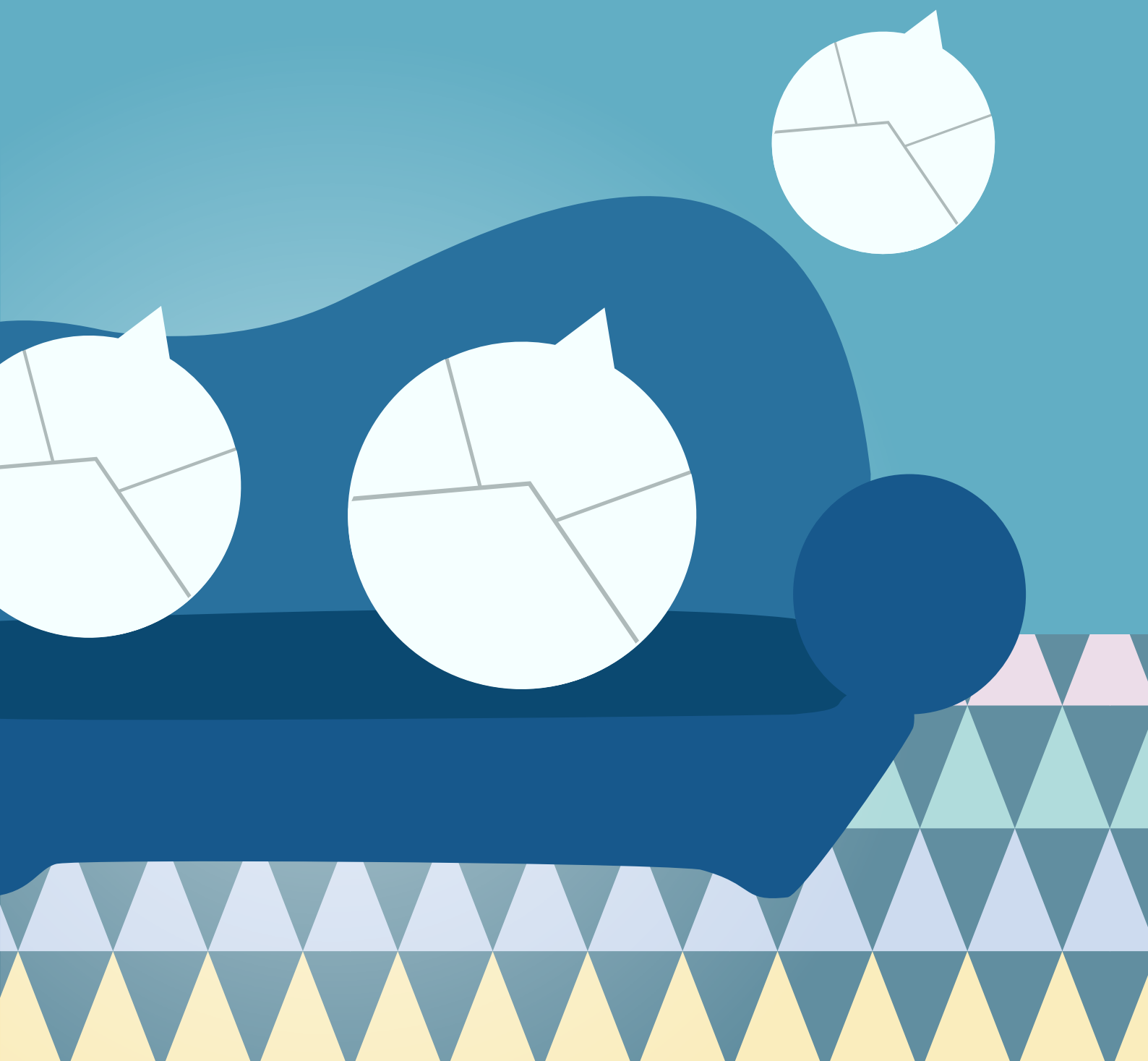
SMS IN

a Geodrop guide



geodrop

SMS In® 2.3

Geodrop API for receiving SMS messages

Quick integration guide

RESTful API User Guide v1.3-EN, 21 September 2012

Contents

Changelog

Version	When	What	Who
1.0	Aug, 3 2012	First official version	Giuseppe Costa
1.1	Aug, 17 2012	Changed document name.	Giuseppe Costa
1.2	Aug, 28 2012	Minor changes	Stefano di Sandro
1.3	Sept, 21 2012	Minor changes. Updated SMS In module version number.	Giuseppe Costa

Abstract

This guide provides useful information for a quick set up of a system enable to receive SMS messages via "SMS IN" API module.

The description of the data sent and received by the API is provided using *JSON Schema* (http://tools.ietf.org/html/draft-zyp-json-schema-03), a formalism for the schematic definition of JSON objects, created by Kris Zyp (http://twitter.com/#!/kriszyp).

To build the examples we used the cURL utility (http://curl.haxx.se/).

