Campaign API

Revisions

Revision	Author	Date	Comment
1	pmarechal	05/03/2015	Version initiale de la documentation
2	cenee	09/03/2015	Relecture, modifs mineures, commentaires
3	pmarechal	26/03/2015	Ajout des sections concernant le tracking des liens, les liens de dés-inscription et de prévisualisation.
4	blegoff	31/12/2015	Ajout d'explications et d'exemple sur les campagnes transactionnelles.
5	pmarechal	11/01/2015	Ajout du lien vers la bibliothèque PHP facilitant l'intégration
6	pmarechal	22/03/2016	English version

This document describes the programming interface, or API, of Digitaleo's platform for sending multichannel marketing campaigns. This API allows you to create multichannel and multistep campaigns.

Table of contents

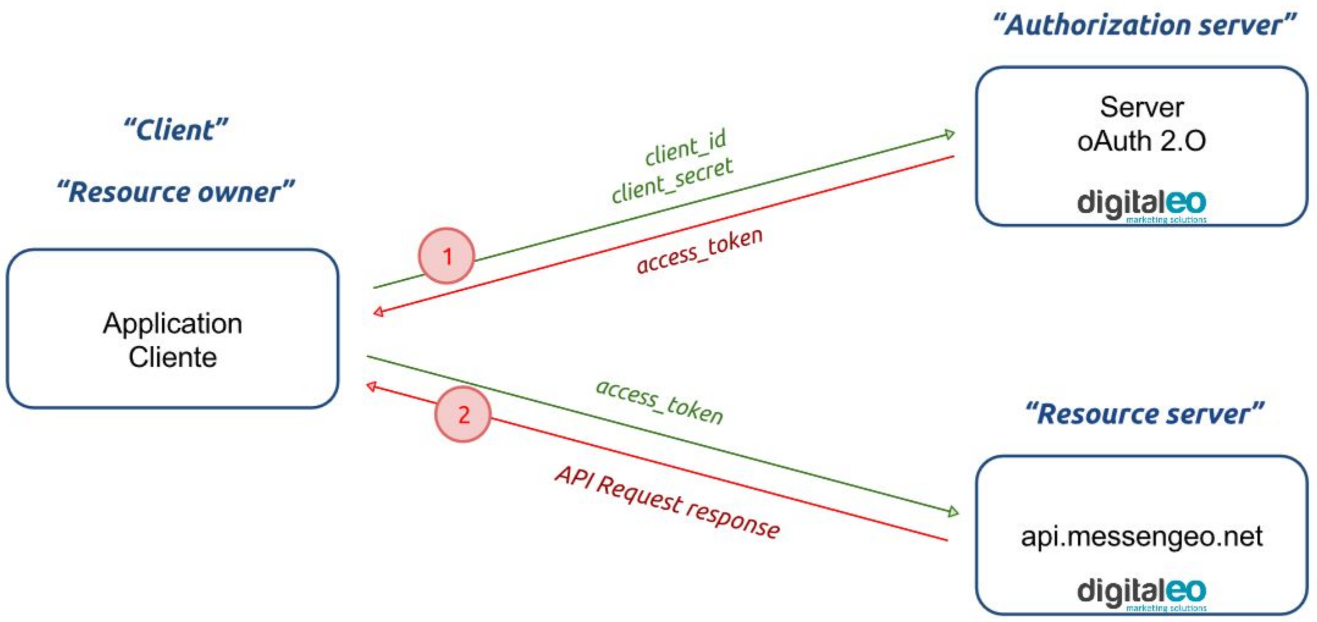
1. Overview

This API is not RESTfull because for all of the calls, the verbs HTTP GET and POST can be used. However, it is based on various resources of which the details are provided further on in this document.

The purpose of this first section is to help you understand the various types of calls to our APIs, regardless of the resource.

1.1. Authentification

Authentication to our APIs is based on the oAuth 2.0 protocol. Each call to our APIs has to contain an access_token that the client application will have requested beforehand from the Digitaleo authorization server:



1.1.1. Retrieving the application ids

To retrieve a client_id and a client_secret, you must declare an application in the Digitaleo platform.
For this,

1. Connect to app.digitaleo.com
2. Click on the Parameters menu
3. Go to the API tab

1.1.2. Retrieve an authentication token

The client must perform a POST request with the following parameters:

- grant_type: The value must be "client_credentials" for this type of authorization
- client_id: The id of the application (client)
- client_secret: The secret key of the application (client)

Note: The client_id and client_secret will be sent to you.

The URL for retrieving a token is as follows

```
https://oauth.messengeo.net/token
```

Example of an HTTP request

```
POST /token HTTP/1.1
Host: oauth.messengeo.net
Content-Type: application/x-www-form-urlencoded

client_id=51612c780b4dbaea8f81995beccbcfec08969d0e&
client_secret=p280edbd76d510c41990cbe5e6108c7e&
grant_type=client_credentials
```

Example of a request with Curl

```
curl https://oauth.messengeo.net/token
  -d 'client_id=51612c780b4dbaea8f81995beccbcfec08969d0e'
  -d 'client_secret=p280edbd76d510c41990cbe5e6108c7e'
  -d 'grant_type=client_credentials'
```

Return

if successful, the authorization server will return a code 200 HTTP response of which the body will contain the following JSON flow

```
{
   "access_token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2...",
   "expires_in":"3600",
   "token_type":"bearer",
   "scope":"basic",
}
```

Description of the various fields:

Property	Description
access_token	The token issued by the authorization server. *Note: The size of the token can range up to 50,000 characters*
expires_in	The lifespan in seconds of the token issued
token_type	The type of token. The Digitaleo server only supports the "bearer" type
scope	The scope of the token

If one of the parameters is not correct, the authorization server will return a code 400 http response (HTTP/1.1 400 Bad Request) of which the body will contain the following json flow:

```
{
   "error":"invalid_client",
   "error_description":"The client credentials are invalid",
}
```

1.1.3. Using the authorization token (access_token)

The authorization token is sent to the API in the header of the HTTP request and more particularly in the header “Authorization: Bearer”. Note that the “Authorization: Bearer” is case-sensitive.

