Contacts API

Revisions

Revision	Author	Date	Comment
1	pmarechal	05/03/2015	Version initiale de la documentation
2	pmarechal	03/08/2015	Ajout de la section expliquant comment récupérer les identifiants d'une application depuis app.digitaleo.com
3	pmarechal	01/12/2015	Correction du filtre permettant de lister les contacts d'une liste (listIds => listId)
4	pmarechal	22/03/2016	English version

This document describes the programming interface, or API, of Digitaleo's platform for managing contacts. This API allows you to create/import contacts and also organize them into contact lists, perform segmentation, manage unsubscriptions.

Table of contents

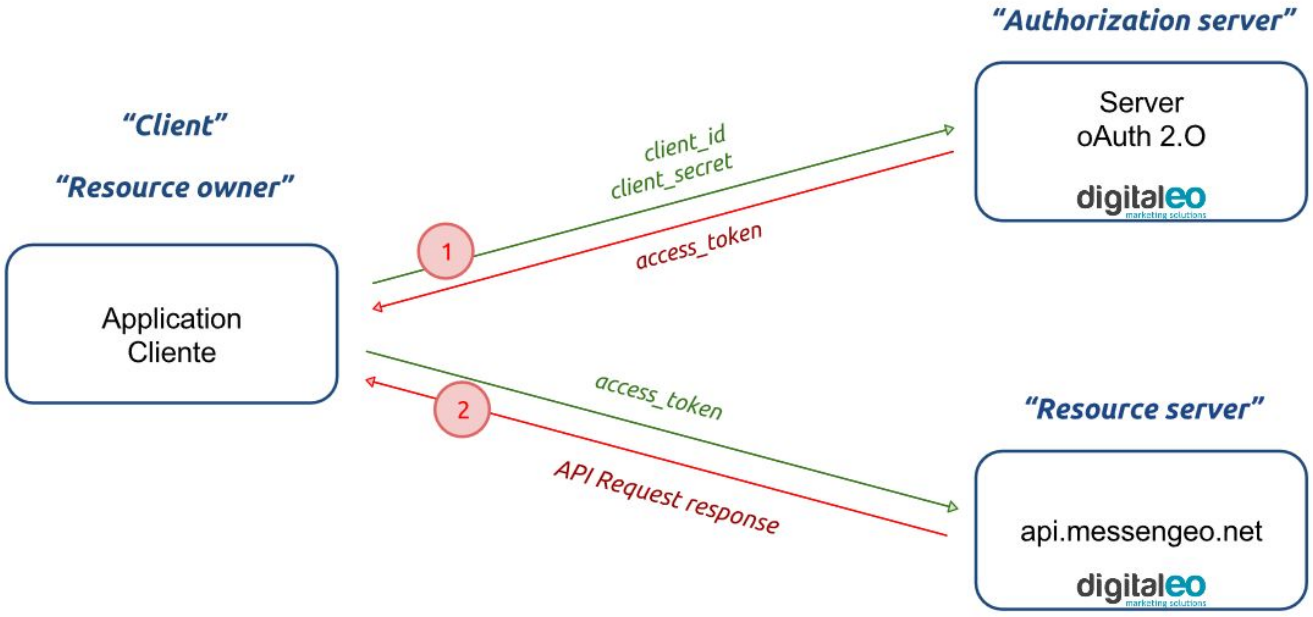
1. Overview

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. Authentication

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. Example

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. Methods profile for each action

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":"New name of the
resource","description":"New description of the resource"}'
```

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/resource
```

