObj
{
public int pinLength { get; set; } ----------> accepted values between 4-32
        public string pinType { get; set; } ----------> accepted  values between numeric, alpha, alphanumeric
        public bool isCaseSensitive { get; set; }
        public int expiration { get; set; } ----------> accepted values between 60-600 (in seconds)
}
```

Variables	Description	Type	Compulsory
pinLength	The length of the pin to be generated Min:4 Max: 32	int	Yes
pinType	Accepted values: • ALPHA (PQRST) • ALPHA_ALPHA_LOWER_NUMERIC (Pg3Gh) • ALPHA_NUMERIC (hEQsa) • NUMERICWITHOUTZERO (5443) • NUMERIC (54034)	String	Yes
isCaseSensitive	Whether the pin should be case sensitive.(alpha, alphanumeric) If false, the case sensitivity would not be checked when validating, if true, code for validation needs to be entered exactly as provided.	bool	Yes
expiration	The time the pin will be active.	int	Yes

	Accepted values between 60-600 (in seconds)		

SendRequest Example

```
{
      callbackUrl = "http://test.com",
      dlr = true,
      option1 = "option 1",
      option2 = "option 2",
      //single recipient//
      phonenumbers = "306946XXXXXX",
      //multiple recipients//
      phonenumbers = [ "306936XXXXXX", "306936XXXXXX" ],
      dateinToSend = 20180101,
      timeinToSend = 1000,
      sms = new SmsObj
      {
          sender = "Demo",
          text = " This is a test sms fallback",
          validity = 100,
          typesms = "sms",
          priority = 1
      },
      viber = new ViberObj
      {
          sender = "Demo",
          text = " This is an omni viber message",
          validity = 15,
          image = "https://someurl/banner.jpg",
          buttonAction = "https://someurl/contact",
          buttonCaption = "Contact us",
          expiryText = "This viber message expired",
          priority = 0
      }
      },
      TwoFa = new TwoFaObj
      {
          pinLength = 4,
          pinType = " numeric",
          isCaseSensitive = true,
          expiration = 300

      }

}
```

Response

The response body is of type SendResponse.

```
public class SendResponse
{
```

```
        public int ErrorCode { get; set; }
        public string ErrorMessage { get; set; }
        public List<MessageStatus> Message { get; set; }
}
```

The definition of the MessageStatus is the following.

```
public class MessageStatus
{
      public string id { get; set; }
      public string channel { get; set; }
      public string phonenumber { get; set; }
      public string status { get; set; }
}
```

- **ErrorCode:** The response error code for this call. This will be 0 if successful.
- **ErrorMessage:** The response error message. This will be null if successful.
- **Message:** A list which contains the status of the messages.
    - **id:** The id of message status.
    - **channel:** The channel that the message will be send (SMS or Viber).
    - **phonenumber:** Refers to the phone number of the recipient of the text message.
    - **status:** The status of the message.

The SendResponse defines if an error occurred during the sending of a message. If a message sent successfully then the ErrorCode has the value 0 and the ErrorMessage contains a zero length string. If an error occurred then, consult the error message that appears.

SendResponse Example

