Digitaleo

Email API

Revisions

Revision	Author	Date	Comments
1	pmarechal	01/02/2014	Version initiale de la documentation
2	pmarechal	03/08/2015	Authentification oAuth 2.0
3	pmarechal	11/01/2015	Ajout du lien vers la bibliothèque PHP facilitant l'intégration
4	pmarechal	25/02/2016	English version

Table of contents

1. Overview
    1.1. Authentification
        1.1.1. Retrieving the application ids
        1.1.2. Retrieve an authentication token
        1.1.3. Using the authorization token (access_token)
    1.2. Pagination, sorting and filtering
        1.2.1. Pagination
            1.2.1.1. introduction
            1.2.1.2. Example 1: Limiting results
            1.2.1.3. Example 2: Pagination
        1.2.2. Sorting
            1.2.2.1. Introduction
            1.2.2.2. Exemple
        1.2.3. Limiting the list of attributes returned per resource
            1.2.3.1. Introduction
            1.2.3.2. Example 1
            1.2.3.3. Example 2
    1.3. The lists of resources returned
    1.4. . The various actions on a resource
        1.4.1. Introduction
        1.4.2. Example
        1.4.3. Profil des méthodes en fonction des actions
    1.5. Updating resources
    1.6. Return codes
    1.7. Response formats
        1.7.1. Introduction
        1.7.2. Examples
        1.7.3. Gestion du cross-domain
    1.8. Filters and passing multiple values
    1.9. Integrating our API as PHP
2. Creating an emailing
    2.1. List of parameters to supply in order to create a mailings resource
    2.2. Link for unsubscribing
    2.3. Preview link
    2.4. Links present in the HTML and TEXT portions
    2.5. Specifying a URL or PING
    2.6. Limit in the number of contacts per mailing creation
    2.7. Example: Sending a customized emailing
3. Reading the emailings created
    3.1. List of properties of the Mailing resource
    3.2. Listing mailings
        3.2.1. List of available filters
        3.2.2. Example: Retrieving a mailing from its id
        3.2.3. Return
    3.3. Retrieving deliverability statistics for an emailing
4. Retrieving the list of clickers, openers, unsubscribed contacts, etc.
    4.1. List of properties of the messages resource
    4.2. List of available filters
    4.3. Examples

4.3.1. Example 1: Retrieving the lists of recipients who have received the email
4.3.2. Example 2: Retrieving the list of hardbounced email addresses
4.3.3. Example 3: Retrieving the list of recipients who have unsubscribed
4.3.4. Example 4: Retrieving the list of recipients who have opened the email
4.3.5. Example 5: Retrieving the list of recipients who have clicked on one of the links present either in the HTML portion or in the TEXT portion
5. Finding out which recipient clicked where
5.1. List of properties of the links resource
5.2. List of available filters
5.3. Examples
5.3.1. Retrieving links using an emailing name
5.3.2. Retrieving a particular link
5.3.3. Retrieving the links of an emailing but only those present in the HTML portion
6. Retrieving the status of a message
6.1. Example: Retrieving the current status of message No. 16160662
6.2. Return
6.3. List of the properties of the currentStatus field
Copyright

This document describes the programming interface, or API, of Digitaleo's platform for sending email. It allows you to send mailings and to monitor their statuses.