Example of an HTTP request

```
GET /rest/campaigns HTTP/1.1

Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
```

Example of a request with Curl

```
curl -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2..."
https://api.messengeo.net/rest/campaigns
```

1.2. Pagination, sorting and filtering

1.2.1. Pagination

1.2.1.1. introduction

Three parameters allow you to manage the paginated resource display.

The parameters are

- **limit**: allows you to limit the number of elements returned
- **offset**: allows you to ignore the first n elements of the list
- **total**: allows you to retrieve the total number of resources if indeed the request had not limited the number of resources returned. It is therefore useful in the framework of using a limit for pagination. (the default is false). This feature uses a lot of resources. It is recommended that it not be activated in the case there is no pagination.

1.2.1.2. Example 1: Limiting results

Retrieving only 20 resources and the total number of resources if the result were limited to 20 resources

```
GET /rest/ressource HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded

limit=20&total=true
```

with Curl

```
curl
 -X GET
 -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
 -d limit=20
 -d total=true
https://api.messengeo.net/rest/ressource
```

1.2.1.3. Example 2: Pagination

Retrieving 10 resources, leaving aside the first 20. This boils down to reading the 3rd page knowing that each page lists 10 resources.

```
GET /rest/resource HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded

limit=10&offset=20
```

1.2.2. Sorting

1.2.2.1. Introduction

The **sort** parameter allows you to order the list of resources returned according to one of the attributes of the resource concerned. This parameter is comprised of two elements separated by a space:

- The name of the attribute according to which you want to order the list
- The sorting order, ascending order (ASC) or descending order (DESC)

1.2.2.2. Exemple

Sorting a list of resources in descending order according to the id of this resource:

```
GET /rest/resource HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded

sort=id%20DESC
```

with Curl

```
curl
 -X GET
 -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
 -d sort=id DESC
https://api.messengeo.net/rest/ressource
```

1.2.3. Limiting the list of attributes returned per resource

1.2.3.1. Introduction

It is possible to limit the list of attributes of the resources returned. In other words, this entails returning incomplete resources in order to focus on the attributes that are really necessary for the client that generated the call.

This makes it possible to save both bandwidth and processing on the server side. Indeed, certain attributes are calculated at the time of the call and not retrieving them makes it possible to reduce the request time.

The parameter that allows you to define the attributed return is called properties.

For each resource, a list of attributes returned by default (if the properties parameter is not defined) is imposed. An alias called DEFAULT makes it possible to specify that you want to retrieve the attributes by default + another or – another.

1.2.3.2. Example 1

Retrieving only the is and the name of each resource

```
GET /rest/resource HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded

properties=id,name
```

with Curl

```
curl
 -X GET
 -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
 -d properties=id,name
https://api.messengeo.net/rest/ressource
```

1.2.3.3. Example 2

Retrieving the attributes returned by default except the id

```
GET /rest/resource HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded

properties=DEFAULT,-id
```

with Curl

```
curl
 -X GET
 -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
 -d properties=DEFAULT,-id
https://api.messengeo.net/rest/ressource
```

1.3. The lists of resources returned

Most of the calls to the REST API return a list of resources. This list of resources is comprised of the following three elements:

- **size**: The number of resources returned
- **total**: The number of resources returned if the request had not limited the result
- **list**: The table containing the resources

An example of a list of resources returned in json format

```
{
    "size": 2,        // The number of resources returned
    "total": 600640, // The number of resources returned if the request had not limited the result
    "list":           //  The table containing the resources
    [
    {
        "id": "1",
        "email": "aladdin@digitaleo.com",
        "phone": "+33201010101",
        "mobile": "+33601010101",
    },
    {
        "id": "2",
        "email": "jasmine@digitaleo.com",
        "phone": "+33202020202",
        "mobile": "+33602020202",
    }
    ]
}
```

1.4. . The various actions on a resource

1.4.1. Introduction

Our APIs comply with the HTTP verbs and their correspondence with the CRUD actions (Create, Read, Update, Delete) of a resource. However, it is also possible to perform all of the actions only with HTTP GET requests or only with HTTP POST requests. To do this, the action to be performed must be specified in the URL.

Description of the action	Dedicated HTTP verb	Name of the action
Read resources	GET	read
Create a resource	POST	create
Update resources	PUT	update
Delete resources	DELETE	delete

1.4.2. Example

The two following Curl requests are considered to be equivalent by our APIs

```
curl
 -X GET
 -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
https://api.messengeo.net/rest/ressource
```

```
curl
 -X POST
 -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
https://api.messengeo.net/rest/ressource?action=read
```

1.4.3. Profil des méthodes en fonction des actions

Description of the action	input	output
read	Filter	List of resources
create	Parameters	Resource created
update	Filter + Parameter	Number of resources updated
delete	Filter	Number of resources deleted

1.5. Updating resources

Updating resources is based on two types of parameters:

- A list of filters that allows you to limit the resources to be updated
- A list of keys/values that allow you to define which resource attributes to update, with which value.