```
{
    "ErrorCode":0,
    "ErrorMessage":"",
    "Message":[
        {
            "id":"MessageID1",
            "channel":"sms",
            "phonenumber":"306936XXXXXX",
            "status":"Submitted"
        },
        {
            "id":"MessageID2",
            "channel":"sms",
            "phonenumber":"306936XXXXXXX",
            "status":"Submitted"
        }
    ]
}
```

1.2 Type of SMS Messages

Through Yuboto platform, you can send:

- Simple SMS (up to 160 characters)
- Flash SMS (up to 160 characters)
- Long SMS (more than 160 characters)
- Unicode SMS (up to 70 characters)

A simple SMS includes all the 7bit alphabet characters as defined by the GSM 03.38.
Some 8bit alphabet characters may also be included and sent as a simple SMS. These will count as 2 characters.

These characters are:

CIRCUMFLEX ACCENT	^
LEFT CURLY BRACKET	{
RIGHT CURLY BRACKET	}
REVERSE SOLIDUS (BACKSLASH)	\
LEFT SQUARE BRACKET	[
TILDE	~
RIGHT SQUARE BRACKET	]
VERTICAL BAR	|
EURO SIGN	€

All the other characters included in the 8bit alphabet can only be sent as Unicode characters (SMS 70 characters).

For more information about Unicode characters, you can visit: http://www.unicode.org/charts/.

If you use small case Greek characters (8bit) in a non-Unicode format, then the system will automatically convert them into Capital Greek characters (7bit).

Long SMS is a text message longer than 160 characters. If the user's mobile phone supports it, then the text message will be received as one. Otherwise the message will be divided into multiple messages of 153 characters each (Maximum number of characters 2000).

If you choose to send a long SMS without previously notifying the system, then the system will limit it to 160 characters (simple SMS).

1.3 System Forwarded Callbacks

If callback is false, then Yuboto's system will not send to the client the final state info. When callback is true, Yuboto's system will forward the final state info to the client. The information that the client will receive, it is possible for the user to pass it on dynamically to the system.

The message info will be forwarded through a get request to the callback url. The following parameters will be included in the query string:

- sender – The message sender
- receiver – The destination phone number
- smsid – The unique id that the message has
- status – The status code that the message has
- statusDescription – See paragraph 2.2 for a detailed description of status values
- dlrDate – The date that the message delivered
- channel – The channel with which the message was sent
- option1 – The user defined value that was passed to method 'send'
- option2 – The user defined value that was passed to method 'send'

1.3.1 Status Expired callback

Expired callback will be sent on iOS once the expiry text message is seen. On Android the expired callback will be sent once the user is online again.

2. DLR Method (POST)

Description

Using this method, you can retrieve information on sent text messages and check their status in real-time.

URL to web service operation

https://services.yuboto.com/omni/v1/Dlr

2.1 Method Parameters, Request and Response

Request

The request body is of type DlrRequest.

```
public class DlrRequest
{
      public string id { get; set; }
}
```

Parameters

The variables used to retrieve information on sent text messages and check their status in real-time are:

Variables	Description	Permitted Values	Required
id	The id of message status.	String	Yes

DlrRequest Example

```
{
    id = "54E3B5F5-2CF3-412E-80A6-A324D94500F6”
}
```

Response
The response body is of type DlrResponse.

```
public class DlrResponse
{
      public int ErrorCode { get; set; }
      public string ErrorMessage { get; set; }
      public string id { get; set; }
      public string phonenumber { get; set; }
      public string option1 { get; set; }
      public string option2 { get; set; }
      public List<DlrChannel> dlr { get; set; }
}
```

The definition of the DlrChannel is the following.

```
public class DlrChannel
{
      public string channel { get; set; }
      public int priority { get; set; }
      public string status { get; set; }
      public decimal cost { get; set; }
      public string sender { get; set; }
      public string text { get; set; }
      public DateTime? submitDate { get; set; }
      public DateTime? dlrDate { get; set; }
}
```

- **ErrorCode:** The response error code for this call. This will be 0 if successful.
- **ErrorMessage:** The response error message. This will be null if successful.
- **id:** The id of message status.
- **phonenumber:** Refers to the phone number of the recipient or recipients of the text message.
- **option1:** The value that included in the call to the provided callback_url.
- **option2:** The value that included in the call to the provided callback_url.
- **dlr:** A list with dlr channels and their details.
    - **channel:** The message channel related to DLR request. Possible values are: viber or sms.
    - **priority:** Indicates which channel has priority when it comes to omni messaging (default value is: 0).
    - **status:** The status that the message has.
    - **cost:** The cost of the message.
    - **sender:** The sender of the message.
    - **text:** The text that the message has.
    - **submitDate:** The date the message was sent.
    - **dlrDate:** The date the message was delivered.

The DlrResponse defines if an error occurred during retrieve information on sent text messages and check their status in real-time. If this method called successfully then the ErrorCode has the value 0 and the ErrorMessage contains a zero length string. If an error occurred then, consult the error message that appears.

DlrResponse Example