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

callback=yourFunctionCallback
```

1.7.3. 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>
  $.jsonp({
      url: 'https://api.messengeo.net/rest/resource.js?callback=?',
      beforeSend: function (request) {
          request.setRequestHeader("Authorization", "Bearer " + ($("#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/resource.json HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: api.messengeo.net
Content-Type: application/x-www-form-urlencoded

id=12,13
```

```
GET /rest/resource.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. The various resources proposed by the API

The contact management API has several resources according to the desired functionality.

Of course, the basic resource is the **contact** resource. It provides the basic structure required for storing the various information about a person (last name, first name, mobile telephone number, email address, date of birth...).

The **list** resource allows contacts to be grouped together. The main interest is to establish distribution lists for sending campaigns, but these lists can also be used to deactivate a set of contacts.

As its name indicates, the **statistic** resource allows you to retrieve statistics on the number of contacts in its database or from a particular list and also the number of contacts that can be reached for an SMS, Email or voice sending.

The **optout** resource corresponds to blocking messages to a contact. Several reasons can result in blocking:

- An unsubscription by the contact himself
- The creator of the contact no longer wants any contact. This happens when a user already has an existing base with a list of unsubscribed persons and the user wants to synchronize it with this contact management API.
- The email address is no valid (hardbounce)
- The recipient has complained about an email and has declared it as SPAM

Finally, the **field** resource represents one of the contact attributes (last name, first name, mobile telephone number, …). This resource is used in the two following contexts:

- It is possible to assign an alias to each contact attribute. These aliases are included when importing a contact file. This allows you to import contact files coming from a CRM export for which the column header would be different from those provided by default by this API.
- This resource also contains statistics concerning the contact attributes:
    - The length that this contact attribute can have. This is useful when creating an SMS campaign with customized fields (as the length of the message is important)
    - The number of contacts for which this attribute is defined. Indeed, it is not necessary to use a customized field in a campaign if the latter is defined for only 10% of the contacts.

3. The contact resource

It provides the basic structure required for storing the various information about a person (last name, first name, mobile telephone number, email address, date of birth...).

3.1. List of properties of the contact resource

Administration du contact

Property	Description
id	Id of the contact
listIds	List of ids of contact lists in which the contact is mentioned
dateUpdated	Date of the last modification for the contact
dateCreated	Date contact created

Means of contact

Property	Description
email	Email address
phone	Telephone number
mobile	Mobile phone number
fax	Fax number
facebookId	Facebook id
twitterId	Twitter id

Common core

Property	Description
civility	Civility
firstname	First name
lastname	Last name
address1	Mailing address: First address line

address2	Mailing address: Second address line
zipcode	Mailing address: Postal code
city	Mailing address: City
state	Mailing address: State
longitude	Mailing address: Longitude
latitude	Mailing address: Latitude
country	Mailing address: Country
birthDate	Date of birth
reference	Client reference of the contact
company	Company

Variable fields

Property	Description
field01	Variable field No. 1
...	...
field15	Variable field No. 15

Optout information

Property	Description
smsOptout	null, 'contact' or 'user'
emailOptout	null, 'contact', 'user', 'hardbounced' or 'spam'
voiceOptout	null, 'contact' or 'user'
voicemailOptout	null, 'contact' or 'user'

Example of the resource in JSON format