The keys/values to be updated must be grouped together in a "metaData" attribute, in the form of a JSON flow.

For example, the following request makes it possible to update the name and the description of resources for which the is is either 100, or 200 and for which the name is equal to "former name of the resource"

```
PUT /rest/ressource HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded

id=100,200&name='ancien nom de la ressource'&metaData='{"name":"Le nouveau nom de la
ressource","description":"La nouvelle description de la ressource"}'
```

with Curl

```
curl
 -X PUT
 -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
 -d id=100,200
 -d name='ancien nom de la ressource'
 -d metaData='{"name":"Le nouveau nom de la ressource","description":"La nouvelle
description de la ressource"}'
 https://api.messengeo.net/rest/ressource
```

Actions that update resources return the number of contacts involved by the filter passed as input

```
{
   count: 10,
}
```

1.6. Return codes

The HTTP response code is contained:

- in the HTTP header,
- in the content of the response in the case of an error.

The return codes are based on the HTTP return codes:

- 2XX - The call to the API unfolded correctly
- 4XX – The call to the API has an error in its parameters.

Codes with success:

- **200 OK**: everything went well
- **201 Created**: Resource created
- **204 No Content**: Resource updated or deleted

The error codes that you are likely to see are the following:

- **304 Not Modified**: Error during updating or deleting (the resource is not modified)
- **400 Bad Request**: Missing or incorrect parameter
- **401 Unauthorized**: Authentication failed
- **403 Forbidden**: Access to the requested location is prohibited
- **404 Not Found**: Unknown method or method not indicated
- **405 Method Not Allowed**: You are not authorized to use the method that you are requesting
- **414 Request-URI Too Long**: Your request is too large, please shorten it
- **417 Expectation Failed**: The required parameters are either missing or are incorrect
- **500 Internal Server Error**: Unidentified error

For example, if the authentication token is no longer valid for the following request:

```
curl
 -X GET
 -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
https://api.messengeo.net/rest/ressource
```

The header of the HTTP response will be

```
< HTTP/1.1 401 Unauthorized
< Date: Fri, 06 Mar 2015 21:32:06 GMT
< Server: Apache/2.2.16 (Debian)
< X-Powered-By: PHP/5.3.3-7+squeeze15
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
< Content-Length: 46
< Content-Type: application/json
```

while the body of the HTTP response will be

```
{
    "status": 401,
    "message": "Authenticate failed"
}
```

1.7. Response formats

1.7.1. Introduction

The REST API can respond to the requests in different formats. By default, it returns a response in JSON format but it can also return a response in XML, CSV (for certain resources) and JS (JSONP) formats.

To change the format, .xml, .json, .csv or .js must be added to the end of the URI regardless of the HTTP verb (GET, POST, DELETE or PUT)

1.7.2. Examples

To retrieve the list of mailings in JSON format

```
GET /rest/ressource.json HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded
```

To retrieve the list of mailings in XML format

```
GET /rest/ressource.xml HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded
```

To retrieve the list of mailings in CSV format

```
GET /rest/ressource.csv HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded
```

To retrieve the list of mailings in JSONP format

```
GET /rest/ressource.js HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded

callback=yourFunctionCallback
```

1.7.3. Gestion du cross-domain

Retrieving the response in javascript format allows you to overcome the difficulties linked to the cross-domain. Passing through a server in order to consult the API's directly is thus avoided. On the client side, it is recommended to use the jquery-jsonp plugin (jQuery-jsonp on GitHub) for error management (not initially available in JQuery).

For example, reading the resource of which the id is 2 via an ajax request returns

```
<script type="text/javascript" language="javascript" src="jquery.jsonp.js"></script>
<script>
  $.jsonp({
      url: 'https://api.messengeo.net/rest/ressource.js?callback=?',
      beforeSend: function (request) {
          request.setRequestHeader("Authorization", "Bearer " + ($("#accesstoken").val()));
      },
      data: {
          id: '2',
      }
  }).done(function(data) {
     // data peut être un objectlist (size, total, list) ou une erreur (status, message)
     console.log(data);
  }).fail(function(jqxhr, textStatus, errorThrown) {
     console.log('Errors occured');
  });
</script>
```

1.8. Filters and passing multiple values

The read, update and delete actions use as input a filter which makes it possible to select only the resources to be read or to be affected. Most of these filters take several values.

There are two ways to pass these multiple values:

1. in the form of a character string with the values separated by commas;
2. in the form of a table.

For example, the following two requests allow you to retrieve the resources for which the is is equal either to 12, or equal to 13.

```
GET /rest/ressource.json HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded

id=12,13
```

```
GET /rest/ressource.json HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded

id[0]=12&id[1]=13
```

1.9. Integrating our API as PHP

In order to simplify the integration of our REST APIs, we provide you with a library that facilitates the various calls from code written in php. This library is hosted on GitHub.

2. Campaigns

2.1. A list of associated contacts

Each campaign is associated with a list of contacts This list of contacts is managed by Digitaleo's contact management API. Please refer to the documentation for this API for more information.

2.2. Multistep campaigns

A Digitaleo campaign can be multistep, with each step itself able to be multichannel. This allows you to generate a single campaign for all of the communication concerning an event (save the date, D-7 reminder, thank yous...).