```
{
    "ErrorCode":0,
    "ErrorMessage":"",
    "id":"54E3B5F5-2CF3-412E-80A6-A324D94500F6",
    "phonenumber":"306936XXXXXXX",
    "option1":"option1 value",
    "option2":"option2 value",
    "dlr":[
        {
            "channel":"viber",
            "priority":0,
            "status":"Not Delivered",
            "cost":1,
            "sender":"Demo",
            "text":"This is a demo viber msg",
            "submitDate":"\/Date(1500550221991)\/",
            "dlrDate":"\/Date(1500550221990)\/"
        },
        {
            "channel":"sms",
            "priority":1,
            "status":"Delivered",
            "cost":1,
            "sender":"Demo",
            "text":"This is a demo sms msg",
            "submitDate":"\/Date(1500550281991)\/",
            "dlrDate":"\/Date(1500550341991)\/"
        }
    ]
}
```

2.2 Status of Messages

The following table shows the possible status of a message (SMS or Viber):

Initial Status	Final Status *
Sent	No
Pending	No
Submitted	No
Buffered	No
Delivered	Yes
Not Delivered ******	Yes
Unknown*****	Yes
Error	Yes
Expired*****	Yes
Failed ***	Yes
Rejected **********	Yes
Scheduled	No
Canceled	Yes
Deleted*****	Yes
Seen	Yes

* Indicates if this is the final status of the message or it is going to change.

** Some of the possible reasons for failure of delivery might be:

i) Invalid telephone number ii) telephone deactivated or switched-off. In the last two cases, the SMSC holds the message for 3 days and before rejecting it, allows you to select shorter time period (the variable validity of SmsObj see par. 2.2)

*** Delivery fails when there are no available Credits to your account.

**** Messages are rejected when the recipient of the SMS or Viber message has an invalid format or when your account or Yuboto platform do not support it.

Failed or Rejected messages are not charged (to your account).

***** These status (final status) cause fallback to your 2nd priority channel. For example, if your priority channel is Viber, at first Yuboto will try to deliver a Viber service message. If the Viber Service Message is undeliverable for whatsoever reason, Yuboto will send an SMS message.

Possible reasons for Viber Service Message non-delivery are:

- Subscriber doesn't have Viber app installed on device
- Subscriber is not reachable within given TTL
- Subscriber has Viber app that does not support Viber Service Messages (e.g. Windows Phone OS version of Viber app)

3. Cost Method (POST)

Description

Through the following method you can request the cost of sending a simple SMS or Viber.

URL to web service operation

https://services.yuboto.com/omni/v1/Cost

Request
The request body is of type CostRequest.

```
public class CostRequest
{
      public string iso2 { get; set; }
      public string phonenumber { get; set; }
      public string channel { get; set; }
}
```

Parameters

The variables used to request the cost of sending a simple SMS to one or multiple recipients are:

Variables	Description	Permitted Values	Required
iso2	The ISO_3166-1_alpha-2 code of the country.	String. 2-letter code	Yes*
phonenumber	Refers to the phone number of the recipient of the text message.	String	Yes*
channel	The channel that the message will be send (SMS or Viber).	String	Yes**

** One of two these parameters must always exists.*

*** In case you have an **omni API Key** (you send SMS and/or Viber messages), you need to specify for which channel you want to learn the cost. If your channel is not omni, then this parameter is not required.*

CostRequest Example

```
{
    iso2 = "gr",
    channel = "sms"
}
```

Response

The response body is of type CostResponse.

```
public class CostResponse
{
      public int ErrorCode { get; set; }
      public string ErrorMessage { get; set; }
      public string channel { get; set; }
      public string type { get; set; }
      public List<CostInfo> costInfo { get; set; }
}
```

The definition of the CostInfo is the following.