```
{
   "id": "123456789",
   "listIds": [
       "2",
       "9",
       "12"
   ],
   "dateUpdated": "2013-01-02 14:00:00",
   "dateCreated": "2013-01-01 12:00:00",
   "email": "john.doo@domain.com",
   "phone": "+332000000001",
   "mobile": "+336000000001",
   "facebookId": "745932955",
   "twitterId": "14656145",
   "civility": "M.",
   "firstName": "John",
   "lastName": "DOO",
   "address1": "1 rue des Champs Elysée",
   "address2": "Bâtiment C",
   "zipcode": "75000",
   "city": "Paris",
   "country": "France",
   "state": NULL,
   "birthDate": "Paris",
   "reference": "France",
   "field01": "M.",
   "field02": "Michel",
   "field03": "DUPONT",
   "...": "1 rue des Champs Elysée",
   "field15": "Bâtiment C",
   "smsOptout": null,
   "voiceOptout": "user",
   "emailOptout": "contact",
}
```

3.2. Reading contacts

3.2.1. List of available filters

Common filters

Filter	Description
id	Allows you to filter according to one or more contact ids
listId	Allows you to filter the contacts according to the contact lists that they are members of

Filters on the contact content (targeting)

Filter	Description
email	Filter according to one or more email addresses
phone	Filter according to one or more fixed telephone numbers
mobile	Filter according to one or more mobile telephone numbers
facebookId	Filter according to one or more Facebook ids
twitterId	Filter according to one or more Twitter ids
civility	Filter according to one or more civilities
firstname	Filter according to one or more first names
lastname	Filter according to one or more last names
address1	Filter according an address line 1
address2	Filter according an address line 2
zipcode	Filter according to one or more postal codes
city	Filter according to one or more cities
state	Filter according to one or more states
country	Filter according to one or more countries
birthDate	Filter according to one or more dates of birth

Complex filters

Filter	Description
complex	This filter allows you to combine several filters with ORs in place of the ANDs. List of available operators: ● "CONTAINS": contains ● "=": is equal to ● "<>": is different from ● "EMPTY": is empty ● "NOT EMPTY": is defined ● "OPTOUT": is deactivated (unsubscription, hardbounced, spam, invalid...). This operator is only available for the mobile, email, phone, facebookId and twitterId attributes

3.2.2. Attributes for a contact returned by default

By default, only the id, email, phone, mobile, facebookId, twitterId attributes are returned. To access the other attributes, they must be specified in the properties parameter. Recall that the default attributes have an alias via the 'DEFAULT' keyword.

For example, to retrieve the default parameters and the information on unsubscription, the following must be specified

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

properties=DEFAULT,smsOptout,emailOptout,voiceOptout,voicemailOptout
```

3.2.3. Examples

3.2.3.1. Example 1: Retrieving a contact from its id

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

id=1234,5678
```

With Curl

```
curl
-X GET
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d id=1234,5678
https://contacts.messengeo.net/rest/contacts
```

3.2.3.2. Example 2: Targeting

Retrieving 10 contacts for which the mobile contains "03" and for which the country is "France"

```
curl
-X GET
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d limit=10
-d complex[logical]='OR'
-d complex[0][field]='mobile'
-d complex[0][operator]='CONTAINS'
-d complex[0][value]='mobile'
-d complex[1][field]='country'
-d complex[1][operator]='='
-d complex[1][value]='FRANCE'
https://contacts.messengeo.net/rest/contacts
```

3.2.3.2. Example 3

Retrieving all of the contacts for whom the email address is deactivated

```
curl
-X GET
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d email[0][operator]='OPTOUT'
https://contacts.messengeo.net/rest/contacts
```

3.3. Creating contacts

3.3.1. List of parameters to supply in order to create contacts

Parameters	Description
contacts	Table of contacts. Each contact is in itself a table key/value. Only the contact data listed hereinbelow is accepted: email, phone, mobile, fax, facebookId, twitterId, civility, firstName, lastName, address1, address2, zipcode, city, state, longitude, latitude, country, birthdate, reference, company, field01... field15

3.3.2. Validity of a contact

For a contact to be valid, one of the following five parameters has to be defined and valid:

- Email address (email)
- Fixed telephone number (phone)
- Mobile telephone number (mobile)
- Facebook id (facebookId)
- Twitter id (twitterId)

For users for whom the uniqueness criterion is the customer reference, the reference column must be completed.

3.3.3. Deduplicating a contact

There are three possible deduplicating criteria:

- No deduplicating
- quintuplet {email, fixed, mobile, facebookid, twitterid}
- customer reference[1] (reference attribute of each contact)

Each contract chooses his deduplicating criterion When a contact is created that already exists in the database (through its deduplicating criterion), the existing contact is updated with the properties of the new contact.

[1]For this uniqueness criterion, any contact whose reference column is not defined will be considered as invalid

3.3.4. Return

This action returns the list of resources created.