2.3. Multichannel campaigns

Our campaign API now makes it possible to manage the SMS, EMAIL, VOICE and VOICEMAIL channels.

The VOICE channel corresponds to telephone calls (fixed and mobiles) broadcasting a voice message. The VOICEMAIL channel entails leaving a voice message on the answering machine (without ringing the telephone) and is for mobile telephones only.

2.4. Campaign types

Digitaleo currently supports 4 different campaign types

STANDARD: Conventional marketing sending campaigns

TRIGGERED: Campaigns for which the sending is triggered according to a date present at the contact level. For example, this type of campaign allows you to wish your contacts happy birthday

RECURRING: Campaigns sent at regular intervals across the entire list of contacts

TRANSACTIONAL: Campaigns for which the sending is done when contacts are added to it. A date can be defined in order to defer the sending of the messages to these new contacts. This type of campaign is not associated with a list of contacts as can be understood for the other types.

3. Reference documentation

3.1. Campaigns

3.1.1. List of properties of the Campaign resource

Property	Description
id	Campaign id
name	Name/Label for the campaign
reference	Free text (can be for example an external id given by the creator of the campaign)
listId	Id of the list of contacts associated with the campaign
listName	Name of the list of contacts associated with the campaign
listCount	Number of contacts in the list of contacts associated with the campaign
nbSteps	Number of steps in the campaign
type	Campaign type **STANDARD**: Conventional marketing sending campaign; **TRIGGERED**: Campaign based on the date in the contacts for the sending; **RECURRING**: Campaign sent at regular intervals across the entire list of contacts; **TRANSACTIONAL**: Campaign without a list of contacts, to which can be added new contacts after it is created.
dateStart	Start date for the campaign
dateEnd	End date for the campaign
cancelled	Cancellation indicator for the campaign (1 if the campaign has been canceled, 0 otherwise)
status	Campaign status ● created: campaign ready to be built (default status) ● canceled: campaign canceled before its start date (so no message has been sent) ● interrupted: campaign canceled after its start date (so a portion of the messages may have been sent) ● ended: campaign ended (each mailing has exceeded the timeout for its media)
comment	Comment concerning the campaign
dateCreated	Creation date for the campaign

3.1.2. Listing campaigns

3.1.2.1. List of available filters

Property	Description
id	Filter according to one or more campaign ids
reference	Filter according to the customer reference of the campaign
type	Filter allowing for selections according to the type of campaign
cancelled	Filter allowing for selection only of canceled or non-canceled campaigns
status	Filter according to the status of the campaign
dateStart	Allow you to filter campaigns that are more recent than dateStart
dateEnd	Allow you to filter campaigns that are older than dateEnd

Example

List the campaigns for which the id is either 24 or 25

```
GET /rest/campaigns HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded

id=24,25
```

or in POST (but by specifying the action)

```
POST /rest/campaigns HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded

action=read&id=24,25
```

And with Curl

```
curl  -H  "Authorization:  Bearer  eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…"
https://api.messengeo.net/rest/campaigns?id=24,25
```

3.1.2.2. Return

This method sends back as return campaigns encapsulated in a structure that also contains:

```
{
       "size": 2,         // Nombre campagnes renvoyées
       "total": 2,        // Nombre total de campagnes si la requête n’était pas limitée
       "list": [          // Liste des ressources campaigns
       {
            "id": "24",
            "name": "Campagne 15/10/2013 11:50",
             "cancelled": "0",
             ...
             "status": "ended",
             "type": "STANDARD"
             "dateCreated": "2014-06-09 10:51:59",
       },
       {
            "id": "25",
            "name": "Campagne 09/06/2014 10:50",
             "cancelled": "0",
             ...
             "status": "ended",
             "type": "STANDARD"
             "dateCreated": "2014-06-09 10:51:59",
       }
       ]
}
```

3.1.3. Creating a campaign

3.1.3.1. List of parameters to supply in order to create a campaign

Parameter	Type	Description	Required?
name	string	Name of the campaign	yes
listId	integer	Id of the list of contacts	if at least 1 PUSH mailing (EMAIL, SMS, VOICE, VOICEMAIL)
steps	array	Table of the steps that the campaign will contain. For details on the list of parameters to supply for each step, cf. #3.1.3.2.	yes
reference	string	Free text (can be for example an external id given by the creator of the campaign)	no
type	string	Campaign type: **STANDARD**: Conventional marketing sending campaign **TRIGGERED**: Campaign based on the date in the contacts for the sending **RECURRING**: Campaign sent at regular intervals across the entire list of contacts **TRANSACTIONAL**: Campaigns to which new contacts can be added (single-step campaigns)	no (default: STANDARD)
comment	text	Comment concerning the campaign	no

3.1.3.2. List of parameters for the steps table

Parameter	Type	Description	Required?
mode	string	Sending mode ● combined: combined sending across all the channels ● prioritized: prioritized sending	if at least 2 PUSH mailings (EMAIL, SMS, VOICE, VOICEMAIL)
date	string	Send date for the step *Note: Only for STANDARD campaigns (if omitted or in the past, immediate sending)*	no
mailings	array	Table of the mailings that the step will contain. For details on the list of parameters to supply for each mailing, cf. 3.1.3.3.	yes
priorities	Array or string	List of PUSH channels (EMAIL, SMS, VOICE, VOICEMAIL) to prioritize in the framework of substitution *Note: 2 equivalent examples:* *array('SMS','EMAIL')* *"SMS,EMAIL"*	if "prioritized" mode
autoPeriod	string	String of characters representing the periodicity at which the campaign is to be sent in the framework of automated or recurring campaigns *Note: See syntax hereinbelow*	if campaign of the TRIGGERED or RECURRING type
autoField	string	Contact field used as a basis in the framework of automated campaigns	if campaign of the TRIGGERED type
reference	string	Free text (can be for example an external id given by the creator of the campaign)	no