```
public class CostInfo
{
      public string networkName { get; set; }
      public decimal cost { get; set; }
}
```

- **ErrorCode:** The response error code for this call. This will be 0 if successful.
- **ErrorMessage:** The response error message. This will be null if successful.
- **channel:** The channel that the message will be send (SMS or Viber).
- **type:** Indicates the type of your cost (e.g. credits or money).
- **costInfo:** A list with all the details about the cost of sending a simple SMS or Viber message to one or multiple recipients.
    - **networkName:** The name of the network (e.g. `VODAFONE - PANAFON Hellenic Telecommunications Company`).
    - **cost:** The cost of sending a simple SMS or Viber message to one or multiple recipients.

The CostResponse defines if an error occurred during request the cost of sending a simple SMS or Viber message to one or multiple recipients. If method called successfully then the ErrorCode has the value 0 and the ErrorMessage contains a zero length string. If an error occurred then, consult the error message that appears.

CostResponse Example

```
{
    "ErrorCode":0,
    "ErrorMessage":"",
    "channel":"sms",
    "type":"credits",
    "costInfo":[
        {
            "networkName":"Network name1",
            "cost":1
        },
        {
            "networkName":"Network name2",
            "cost":1
        }
    ]
}
```

4. Cost Details Method (POST)

Description

Using this method, you can retrieve the cost details of sending a simple SMS or Viber message for a specific iso2. The difference from Cost Method is that this method returns for a specific iso2, the mcc and mnc.

URL to web service operation

https://services.yuboto.com/omni/v1/CostDetails

Request
The request body is of type CostRequest.

```
public class CostRequest
{
      public string iso2 { get; set; }
}
```

Parameters

The variables used to retrieve the cost details of sending a simple SMS or Viber message to one or multiple recipients are:

Variables	Description	Permitted Values	Required
iso2	The ISO_3166-1_alpha-2 code of the country.	String. 2-letter code	Yes

CostRequest Example

```
{
    iso2 = "gr"
}
```

Response
The response body is of type CostDetailsResponse.

```
public class CostDetailsResponse
{
      public int ErrorCode { get; set; }
      public string ErrorMessage { get; set; }
      public string channel { get; set; }
      public string type { get; set; }
      public List<CostInfoDetails> costInfoDetails;
}
```

The definition of the CostInfoDetails is the following.

```
public class CostInfoDetails
{
      public string networkName { get; set; }
      public string mcc { get; set; }
      public string mnc { get; set; }
      public decimal cost { get; set; }
}
```

- **ErrorCode:** The response error code for this call. This will be 0 if successful.
- **ErrorMessage:** The response error message. This will be null if successful.
- **channel: :** The channel that the message will be send (SMS or Viber).
- **type:** Indicates the type of your cost (e.g. credits or money).
- **costInfoDetails:** A list with all the details about the cost info of sending a simple SMS or Viber message to one or multiple recipients.
    - **networkName:** The name of the network (e.g. `VODAFONE - PANAFON Hellenic Telecommunications Company`).
    - **mcc:** The mobile country code (MCC) consists of 3 decimal digits (e.g. `202`).
    - **mnc:** The mobile network code (MNC) consists of 2 or 3 decimal digits (for example: MNC of 001 is not the same as MNC of 01). The first digit of the mobile country code identifies the geographic region as follows (the digits 1 and 8 are not used):
        - 0 - Test networks
        - 2 - Europe
        - 3 - North America and the Caribbean
        - 4 - Asia and the Middle East
        - 5 - Oceania
        - 6 - Africa
        - 7 - South and Central America
        - 9 - Worldwide (Satellite, Air - aboard aircraft, Maritime - aboard ships, Antarctica)
    - **cost:** The cost of sending a simple SMS or Viber message to one or multiple recipients.

The CostDetailsResponse defines if an error occurred during retrieve the cost details of sending a simple SMS or Viber message to one or multiple recipients. If the method called successfully then the ErrorCode has the value 0 and the ErrorMessage contains a zero length string. If an error occurred then, consult the error message that appears.

CostDetailsResponse Example