System overview

This chapter is intended to provide an overview of the various components of the module for receiving SMS messages, without going into details, in order to enable the reader to understand roughly how it works.

Shared and dedicated numbers

Geodrop can receive SMS messages sent from mobile phones through a set of dedicated phone numbers. Each phone number can be dedicated for a single user or shared between two or more of them.

To be able to receive SMS messages through Geodrop you have to activate one or more numbers on your account.

While shared numbers can be freely activated from the web interface or via API, to activate dedicated numbers you must contact Geodrop sales support (sales@geodrop.com) to get a custom quote.

Shared numbers and keywords

In order to properly route received SMS messages to their final destination (your Inbox), since the numbers are common to a set of users, it is necessary that every SMS message begins with a special word: the *keyword*.

Each keyword is associated to a single user through a registration process operated independently by users themselves. According to the initial word of each SMS message is thus possible to uniquely identify the recipient among all users sharing a given number.

The SMS messages sent to shared numbers that begin with words not recorded as a keyword will be lost as the system is unable to determine the intended recipient and to proceed to relay.

Keywords are therefore special words, recorded by each user on a given number (resulting in a configured Inbox) and having the following characteristics:

- a keyword is registered by a user on one of the shared numbers (this is called *setting an Inbox)* ;
- a user may register more than one keyword on a given shared number (multiple Inbox on the same number);
- a user may register the same keyword on more than one shared number (multiple Inbox on different numbers);
- a given keyword may be registered on a given shared number by one and only one user. Therefore, if the user “U” has registered the keyword "foo" on the number +393202041300, nobody else will do. This ensures that all SMS messages sent to the number +393202041300 that begin with the word "foo" can be properly routed to the user “U”;

- two users can register the same keyword only on different shared numbers. This means that if the user “U1” has registered the keyword “foo” on the number +393202041300, a user “U2” can register the keyword “foo” on the number +393399941199. The effect is that all SMS messages that begin with the keyword "foo" sent to the number +393202041300, will be forwarded to the user U1 and all SMS messages that begin with the keyword "foo" sent to the number +393399941199, will be forwarded to the user U2;

- a user can register the same keyword on ALL available shared numbers. This effectively provides the user with the exclusive use of the keyword on all shared numbers, since all SMS messages received by Geodrop, on any shared number, will be forwarded to him;

- a given keyword may be *active* or *inactive*. An inactive keyword is not used by Geodrop for message forwarding. This means that all SMS messages that have, as the first word, an inactive keyword, are discarded. Keyword activation and deactivation is performed in autonomy by the user;

- a user pays a periodic fee for each of its registered and active keywords. Inactive keywords are not subject to the payment of the fee;

- keywords can remain inactive for up to 30 days. After this period the system will proceed to their automatic deletion to prevent abuses;

- SMS messages sent to a given shared number, whose first word is a deleted or inactive keyword, are discarded by the system and will not be recoverable;

- a keyword that was automatically deleted by the system or explicitly by the user, is made available again for registration after a period of 40 days (quarantine) to preserve system to route old Inbox destinated incoming messages to the new one.

Dedicated numbers

Users who activates a dedicated number does not need to register a keyword, since all SMS messages sent to a dedicated number are correctly routed to the sole owner and stored in a default Inbox.

Owners of dedicated numbers may, however, register keywords to classify incoming SMS messages as needed.

Inbox

A pair <number, keyword> is called *Inbox*. An Inbox belongs to the single user who registered the keyword on the number. The number may be either shared or dedicated.

All SMS messages sent to a given number (shared or dedicated) and beginning with a specific keyword are stored in the Inbox identified by the keyword itself.

For a dedicated number there is always a default Inbox, where Geodrop stores all messages which cannot be routed to anyone of the Inboxes identified by a keyword.

Deleting a keyword, will delete the corresponding Inbox and all the messages stored in it. The default Inbox for dedicated numbers cannot be deleted, because it's not related to any keyword.

Rules

Geodrop can perform several actions on incoming SMS messages:

- send a response back to the sender;
- forward the SMS message to a given mobile number, URI or e-mail address;
- insert the sender's *MSISDN* (Mobile Subscriber Integrated Services Digital network Number... aka “mobile phone number”) into a group of contacts in the user's contacts books;
- remove the sender's MSISDN from a group of contacts in the user's contacts books.

An action can be performed on each incoming message or only on messages that match certain conditions.

A pair <condition, action> is called *rule*. If the condition of a rule is empty, the corresponding action is intended to be performed on every incoming messages. Otherwise, the action will be performed only on incoming messages matching the specified condition.

How to set-up a simple service for receiving SMS messages

To start receiving SMS messages through Geodrop's “Sms in” API module you have to:

1. choose a dedicated or shared number to which SMS messages will be sent
2. register one or more keywords on the chosen number (optional if you choose a dedicated number)

If you want to use a shared number, both tasks can be accomplished using the API of SMS In module. Otherwise, to complete the first one, you have to contact Geodrop sales support (sales@geodrop.com) before any other actions.