3.1.3.3. List of parameters for the mailings table

Parameter	Type	Description	Required?
text	string	Content in TEXT format ● SMS: content of the SMS ● EMAIL: text version	if EMAIL, SMS media
html	string	Content in HTML format *Notes:* ● *Only for the EMAIL media* ● *At least one of the two "text" and "html" fields must be present*	if EMAIL media
link	string	Link representing the content: ● VOICE: link to a VXML, WAV or MP3 file ● VOICEMAIL: link to a WAV or MP3 file	if VOICE, VOICEMAIL media
subject	string	Subject of the e-mailing *Note: Only for the EMAIL media*	if EMAIL media
replyContact	string	Reply means of the e-mailing *Note: Only for the EMAIL media*	no
sender	string	Name of the sender ● SMS: value of the TPOA ● EMAIL: name of the sender	no
media	string	Media used to send messages SMS, EMAIL, VOICE, VOICEMAIL	yes
type	string	Type associated with the media Possible types: ● VOICE: "vxml","wav" (choose "wav" for WAV and MP3 files) ● SMS: "response"	if VOICE media
pingUrl	string	Notification url for a status change on one or more messages.	no
batchVolume [2]	int	If the mailing has to be sent in packets, batchVolume defines the number of messages of each packet. *Note: The programming rate for messages per minute cannot exceed the maximum sending rate per minute of the various providers.*	no
batchDelay	int	If the mailing has to be sent in packets, batchDelay defines the number of minutes between the programming dates of the various message packets *Note: If one of the fields "batchVolume" or "batchDelay" is present, the second one must also be present*	no

3.1.3.4. Link for unsubscribing

So that the recipient of emails can unsubscribe from the service, a link for unsubscribing must be present in the body of the email (HTML and TEXT). This link must be inserted via the #OPTOUTLINK# field which will be replaced with the link for unsubscribing that is proper to each recipient.

The #OPTOUTLINK# field in the HTML and TEXT portions is required in order to save the mailing.

For example

```
<html>
...
Si vous souhaitez vous désabonner,
rendez-vous sur ce <a href='#OPTOUTLINK#'>lien</a>.
...
</html>
```

3.1.3.5. Preview link

Just like the link for unsubscribing, there is also a customized field allowing a preview link to be inserted into the HTML creation. This link must be inserted via the #PREVIEWLINK# field which will be replaced with the preview link.

For example

```
<html>
...
Si le message ne s'affiche pas,
merci de suivre ce <a href='#OPTOUTLINK#'>lien</a>.
...
</html>
```

3.1.3.6. Links present in the HTML and TEXT portions

When mailings are created, the links present in the HTML and TEXT portions are detected in order to be tracked. For each link, an id corresponding to both the link and to the contact is added which allows us to know when link was clicked, by whom, when, etc.
To deactivate link tracking, simply add the attribute rel= "notrack" in each of the links.

For example, to not track a link present in an image

```
<html>
...
<a rel="notrack" href="http://monurl.com">
  <img src="http://monimage.com" />
</a>
...
</html>
```

3.1.3.7. Specifying a URL or PING

Instead of regularly polling the status of a message until it is received ("pull" mode), you can be informed ("push" mode) of the arrival of a status linked to the messages that you have sent.

For this, you supply, when the mailing is created, an address on your server 'http://myserver.com/ping.php ?id=#id# ' for example, and the Digitaleo platform will call this address with the id of the email when its status changes.

This URL is passed in the pingUrl parameter.

Cumulating changes in status ("batch") is possible by using the #ids# pattern instead of #id#. The ids are then separated by commas.

3.1.3.8. Syntax of the period for automated or recurring campaigns

The expected syntax is inspired by the crontab format

```
<minute> <hour> <day> <month> <dayWeek>
```

Note: dayWeek is a series of indexes (0=Sunday, 1=Monday, etc)

For recurring campaigns, the possible values are:

- *minute & hour*: value
- *day*: value, interval, every day ("*")
- *month*: value, interval, every month ("*")
- *dayWeek*: value, interval, every day of the week ("*")

For triggered campaigns, a reference to the day and to the month of the autoField can be used with the values "D" and "M". The possible values are:

- *minute & hour*: value
- *day*: reference to the day of the "autoField" field ("D") - a number of days
- *month*: all months ("*") or reference to the month of the "autoField" field ("M") - a number of months
- *dayWeek*: value, interval, every day of the week ("*")

Example for recurring campaigns

00 12 1 * *	Every 1st day of the month at 12:00, regardless of the day of the week

Example for triggered campaigns

30 11 D-1 M 1,2,3,4,5,6	Once a year at 11:30 the day before the date in the contact field, with a shift to Saturday if the date falls on a Sunday