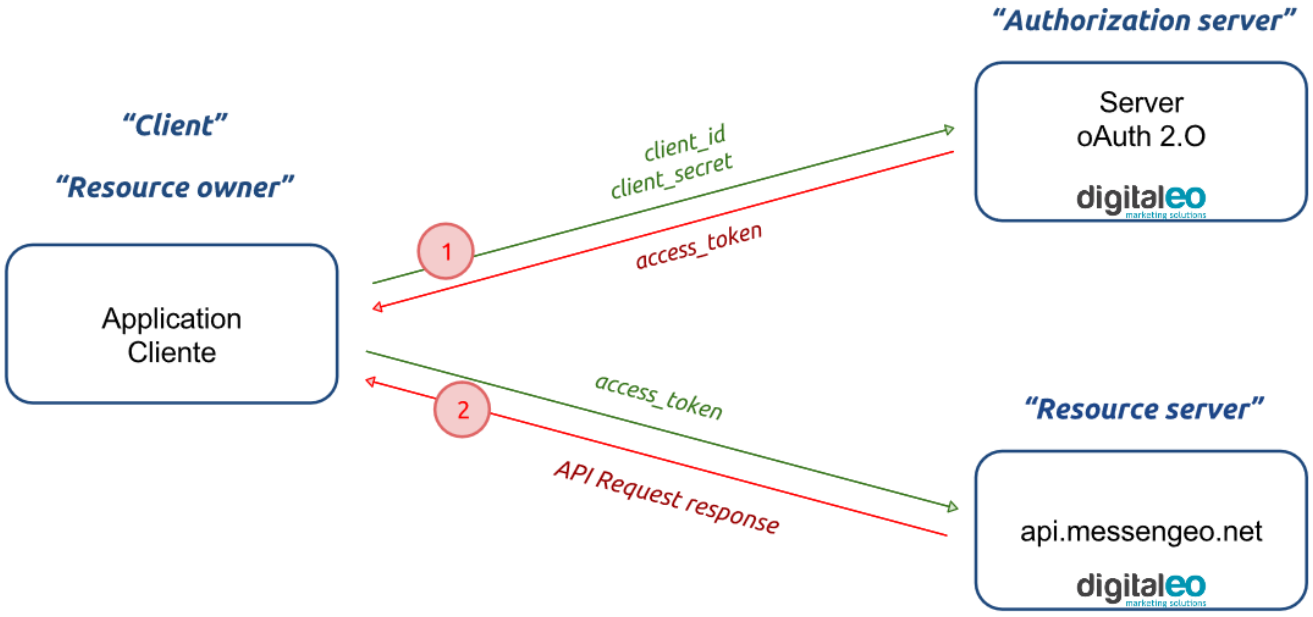
1. Overview

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. Creating an emailing

The resource to create if you want to do sendings is the **mailings** resource

2.1. List of parameters to supply in order to create a mailings resource

Parameter	Type	Description	Required?
text	string	Content in TEXT format • SMS: content of the SMS • EMAIL: text version	yes
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*	yes
subject	string	Subject of the e-mailing *Note: Only for the EMAIL media*	yes
replyContact	string	Reply means of the e-mailing *Note: Only for the EMAIL media*	no
sender	string	Name of the sender	no
media	string	Media used to send messages (EMAIL)	yes
pingUrl	string	Notification url for a status change on one or more messages.	no

contacts	array	Table of contacts (destination email address and customized fields) for this campaign.
		
		This field must have the following tree structure:
		
		`{`
		`{`
		`"recipient":"michel.dupond@gmail.com",`
		`"lastname":"Michel",`
		`"firstname":"Dupont",`
		`},`
		`{`
		`"recipient":"robert.cantin@gmail.com",`
		`"lastname":"Robert",`
		`"firstname":"Cantin",`
		`},`
		`{`	

		"recipient":"jean.durant@gmail.com", "lastname":"Jean", "firstname":"Durant", }, } A concrete example is provided at the end of this documentation. *Note: Only the "recipient" field of each contact is required; it allows you to define the recipient email address of the message.*	

2.2. 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>
```

2.3. 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>
```

2.4. 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>
```

2.5. 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://monserveur.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.

2.6. Limit in the number of contacts per mailing creation

Creating an emailing is currently limited to 1000 contacts. However, when reading statistics, you can consolidate the statistics of several mailings that have the same name.

2.7. Example: Sending a customized emailing

```
curl
  -X POST
  -H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
  -d name=”Mon mailing pour les soldes”
  -d date=”2012-06-29 12:00:00”
  -d media=”EMAIL”
  -d subject=”Soldes le 1er juillet”
  -d html=”<html>Bonjour Mr. #nom#,...</html>”
  -d text=”Bonjour Mr. #nom#,...”
  -d contacts[0][recipient] = “michel.durand@gmail.com”
  -d contacts[0][prenom] = “Michel”
  -d contacts[0][nom] = “DURAND”
  -d contacts[0][recipient] = “robert.cantin@gmail.com”
  -d contacts[0][prenom] = “Robert”
  -d contacts[0][nom] = “CANTIN”
https://api.messengeo.net/rest/mailings
```

3. Reading the emailings created

This section covers how to read the emailings in order to retrieve statistics on deliverability, opening, clicks, unsubscription...

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
subject	string	Subject of the e-mailing
replyContact	string	Reply means of the e-mailing
sender	string	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
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

3.2. Listing mailings

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

3.2.2. Example: 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
```

3.2.3. 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. Retrieving deliverability statistics for an emailing