```
{
	"list": [
	{
		"email": null,
		"facebookId": null,
		"firstName": "Jean",
		"id": "98117463",
		"lastName": "DUPONT",
		"mobile": null,
		"phone": "+33296300056",
		"twitterId": null
	}
	],
	"size": 1,
	"total": null
}
```

3.3.5. Example

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

contacts[0][civility]=M.&contacts[0][firstName]=Jean&contacts[0][lastName]=DUPONT&
contacts[0][phone]=0296300056
```

With curl

```
curl
-X POST
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d contacts[0][civility]='M.'
-d contacts[0][firstName]='Jean'
-d contacts[0][lastName]='DUPONT'
-d contacts[0][phone]='0296300056'
https://contacts.messengeo.net/rest/contacts
```

3.4. Updating a contact

3.4.1. List of available filters

Filter	Description
id	Allows you to select a contact to be updated according to its id
reference	Allows you to select the contacts to be updated according to its customer reference

3.4.2. List of data of the contact that can be updated

Means of contact
email, phone, mobile, fax, facebookId, twitterId

Common core
civility, firstName, lastName, address1, address2, zipcode, city, state, longitude, latitude, country, birthDate, reference, company

Variable fields
field01...field15

3.4.3. Return

This method returns the number of contacts concerned by the filter passed as input

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

3.4.4. Example

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

id=1234&metaData='{"name":"Mon nouveau nom de famille"}'
```

With Curl

```
curl
-X PUT
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d id=1234
-d metaData='{"name":"Mon nouveau nom de famille"}'
https://contacts.messengeo.net/rest/contacts
```

3.5. Deleting one or more contacts

3.5.1. List of available filters

Filter	Description
id	Allows you to select a contact to be updated according to its id
reference	Allows you to select the contacts to be updated according to its customer reference

3.5.2. Return

This method returns the number of contacts concerned by the filter passed as input

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

3.5.3. Example

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

reference=ABCD
```

With Curl

```
curl
-X DELETE
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d reference=ABCD
https://contacts.messengeo.net/rest/contacts
```

3.6. Removing one or more contacts from sendings (action=optout)

This action allows you to remove one or more contacts from sendings without necessarily deleting them from the contact database. Typically, this action is used when the contact does not go through the unsubscription link, or a SMS STOP but when he has used another means.

3.6.1. List of parameters

Parameter	Description
contactIds[1]	Ids of the contacts that one wants to remove from sendings
listIds[1]	Ids of the lists of contacts that one wants to remove from sendings
media[2]	'SMS', 'EMAIL', 'VOICE' or 'VOICEMAIL' or a combination of the three.

[1] Once of these two parameters is required

[2] This parameter is required

3.6.2. Means of contact taken into account

For a contact whose email, phone and mobile fields are present, if the media filter is not used, this service will unsubscribe:

- The Email address for the EMAIL media
- The mobile telephone number for the SMS and VOICEMAIL media
- The fixed telephone number for the VOICE media

3.6.3. Return

This action returns the number of contacts removed from sendings

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

3.6.4. Example 1

Removing three contacts from sendings regardless of the media

```
curl
-X POST
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d contactIds='12,15,20'
-d media='SMS,EMAIL,VOICE,VOICEMAIL'
https://contacts.messengeo.net/rest/contacts?action=optout
```

3.6.5. Example 2

Removing all of the contacts of list No. 3 from sendings for the SMS and VOICE media

If you have a file that contains the list of all of your unsubscribed contacts, this is then the example to follow for removing them from sendings on the Digitaleo side.

```
curl
-X POST
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d listIds='3'
-d media='SMS,VOICE'
https://contacts.messengeo.net/rest/contacts?action=optout
```

3.7. Reactivating one or more contacts from sendings (action=reactivate)

This action allows you to reactivate one or more contacts for sendings. Of course, if the contact has unsubscribed since the message that he received (unsubscription link for email, SMS STOP, …), this action will not reactivate it. The action makes it possible to reactivate only those contacts that were deactivated via the optout action described hereinabove.