```
{
    "costInfoDetails":[
        {
            "networkName":"Wind Hellas Telecommunications SA",
            "mcc":"202",
            "mnc":"009",
            "cost":1.00
        },
        {
            "networkName":"Wind Hellas Telecommunications SA",
            "mcc":"202",
            "mnc":"010",
            "cost":1.00
        },
        {
            "networkName":"VODAFONE - PANAFON Hellenic
Telecommunications Company SA",
            "mcc":"202",
            "mnc":"005",
            "cost":1.00
        },
        {
            "networkName":"COSMOTE Mobile Telecommunications SA",
            "mcc":"202",
            "mnc":"001",
            "cost":1.00
        },
        {
            "networkName":"COSMOTE Mobile Telecommunications SA",
            "mcc":"202",
            "mnc":"002",
            "cost":1.00
        }
    ],
    "ErrorCode":0,
    "ErrorMessage":null,
    "channel":"sms",
    "type":"credits"
}
```

5. Balance Method (POST)

Description

Through the following method you can retrieve information on your current balance.

URL to web service operation

https://services.yuboto.com/omni/v1/Balance

Response
The response body is of type BalanceResponse.

```
public class BalanceResponse
{
      public int ErrorCode { get; set; }
      public string ErrorMessage { get; set; }
      public decimal balance { get; set; }
      public decimal balanceLimit { get; set; }
      public string type { get; set; }
}
```

- **ErrorCode:** The response error code for this call. This will be 0 if successful.
- **ErrorMessage:** The response error message. This will be null if successful.
- **balance:** Your current balance in credits.
- **balanceLimit:** To what limit your account can send messaging (default value is: 0).
- **type:** The type of your balance based on user's configuration (e.g. credits or money).

The BalanceResponse defines if an error occurred during retrieve information on your current balance in credits. If the method called successfully then the ErrorCode has the value 0 and the ErrorMessage contains a zero length string. If an error occurred then, consult the error message that appears.

BalanceResponse Example

```
{
    "ErrorCode":0,
    "ErrorMessage":"",
    "balance":100,
    "balanceLimit":0,
    "type":"credits"
}
```

6. Cancel Method (POST)

Description

Through the following method you can cancel a scheduled message, before the scheduled date and time. You are able to cancel the sending of a message up to **three minutes** before the time it is scheduled to send.

URL to web service operation

https://services.yuboto.com/omni/v1/Cancel

Request

The request body is of type CancelRequest.

```
public class CancelRequest
{
      public string id { get; set; }
}
```

Parameters

The variables used to create a user are:

Variables	Description	Permitted Values	Required
id	The id of message status.	String	Yes

CancelRequest Example

```
{
      id = "601756D5-6537-4DC5-BD07-10D95BF1621E"
}
```

Response
The response body is of type CancelResponse.

```
public class CancelResponse
{
      public int ErrorCode { get; set; }
      public string ErrorMessage { get; set; }
      public string channel { get; set; }
      public string id { get; set; }
      public string status { get; set; }
}
```

- **ErrorCode:** The response error code for this call. This will be 0 if successful.
- **ErrorMessage:** The response error message. This will be null if successful.
- **channel:** The channel that the message is scheduled to be send (SMS or Viber).
- **id:** The id of message status.
- **status:** The status of the message.

The CancelResponse defines if an error occurred during the cancellation of a message. If the cancellation of a message be done successfully then the ErrorCode has the value 0 and the ErrorMessage contains a zero length string. If an error occurred then, consult the error message that appears.

CancelResponse Example

```
{
    "ErrorCode":0,
    "ErrorMessage":"",
    "channel":"sms",
    "id":"601756D5-6537-4DC5-BD07-10D95BF1621E",
    "status":"Canceled"
}
```

7. Create Key Method (POST)

Description

This method creates an API Key for your subaccounts. Contact with Account Manager, in order to give you more information about how getting an API Key. Thanks to this method, you can provide your subaccounts with an API Key that they can use.

URL to web service operation

https://services.yuboto.com/omni/v1/CreateKey

Request

The request body is of type CreateKeyRequest.

```
public class CreateKeyRequest
{
      public string username { get; set; }
}
```

Parameters

The variables used to create a user's API Key are:

Variables	Description	Permitted Values	Required
username	The username of your subaccount.	String	Yes

CreateKeyRequest Example

```
{
    username = "demouser22"
}
```

Response
The response body is of type CreateKeyResponse.