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
```


```
{
  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
      }
    }
  ],
  httpStatusCode: 200
}
```

The statistics returned in this resource have the following meaning:

Code returned by the API	Description
wait	The email has been taken into account but has not yet been sent
on	The email ha been sent but has not yet been received by the recipient
ko	The email could not be delivered
sb	The email could not be delivered because it is in SOFTBOUNCED (the number of “sb” emails is included in the number of “ko” emails).
hb	The email could not be delivered because it is in HARDBOUNCED (the number of “hb” emails is included in the number of “ko” emails).
no	The final status of the email is not known
ok	The email was delivered to its recipient
opened	The email has been opened
clicked	One of the links in the email was clicked
optout	The user unsubscribed from this email

4. Retrieving the list of clickers, openers, unsubscribed contacts, etc.

The previous section covered how to retrieve the statistics on an emailing; this section covers how to retrieve the list of contacts linked to these statistics. These lists are based on the **messages** resource.

4.1. List of properties of the messages resource

Property	Type	Description
id	string	Id of the mailing
recipient	string	Recipient of the message

4.2. List of available filters

Property	Description
mailingId	Filter according to one or more mailing ids
mailingName	Filter according to the name of the mailing
status	Allows you to select the recipients according to the status of the message that was sent to the recipient

4.3. Examples

4.3.1. Example 1: Retrieving the lists of recipients who have received the email

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

mailingName=soldesdu10mai & status=ok
```

4.3.2. Example 2: Retrieving the list of hardbounced email addresses

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

mailingName=soldesdu10mai & status=hardbounced
```

4.3.3. Example 3: Retrieving the list of recipients who have unsubscribed

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

mailingName=soldesdu10mai & status=optout
```

4.3.4. Example 4: Retrieving the list of recipients who have opened the email

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

mailingName=soldesdu10mai & status=opened
```

4.3.5. Example 5: Retrieving the list of recipients who have clicked on one of the links present either in the HTML portion or in the TEXT portion

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

mailingName=soldesdu10mai & status=clicked
```

5. Finding out which recipient clicked where

Retrieving the links present in the HTML and/or TEXT portion of an emailing is done with the **links** resource. Each one of these resources contains the number of clicks and the list of clickers.

5.1. List of properties of the links resource

Property	Description
code	Unique id of the link
mailingId	Id of the mailing to which the link is attached
url	URL associated with the link
title	Title of the link. The title is retrieved from the HTML content from the title attribute of the <a> tag. If the attribute is not present, the alt attribute is used.
html	HTML content of the <a> tag
part	Source of the link: HTML portion or TEXT portion
position	Position of the link in the HTML portion or TEXT portion
nbClickedThru	Number of times the link was clicked
clickThruRecipients	List of the email addresses of contacts who clicked on the link.

5.2. List of available filters

Property	Description
code	Allows you to retrieve the links by specifying their code.
mailingId	Allows you to retrieve the links that belong to a mailing by specifying its id.
mailingName	Allows you to retrieve the links that belong to a mailing by specifying its name.
part	Allows you to limit the results to the links present either in the HTML portion or in the TEXT portion

5.3. Examples

5.3.1. Retrieving links using an emailing name

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

mailingName=soldesdu10mai
```

5.3.2. Retrieving a particular link

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

code=abcdef
```

5.3.3. Retrieving the links of an emailing but only those present in the HTML portion

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

mailingName=soldesdu10mai & part=html
```

6. Retrieving the status of a message

Recall that a message is a sending to one recipient. An emailing therefore contains n messages.

If you have specified a ping URL (cf. #2.5. Specifying a PING URL) in order to be informed on the change in status of a message and as such you want to retrieve the current status of the message in question, you must consult the **currentStatus** field of the **message** resource.

6.1. Example: Retrieving the current status of message No. 16160662

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

id=16160662&properties=id,currentStatus
```

6.2. Return

```
{
 "size": 1,
 "total": 1,
 "list": [
 {
   "id": "16160662",
   "currentStatus": {
       "code": "DIG12:8",
       "shortKey": "opened",
       "description": "mail opened",
       "date": "2016-01-08 09:58:04",
 }
}
```

6.3. List of the properties of the currentStatus field

Property	Description
code	Detailed code of the status. This is the code that you must provide us with in the event of a problem with a particular message.
shortKey	This is the deliverability status of the message (cf. table in section #3.3)
description	Detailed description of the status
date	Date of the status (date clicked, date opened...)

Copyright