3.7.1. List of parameters

Parameter	Description
contactIds[1]	Ids of the contacts that one wants to reactivate
listIds[1]	Ids of the lists of contacts that one wants to remove from sendings
media[2]	'SMS', 'EMAIL', 'VOICE' or 'VOICEMAIL' or a combination of the three.

[1] Once of these two parameters is required

[2] This parameter is required

3.7.2. Means of contact taken into account

For a contact whose email, phone and mobile fields are present, if the media filter is not used, this service will reactivate:

- The Email address for the EMAIL media
- The mobile telephone number for the SMS and VOICEMAIL media
- The fixed telephone number for the VOICE media

3.7.3. Return

This action returns the number of contacts reactivated

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

3.7.4. Example 1

Reactivating three contacts for sendings regardless of the media

```
curl
-X POST
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d contactIds='12,15,20'
-d media='SMS,EMAIL,VOICE,VOICEMAIL'
https://contacts.messengeo.net/rest/contacts?action=reactivate
```

3.7.5. Example 2

Reactivating all of the contacts of list No. 3 for sendings for the SMS and VOICE media

```
curl
-X POST
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d listIds='3'
-d media='SMS,VOICE'
https://contacts.messengeo.net/rest/contacts?action=optout
```

4. The list resource

The list resource allows contacts to be grouped together. The main interest is to establish distribution lists for sending campaigns, but these lists can also be used to deactivate a set of contacts.

4.1. List of properties of the list resource

Property	Description
id	Id of the list of contacts
name	Name of the list of contacts
description	Description of the list of contacts
stats	Statistics of the list of contacts (cf. paragraph for the details)
importErrors	Link to the CSV file containing the list of invalid contacts as well as the reason for which they were not imported. If the list was not created from a file, this parameter is null.
importLines	Number of contacts present in the file that was used to create this list. If the list was not created from a file, this parameter is null.
importInvalid	Number of contacts present in the file that was used to create this list but which were not important because they are invalid. If the list was not created from a file, this parameter is null.
importStatus	Flag making it possible to know whether or not a list is in the process of being imported. ● "on": The file is being processed ● "10": percentage processed of the file ● "ok": The file is fully processed
dateUpdated	Date of the last modification for the contact
dateCreated	Date of creation of the contact

By default, only the id, name and description attributes are returned. To access the other attributes, they must be specified in the properties parameter. Recall that the default attributes have an alias via the 'DEFAULT' keyword.

List of statistics returned

Property	Description
subscribers	Number of contacts present in the list
sms	Number of contacts that can be reached for an SMS sending. As such, the following are not counted ● Contacts that have unsubscribed for this media ● Contacts that the customer has deliberately unsubscribed ● Deleted contacts
email	Number of contacts that can be reached for an email sending. As such, the following are not counted ● Contacts that have unsubscribed for this media ● Contacts that the customer has deliberately unsubscribed ● Deleted contacts ● Hardbounced contacts
voice	Number of contacts that can be reached for a PUSH voice sending. As such, the following are not counted ● Contacts that have unsubscribed for this media ● Contacts that the customer has deliberately unsubscribed ● Deleted contacts
voicemail	Number of contacts that can be reached for Message on answering machine sending. As such, the following are not counted ● Contacts that have unsubscribed for this media ● Contacts that the customer has deliberately unsubscribed ● Deleted contacts

Example of the resource in JSON format

```
{
   "id": "123456789",
   "name": "Ma liste de contacts",
   "dateUpdated": "2013-01-02 14:00:00",
   "dateCreated": "2013-01-01 12:00:00",
   "importErrors": "https://contacts.messengeo.net/rest/list.csv...",
   "importLines": "204586",
   "importInvalid": "50007",
   "importStatus": "ok",
   "stats": {
       "subscribers": "103456",
       "sms": "100003",
       "email": "45689",
       "voice": "564",
       "voicemail": "1025",
   }
}
```

4.2. Reading contact lists

4.2.1. List of available filters

Filter	Description
id	Allows you to select the lists to be retrieved from their id

4.2.2. Example

Retrieving two lists from their ids:

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

id=12,65
```

4.3. Creating a list of contacts

4.3.1. List of parameters to supply in order to create contacts

Parameters	Description
name	Name of the list of contacts This parameter is required
description	Description of the list of contacts

4.3.2. Return

This action returns the list of contact lists created.