3.1.3.9. Example 1: Creating a campaign comprising 2 steps, of one is prioritized and one combined

```
$myNewCampaign = $messengeo->campaignsCreate(
 array(
   'name'           => 'Soldes du 10 juillet',
   'reference'      => '12',
   'listId'         => '2674',
   'steps'          => array(
       array(
           'mode'          => 'prioritized',
           'priorities' => 'EMAIL,SMS',
           'date'          => '2013-07-10 12:00:00',
           'mailings'   => array(
               array(
                   'name'         => 'mailing email du 10 juillet',
                   'text'         => 'Bientôt les soldes',
                   'html'         => '<html><head>Titre de la page</head><body>Bientôt les
soldes</body></html>',
                   'subject'      => "Info soldes",
                   'replyContact' => 'sender@gmail.com',
                   'media'        => 'EMAIL',
               ),
               array(
                   'name'         => 'mailing sms du 10 juillet',
                   'text'         => 'Bientôt les soldes',
                   'media'        => 'SMS',
               ),
           ),
           'reference' => 'Ref1',
       ),
       array(
           'mode'          => 'combined',
           'date'          => '2013-07-15 12:00:00',
           'mailings'   => array(
               array(
                   'name'         => 'mailing email du 15 juillet',
                   'text'         => 'Profitez des soldes',
                   'html'         => '<html><head>Titre de la page</head><body>Profitez des
soldes</body></html>',
                   'subject'      => "Rappel des soldes",
                   'replyContact' => 'sender@gmail.com',
                   'media'        => 'EMAIL',
               ),
               array(
                   'name'         => 'mailing sms du 15 juillet',
                   'text'         => 'Profitez des soldes',
                   'media'        => 'SMS',
               ),
           ),
       ),
   )
 )
);
```

3.1.3.10. Example 2: How to create an automated campaign

Creating an **automated campaign** using the birthDate field knowing that the message must be sent once a year at 11:45 the day before the birthday, with a shift to Saturday if the date falls on a Sunday.

```
$autoCampaign = $messengeo->campaignsCreate(
  array(
    'type'  => 'TRIGGERED',
    'steps' => array(
       array(
         ...
         'autoField'  => 'birthDate',
         'autoPeriod' => '45 11 D-1 M 1,2,3,4,5,6',
         ...
      )
    )
  )
);
```

3.1.3.11. Example 3: How to create an automated campaign - II

Creating an **automated campaign** using the field03 field knowing that the message has to be sent every month at 12:30 two days before the date mentioned in the contact.

```
$autoCampaign = $messengeo->campaignsCreate(
 array(
   'type'  => 'TRIGGERED',
   'steps' => array(
      array(
        ...
        'autoField'  => 'field03',
        'autoPeriod' => '30 12 D-2 * *',
        ...
      )
    )
  )
);
```

3.1.3.12. Example 4: How to create an automated campaign - III

Creating an **automated campaign** using the field06 field knowing that the message has to be sent every year at 12:30 1 month before the date mentioned in the contact.

```
$autoCampaign = $messengeo->campaignsCreate(
 array(
   'type'  => 'TRIGGERED',
   'steps' => array(
     array(
        ...
        'autoField' => 'field06',
        'autoPeriod'   => '30 12 D M-1 *',
        ...
     )
   )
 )
);
```

3.1.3.13. Example 5: How to create a recurring campaign

Creating a recurring campaign knowing that the message has to be sent (to all of the contacts) on the 10th of every month at 12:30. If the 10th falls on a Sunday, the message will be sent on the 9th.

```
$autoCampaign = $messengeo->campaignsCreate(
 array(
   'type'  => 'RECURRING',
   'steps' => array(
     array(
        ...
        'autoPeriod'   => '30 12 10 * 1,2,3,4,5,6',
        ...
     )
   )
 )
);
```

3.1.3.14. Example 6: How to create a transactional campaign

Creating a **transactional campaign** of which the message will be sent to each new recipient that is associated with it.

```
$myNewCampaign = $messengeo->campaignsCreate(
 array(
   'name'           => 'Campagne nouveau client',
   'type'          => 'TRANSACTIONAL',
   'steps'          => array(
       array(
           'priorities' => 'EMAIL',
           'mailings'   => array(
               array(
                   'name'         => 'mailing pour les nouveaux clients',
                   'html'         => '<html><head>Titre de la page</head><body>Bienvenue en
tant que nouveau client</body></html>',
                   'subject'      => "Bienvenue",
                   'replyContact' => 'sender@gmail.com',
                   'media'        => 'EMAIL',
               ),
           )
   ))
));
```

Creation of two new contacts associated with the previously created transactional campaign, having 15 for id. In the following example, the messages will be sent to the contacts on the date of 16/02/2015 at 5 o'clock. If no date is defined, the messages will then be sent automatically.

```
$myNewContacts = $messengeo->campaigncontactsCreate(
 array(
   'campaignId'     => 15,
   'date'          => '2015-02-16 05:00:00',
   'contacts'       => array(
       array(
           'email'     => 'contact1@digitaleo.com',
           'firstName' => 'Contact1 FirstName',
           'LastName'  => 'Contact1 LastName'
       ),
       array(
           'email'     => 'contact2@digitaleo.com',
           'firstName' => 'Contact2 FirstName',
           'LastName'  => 'Contact2 LastName'
       )
   )
 )
);
```