```
public class CreateKeyResponse
{
      public int ErrorCode { get; set; }
      public string ErrorMessage { get; set; }
      public string username { get; set; }
      public string apiKey { get; set; }
      public string channel { get; set; }
}
```

- **ErrorCode:** The response error code for this call. This will be 0 if successful.
- **ErrorMessage:** The response error message. This will be null if successful.
- **username:** The user's username.
- **apiKey:** The unique API Key of the user.
- **channel:** The channel that your API Key can be used. Possible values are:
  viber - sending only VIBER message,
  sms - sending only SMS message,
  omni - a combination of all available channels. In case there are more than two channels, then the system will see the priority of each channel and send the messages to the first priority channel.

The CreateKeyResponse defines if an error occurred during the user's API Key creation. If the API Key created successfully then the ErrorCode has the value 0 and the ErrorMessage contains a zero length string. If an error occurred then, consult the error message that appears.

CreateKeyResponse Example

```
{
    "ErrorCode":0,
    "ErrorMessage":"",
    "username":"demouser22",
    "apiKey":"-----NewApiKey-----",
    "channel":"sms"
}
```

8. Two Way Authentication Validation Method (POST)

Description

This method validates the pin for specific sms id for two factor authentication messages.

URL to web service operation

https://services.yuboto.com/omni/v1/verifypin

Request

The request body is of type VerifyRequest.

```
public class VerifyRequest
    {
        public string id { get; set; }
        public string pin { get; set; }
    }
```

Parameters

The variables used to create a user's API Key are:

Variables	Description	Permitted Values	Required
id	Sms id system returned upon submition	String	Yes
pin	User enterd pin for validation	String	Yes

Response

The response body is of type VerifyRequest.

```
public class VerifyPinResponse
    {
        public int ErrorCode { get; set; }
        public string ErrorMessage { get; set; }
        public string id { get; set; }
        public string pin { get; set; }
        public string phonenumber { get; set; }
        public string status { get; set; }
    }
```

CreateKeyResponse Example

```
{
    "ErrorCode":0,
    "ErrorMessage":"",
    "id":"---sms id---",
    "pin":"--pin---",
    "phonenumber":"—receiver’s phonenumber ---",
    "status ":"valid",

}
```

If validation fails then the system will return error codes:

Error Code	Description
37	Invalid pin
38	Pin expired

9. Phone-number Reporting (POST)

Description

This method retrieves all existing send/delivery reports for previous campaigns based on recipient's phone number.

URL to web service operation

https://services.yuboto.com/omni/v1/DlrPhonenumber

Request

The request body is of type DlrPhonenumber.

```
Public class DlrPhonenumber
    {
      public string phonenumber { get; set; }
    }
```

Parameters

The variables used to retrieve existing reports for specific recipient:

Variables	Description	Permitted Values	Required
phonenumber	Recipient's Phone number in international format	number	Yes

Response

The response body is of type DlrPhonenumberResponse.

```
Public class DlrPhonenumberResponse
    {
      public int ErrorCode { get; set; }
       public string ErrorMessage { get; set; }
       public string phonenumber { get; set; }
       public List<DlrPhonenumberChannel> dlr { get; set; }
    }
```

DlrPhonenumberChannel object is of type

```
public class DlrPhonenumberChannel
    {
       public string id { get; set; }
       public string system { get; set; }
       public string channel { get; set; }
       public int priority { get; set; }
       public string status { get; set; }
       public decimal cost { get; set; }
       public string sender { get; set; }
       public string text { get; set; }
       public DateTime? submitDate { get; set; }
       public DateTime? dlrDate { get; set; }
    }
```

DlrPhonenumberResponse Example

```
{
    "ErrorCode": 0,
    "ErrorMessage": null,
    "phonenumber": "306972244887",
    "dlr": [
        {
            "id": "XXXX-XXXX-XXXX-XXXX-XXXXX",
            "system": "Api",
            "channel": "sms",
            "priority": 0,
            "status": "Delivered",
            "cost": 0.04,
            "sender": "Test",
            "text": "This is a test",
            "submitDate": "2019-01-11 18:38:03",
            "dlrDate": "2019-01-11 18:38:09"
        }
    ]
}
```