```
{
       "list": [
       {
              "id": "7606",
              "name": "Ma liste de diffusion"
       }
       ],
       "size": 1,
       "total": null
}
```

4.3.3. Example

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

name='Ma liste de diffusion'
```

With Curl

```
curl
-X POST
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d name='Ma liste de diffusion'
https://contacts.messengeo.net/rest/lists
```

4.4. Updating a list of contacts

4.4.1. List of available filters

Filter	Description
id	Allows you to select the lists to be updated according to their id

4.4.2. List of data of the contact list that can be updated

Filter	Description
name	Name of the distribution list
description	Description of the distribution list

4.4.3. Return

This action returns the number of lists concerned by the filter passed as input

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

4.4.4. Example

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

id=1234&metaData='{"name":"New name of the list","description":"New description of
the list"}'
```

With Curl

```
curl
-X PUT
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d id=1234
-d metaData='{"name":"New name of the list","description":"New description of the
list"}'
https://contacts.messengeo.net/rest/lists
```

4.5. Deleting one or more lists of contacts

4.5.1. List of available filters

Filter	Description
id	Allows you to select the lists to be deleted according to their id

4.5.2. Return

This method returns the number of contacts concerned by the filter passed as input

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

4.5.3. Example

```
DELETE /rest/lists HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: contacts.messengeo.net
Content-Type: application/x-www-form-urlencoded

id=12,78
```

With Curl

```
curl
-X DELETE
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d id=12,78
https://contacts.messengeo.net/rest/lists
```

4.6. Merging lists of contacts (action=merge)

This action allows you to create a list of contacts using two (or more) contact lists.

4.6.1. List of parameters

Parameter	Description
listIds	List of the list ids
name	Name of the list of contacts to be created

Note: If one of the contact list ids is not correct, the creating of the new list will not take place. An error is returned indicating the ids with errors.

4.6.2. Return

This action returns the list created following the merger of the other lists:

```
{
   "size": 1,
   "total": null,
   "list": [
   {
       "id": "2671",
       "name": "Ma liste fusionnée"
   }
   ]
}
```

4.6.3. Exemple

```
POST /rest/lists?action=merge HTTP/1.1
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpZCI6ImYyMzE2…
Host: contacts.messengeo.net
Content-Type: application/x-www-form-urlencoded

listIds=12,78,890&name='Ma liste fusionnée'
```

Avec Curl

```
curl
-X POST
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d listIds=12,78,890
-d name='Ma liste fusionnée'
https://contacts.messengeo.net/rest/lists?action=merge
```

4.7. Importing a contact list from a file (action=import)

This action comprises two steps:

- Creating a list of contacts.
- Creating contacts (or updating contacts for those that were already present in the database).

With the return of the REST call, only the list of contacts was created. The contacts are created subsequently asynchronously. It is possible to monitor the unfolding of the import via the import* attributes of the list resource. As such, when importing a large file, the response time corresponds approximately to the upload time of the file on the platform managing the contacts, with the file being processed afterwards.

This action is to be favored (with respect to the create action on a contact) when there is a large number of contacts to create or synchronize.

4.7.1. List of parameters to supply

Parameter	Description
name	Name of the list of contacts to be created
ignoreFirstLine	Indicates if the first line is a column header line (ignoreFirstLine=true) or if it is a contact (ignoreFirstLine=false)
contactFile	Name of the list of contacts

4.7.2. Format of the files to be imported

The csv and excel (xls, xlsx) formats are authorized. It is also possible to zip the file in order to save time during the upload. The file cannot exceed a size of 20Mb.

With regards to the format of the column headers, two things are possible:

- Either the CSV file does not contain any column headers and then the order of the columns of the contacts must comply with the following order: Civility, First name, Last name, Email, Fixed Telephone, Mobile, Fax, Address 1, Address 2, Postal code, City, State, Country, Date of birth, Company, Reference, Field 1, …, Field 15
- Or the file contains column headers and then, the order of the column is not important as long as the name of the column header is complied with.

4.7.3. Return

This action returns the list created:

```
{
   "size": 1,
   "total": null,
   "list": [
   {
       "id": "2671",
       "name": "Ma liste importée"
   }
   ]
}
```

4.7.4. Example

```
curl
-X POST
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d name='Ma liste importée'
-d ignoreFirstLine='true'
-F contactFile=@/path/to/file/liste.csv
https://contacts.messengeo.net/rest/lists?action=import
```

4.8. Adding one or more contacts to a list (action=contactadd)

4.8.1. List of available filters

Filter	Description
id	Id of the list to which the contacts are to be added *This parameter is required*

4.8.2. Specifying the contact or contacts to be added

This data is to be passed in the metaData parameter (cf. examples).

Parameter	Description
contactIds	Ids of the contacts that one wants to add to a contact list *This parameter is required*

4.8.3. Return

This action returns the number of contacts added in the list

```
{
   count: 3,
}
```

4.8.4. Example

```
curl
-X POST
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d id=7606
-d metaData='{"contactIds":"81583724,81583721,81583720" }'
https://contacts.messengeo.net/rest/lists?action=contactadd
```

4.9. Removing one or more contacts from a list (action=contactremove)

4.9.1. List of available filters

Filter	Description
id	Id of the list from which the contacts are to be removed *This parameter is required*

4.9.2. Specifying the contact or contacts to be added

This data is to be passed in the metaData parameter (cf. examples).

Parameter	Description
contactIds	Ids of the contacts that one wants to remove from a contact list *This parameter is required*

4.9.3. Return

This action returns the number of contacts removed from the list

```
{
   count: 3,
}
```

4.9.4. Example

```
curl
-X POST
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d id=7606
-d metaData='{"contactIds":"81583724,81583721,81583720" }'
https://contacts.messengeo.net/rest/lists?action=contactremove
```

5. The statistic resource

As its name indicates, the statistic resource allows you to retrieve statistics on the number of contacts in its database or from a particular list and also the number of contacts that can be reached for an SMS, Email, Push voice or message on answering machine sending.

5.1. List of properties of the statistic resource

Property	Description
subscribers	Number of contacts present in the list
sms	Number of contacts that can be reached for an SMS sending. As such, the following are not counted ● Contacts that have unsubscribed for this media ● Contacts that the customer has deliberately unsubscribed ● Deleted contacts
email	Number of contacts that can be reached for an email sending. As such, the following are not counted ● Contacts that have unsubscribed for this media ● Contacts that the customer has deliberately unsubscribed ● Deleted contacts ● Hardbounced contacts
voice	Number of contacts that can be reached for a PUSH voice sending. As such, the following are not counted ● Contacts that have unsubscribed for this media ● Contacts that the customer has deliberately unsubscribed ● Deleted contacts
voicemail	Number of contacts that can be reached for Message on answering machine sending. As such, the following are not counted ● Contacts that have unsubscribed for this media ● Contacts that the customer has deliberately unsubscribed ● Deleted contacts

Example of the resource in JSON format