3.1.3.15. Example 7: Spreading out the sending

Creating an SMS campaign in packets of **100 messages** every **20 minutes**.

```
$myNewCampaign = $messengeo->campaignsCreate(
 array(
   'name'           => 'appel conseiller',
   'reference'      => '12',
   'listId'         => '2674',
   'steps'          => array(
       array(
           'date'          => '2013-07-10 12:00:00',
           'mailings'   => array(
               array(
                   'text'         =>  'Bonjour,  appelez  votre  conseiller  au  06...  pour
discuter de votre contrat',
                   'media'        => 'SMS',
                   'batchVolume'        => 100, // nombre de messages
                   'batchDelay'         => 20, // nombre de minutes entre chaque paquet
               ),
           ),
       ),
   )
 )
);
```

3.1.4. Canceling a campaign

3.1.4.1. List of parameters to supply

Parameter	Type	Description	Required?
action	string	=cancel allows you to specify the action to be carried out	yes (action=cancel)
id	int	Id of the campaign to be canceled	yes

3.1.4.2. Notes

- The end date of the campaign takes the date of cancellation
- The status of the campaign is modified ("canceled" if the cancellation occurs before the start date of the campaign, "interrupted" otherwise)

3.1.4.3. Return

A resource of which the only property is success is returned. 1 in case of a successful cancellation 0 otherwise.

```
{
  success: 1,
}
```

3.1.4.4. Example

```
POST /rest/campaigns HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded

action=cancel&id=24
```

3.1.5. Updating a campaign

3.1.5.1. List of parameters to supply

Parameter	Type	Description	Required?
action	string	=update allows you to specify the action to be carried out	no if the verb http PUT is used, Otherwise yes (action=update)
id	int	List of campaign Ids to be updated	yes
metaData	array	List of fields of the campaign Ids be updated. cf. 3.1.5.2	yes

3.1.5.2. List of fields of a campaign Ids that can be updated

Parameter	Type	Description
name	string	Name of the campaign
reference	string	Client reference of the campaign
comment	array	Comment on the campaign

3.1.5.3. Return

This method returns the number of campaigns affected by the filter passed as input

```
{
  count: 1,
}
```

3.1.5.4. Example

```
POST /rest/campaigns HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded

action=update&id=24&metaData={"name":"Le nouveau nom de la campagne","comment":"Le nouveau
commentaire de la campagne"}
```

With Curl

```
curl
-X PUT
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d id=24
-d metaData='{"name":"Le nouveau nom de la campagne","comment":"Le nouveau commentaire de la
campagne"}'
https://api.messengeo.net/rest/campaigns
```

3.1.6 Retrieving statistics for a campaign

3.1.6.1 List of available filters

Property	Description
mailingId	Filter according to one or more mailing ids
dateStart	Filter according to a start date of the mailing
dateEnd	Filter according to an end date of the mailing
messageId	Filter according to one or more message ids
media	Filter according to a media
mailingName	Filter according to a mailing name
period	Filter according to a period (statistics group: year, month, day)

3.1.6.2. Return

```
{
	total: 45554,
	wait: 0,
	on: 45554,
	ok: 0,
	ko: 0,
	no: 0,
	optout: 0,
	opened: 0,
	clicked: 0,
	hb: 0,
	sb: 0,
	rep: 0,
	date: null
}
```

3.2.6.3. Examples

Example 1: Retrieving the statistics of two mailings

```
GET /rest/statistics HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded

mailingId=8374,8375
```

With Curl

```
curl
-X GET
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d mailingId=8374,8375

https://api.messengeo.net/rest/statistics
```

Example 2: Retrieving the statistics of a mailing per day

```
GET /rest/statistics HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded

mailingId=8374
period=day
```

With Curl

```
curl
-X GET
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d mailingId=8374
-d period=day

https://api.messengeo.net/rest/statistics
```

3.2. Steps

3.2.1. List of properties of the Step resource

Property	Description
id	Id of the step
mode	Sending mode "combined" or "prioritized"
priorities	List of channels to prioritize in the framework of the substitution (EMAIL, SMS, VOICE, VOICEMAIL)
medias	List of channels used for this step
campaignId	Id of the campaign to which the step is attached
campaignName	Name of the campaign to which the step is attached
campaignListCount	Number of contacts in the list of the campaign
autoPeriod	String of characters representing the periodicity at which the campaign is to be sent in the framework of automated or recurring campaigns
autoField	Contact field used as a basis in the framework of automated campaigns
reference	Client reference for the step
date	Send date for the step
status	Status of the step ● created: step created (status by default) ● canceled: step canceled before its start date (so no message has been sent) ● interrupted: step canceled after its start date (so a portion of the messages may have been sent) ● ended: step ended (each mailing has exceeded the timeout for its media)
dateUpdated	Date of the last modification for the step
dateCreated	Creation date of the step (= confirmation date of the campaign)

3.2.2. Listing the steps

3.2.2.1. List of available filters

Property	Description
id	Filter according to one or more step ids
campaignId	Filter according to one or more campaign ids
reference	Filter on a customer reference
medias	Filter on one or more media of the step

3.2.2.2. Example 1: Retrieving a step from its id

```
GET /rest/steps HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded

id=103
```

With Curl

```
curl
-X GET
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d id=103
https://api.messengeo.net/rest/steps
```

3.2.2.3. Example 2: Retrieving the steps of a campaign containing an SMS sending

```
GET /rest/steps HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded

campaignId=24&medias=SMS
```

With Curl

```
curl
-X GET
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d campaignId=24
-d medias=SMS
https://api.messengeo.net/rest/steps
```