```
{
   "subscribers": "103456",
   "sms": "100003",
   "email": "45689",
   "voice": "564",
   "voicemail": "1065",
}
```

5.2. Retrieving statistics

5.2.1. Filters available

Filter	Description
listId	Filter according to one or more contact list ids

5.2.2. Additional parameter

Filter	Description
priorities	This parameter allows you to know the number of email addresses, mobile telephone or fixed telephones in the framework of substitution[1]. Possible values of the priorities table: SMS, EMAIL, VOICE, VOICEMAIL *Note: For the media not present in the priorities list, the number of contacts that can be reached for these forms of media will be equal to 0.*

[1]Substitution is the principle of prioritizing the communication channels for a multichannel marketing campaign. For example, building a campaign where SMS is prioritized with respect to email will result in sending an SMS to all of the contacts that have a mobile telephone, then in sending an email to all of the contacts that have an email address but no mobile telephone number.

5.2.3. Return

Contrary to the read methods of the other resources, it is not a list that is returned but a single resource.

5.2.4. Example

Retrieving the number of mobile telephones or email addresses that can be reached by prioritizing SMS

```
curl
-X GET
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d listId=7606
-d priorities=SMS,EMAIL
https://contacts.messengeo.net/rest/statistics
```

6. The field resource

The field resource is the structure that allows you to manage the contact attributes (last name, first name, mobile telephone number...).

This resource is used in the two following contexts:

- It is possible to assign an alias to each contact attribute. These aliases are included when importing a contact file. This allows you to import contact files coming from a CRM export for which the column header would be different from those provided by default by this API.
- This resource also contains statistics concerning the contact attributes:
  - The length that this contact attribute can have. This is useful when creating an SMS campaign with customized fields (as the length of the message is important)
  - The number of contacts for which this attribute is defined. Indeed, it is not necessary to use a customized field in a campaign if the latter is defined for only 10% of the contacts.

6.1. List of properties of the field resource

Property	Description
name	Name of the contact attribute
max	Maximum length that this contact attribute can have
count	Number of contacts for which this attribute is defined
alias	Alias of the contact attribute

Example of the resource in JSON format

```
{
   "name": "field01",
   "max": "11",
   "count": "1234556",
   "alias": "numéro de plaque",
}
```

6.2. Retrieving the alias of each contact attribute

6.2.1. List of available filters

Filter	Description
name	Filter according to one or more contact attribute names
alias	Filter according to one or more contact attribute aliases

6.2.2. Small note on default aliases

When a new user connects to the contact management API, the aliases of the columns that are applied are as follows:

name	alias
civility	Civility
firstName	First name
lastName	Last name
email	Email
phone	Fixed Telephone
mobile	Mobile
fax	Fax
address1	Address 1
address2	Address 2
zipcode	Postal code
city	City
state	State
country	Country
birthDate	Date of birth
reference	Reference
company	Company

field01	Field 1
...	....
field15	Field 15

All of these aliases can be customized by contact Digitaleo support.

6.2.3. Return order

The fields are returned in the following order:
civility, firstName, lastName, email, phone, mobile, fax, address1, address2, zipcode, city, state, country, birthDate, company, reference, field01... field15.

6.2.4. Example

Retrieving all the attribute aliases:

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

6.3. Retrieving the statistics on a contact attribute (action=statistics)

The statistics action allows you to retrieve for each contact attribute

- The maximum length that this contact attribute can have
- The number of contacts for which this attribute is defined.

6.3.1. List of available filters

Filter	Description
name	Filter according to one or more contact attribute names
listId	Filter according to contact list id

6.3.2. Example 1

Retrieving the maximum length that the "field01" attribute can have for the contact list 123

```
curl
-X GET
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d listId=123
-d name=field01
-d properties=name,max
https://contacts.messengeo.net/rest/fields?action=statistics
```

6.3.3. Example 2

Retrieving the number of contacts for which each contract attribute of the list 123 is defined

```
curl
-X GET
-H "Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbG...ccOqbVow8xOQyQ"
-d listId=123
-d properties=name,count
https://contacts.messengeo.net/rest/fields?action=statistics
```

Copyright

All of this code is governed by French and international legislation on copyright and intellectual property. All reproduction rights reserved, including for documents that can be downloaded and iconographic and photographic representations. Reproducing all or a portion of this code on any support whatsoever is strictly forbidden unless authorization is obtained in writing from Digitaleo.