3.2.2.4. Return

This method sends back as return steps encapsulated in a structure that also contains:

```
{
   "size": 1,
   "total": 1,
   "list": [
   {
       "autoField": null,
       "autoPeriod": null,
       "campaignId": "615",
       "campaignListCount": "36",
       "campaignName": "Campagne 09/06/2014 10:50",
       "date": "2014-06-09 10:51:59",
       "dateCreated": "2014-06-09 10:51:59",
       "dateUpdated": "2014-06-12 10:52:11",
       "id": "618",
       "medias": [
           "SMS"
       ],
       "mode": "combined",
       "priorities": [
           ""
       ],
       "reference": "1475",
       "status": "ended"
   }
   ]
}
```

3.3. Mailings

3.3.1. List of properties of the Mailing resource

Property	Type	Description
id	string	Id of the mailing
name	string	Name of the mailing
text	string	Content in TEXT format ● SMS: content of the SMS ● EMAIL: text version
html	string	Content in HTML format *Note: Only for the EMAIL media*
link	string	Link representing the content: ● VOICE: link to a VXML, WAV or MP3 file ● VOICEMAIL: link to a WAV or MP3 file
subject	string	Subject of the e-mailing *Note: Only for the EMAIL media*
replyContact	string	Reply means of the e-mailing *Note: Only for the EMAIL media*
media	array	Media used to send messages *Note: Only for the EMAIL media*
sender	string	Name of the sender ● SMS: value of the TPOA ● EMAIL: name of the sender
nbMessages	integer	Number of messages contained in the mailings
date	string	Send date for the messages
stats	objects	Statistics on the status of messages
dateUpdated	string	Date mailing updated
dateCreated	string	Date mailing created
pingUrl	string	Notification url for a status change

stepId	integer	Id of the step if the mailing is in a context of a campaign, NULL otherwise
guid	string	Single global id of the mailing
status	string	Status of the mailing ● created: mailing (status by default) ● canceled: mailing canceled before its start date (so no message has been sent) ● interrupted: mailing canceled after its start date (so a portion of the messages may have been sent) ● ended: mailing ended (timeout of the media exceeded)
nbContacts	integer	Number of contacts

3.3.2. Listing mailings

3.3.2.1. List of available filters

Property	Description
id	Filter according to one or more mailing ids
name	Filter according to the name of the mailing
media	Filter according to the media of the mailing
stepId	Filter allowing you to retrieve the mailings of a step

3.3.2.2. Example 1: Retrieving a mailing from its id

```
GET /rest/mailings HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded

id=4682
```

With Curl

```
curl
-X GET
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d id=4682
https://api.messengeo.net/rest/mailings
```

3.3.2.3. Example 2: Retrieving the EMAIL mailing of a step

```
GET /rest/mailings HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded

stepId=103&media=EMAIL
```

With Curl

```
curl
-X GET
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d stepId=103
-d media=EMAIL
https://api.messengeo.net/rest/mailings
```

3.3.2.4. Return

This method sends back as return campaigns encapsulated in a structure that also contains:

```
{
       "size": 1,
       "total": 1,
       "list": [
       {
              "application": "API",
              "date": "2014-06-09 10:52:00",
              "dateCreated": "2014-06-09 10:52:00",
              "dateUpdated": "2014-06-12 10:52:11",
              "guid": "6d3a0010174cce8208eacfb953445b97",
              "html": null,
              "id": "9210",
              "link": null,
              "media": "sms",
              "metadata": null,
              "name": "6d3a0010174cce8208eacfb953445b97",
              "nbContacts": "36",
              "nbMessages": "36",
              "pingUrl": null,
              "replyContact": null,
              "sender": "",
              "stats": {
              "clicked": 0,
              "date": null,
              "hb": 0,
              "ko": 0,
              "no": 0,
              "ok": 0,
              "on": 0,
              "opened": 0,
              "optout": 0,
                "rep": 0,
                "sb": 0,
                "total": 0,
                "wait": 0
                },
                "status": "ended",
                "stepId": "618",
                "subject": null,
                "text": "Campaign No2 marketing"
        }
        ]
}
```

3.3.3 Retrieving statistics for a mailing

To obtain the statistics of a mailing, the stat property must b explicitly requested which triggers the calculation of the statistics for the mailing. By default, statistics are not returned for questions of performance.

```
GET /rest/mailings HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded

id=4682,properties=stats
```

With Curl

```
curl
-X GET
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d id=4682
-d properties=stats
https://api.messengeo.net/rest/mailings
```

3.3.3.1 Email/Voice/Voicemail Statistics

```
{
        size: 1,
        total: 1,
        list:
        [
                {
                        id: "8374",
                        name: "fa20c7a795658987a33b74cef8e9352f",
                        stats:
                        {
                                total: 45554,
                                wait: 0,
                                on: 45554,
                                ok: 0,
                                ko: 0,
                                no: 0,
                                optout: 0,
                                opened: 0,
                                clicked: 0,
                                hb: 0,
                                sb: 0,
                                rep: 0,
                                date: null
                        }
                }
        ],
        httpStatusCode: 200
}
```

3.3.3.2 SMS statistics

```
{
        size: 1,
        total: 552,
        list:
        [
                {
                        id: "8375",
                        name: "e35b1176a681e93395bbf6701703570f",
                        stats:
                        {
                                total: 4395,
                                wait: 0,
                                on: 0,
                                ok: 3961,
                                ko: 434,
                                no: 0,
                                optout: 0,
                                rep: 0
                        }
                }
        ],
        httpStatusCode: 200
}
```

Copyright