Choosing a shared number

To retrieve the list of all available numbers you must call the **lanes** method as shown in the next example:

```
curl -X GET -H "Authorization: Bearer <access-token>"
https://api.geodrop.net/in/1/sms/1/lanes
```

The “<access-token>” is a 32 hexadecimal digit string used in the Oauth2 authentication and authorization process. To know how to get the access token we remand to the Oauth2 documentation on Geodrop website.

In response to the https request shown above, you will receive a JSON string like the following:

```
{"result": "0", "msisdns": [{"lane": "3404324043", "user_id": "-1", "msisdn": "(+39)
340.43.240.43", "active": 1, "shared": 1, "id": 0}, {"lane": "3202041300", "user_id": "-
1", "msisdn": "(+39) 320.20.41.300", "active": 0, "shared": 1, "id": 1}]}
```

Let's make it more readable:

```
{
```

```
  "result":"0",
  "msisdns":[
    {
      "lane":"3404324043",
      "user_id":"-1",
      "msisdn":"(+39) 340.43.240.43",
      "active":1,
      "shared":1,
      "id":0
    },
    {
      "lane":"3202041300",
      "user_id":"-1",
      "msisdn":"(+39) 320.20.41.300",
      "active":0,
      "shared":1,
      "id":1
    }
  ]
}
```


The **result** field has value “0” if there were no errors. In case of error, please check the result value against the “Staus code” table in the Appendix.

The **msisdns** field contains the list of the available numbers. For each number are returned the following informations:

- **lane**: is the standard representation of the number;
- **user_id**: if the number is dedicated and belongs to you, this field contains your own Geodrop user identifier. Otherwise the value is “-1”;
- **msisdn**: is a free human-readable representation of the lane. You must use this value when calling methods requiring a reference to the number;
- **active**: “1” if the number is active, “0” if it is inactive. Keywords cannot be registered on inactive numbers;
- **shared**: “1” if the number is shared, “0” if it is dedicated;
- **id**: the Geodrop internal unique identifier of the number.

Obviously, if you have not contacted the sales support to activate a dedicated number, you will receive a list containing only shared numbers.

Registering a keyword

Once the id of the number you would like to use to set-up your service is retrieved, you may define one or more keywords which will be used to split the received messages in the corresponding storing

Inbox. **If you choose a shared number this step is mandatory**.

WARNING! Registering and maintaining a keyword has a cost. You can find the cost for keyword registration an the periodic fee on Geodrop website.

To register your first keyword, you have to call the **keywords** method in http PUT, like in the following example:

```
curl -X PUT -H "Authorization: Bearer <access-token>" --data <request-body>
https://api.geodrop.net/in/1/sms/1/keywords
```

The "<request-body>" is a JSON string representing an object containing all the information needed to register the keyword. The JSON object has the following attributes:

- **sub** (mandatory): the keyword to be registered;
- **num** (optional): the number on which the keyword will be registered. If missing, the keyword will be registered on the first number;
- **status** (optional): used to set the keyword in status active or inactive simultaneously with the creation. The accepted values are: ACTIVE (the default) or SUSPENDED.

A complete request body is shown below:

Raw

```
{"sub":"testkey3", "status":"SUSPENDED", "num":"(+39) 340.43.240.43"}
```

Indented

```
{
  "sub": "testkey3",
  "status": "SUSPENDED",
  "num": "(+39) 340.43.240.43"
}
```

If you try to register a keyword already registered on the chosen number, you will get an error like this:

Raw

Keyword registered by a different user

```
{"result": "-2", "details": {"lines": "0", "error": "keyword already in"}}
```

Keyword already registered by the current user

```
{"result": "-3", "details": {"lines": "0", "error": "a keyword is activable only a time
per user!"}}
```

Indented

Keyword registered by a different user

```
{
  "result": "-2",
  "details": {
    "lines": "0",
```

```
    "error": "keyword already in"
  }
}
```

Keyword already registered by the current user

```
{
  "result": "-3",
  "details": {
    "lines": "0",
    "error": "a keyword is activable only a time per user!"
  }
}
```

If the registration process is successful, the response will contain the unique identifier of the registered keyword (field **id** of the **details** object), like in the following example:

Raw

```
{"result": "0", "details": {"errors": "", "lines": "1", "id": 1}}
```

Indented

```
{
  "result": "0",
  "details": {
    "errors": "",
    "lines": "1",
    "id": 1
  }
}
```

Checking registered keywords

To retrieve all your registered keywords and their details, you may call in http GET the same method used for registration:

```
curl -X GET -H "Authorization: Bearer <access-token>"
https://api.geodrop.net/in/1/sms/1/keywords
```

The response will be like the following:

Raw

```
{"result": "0", "numbers": [{"status": "ACTIVE", "last_pay": "2012-04-12 08:13:31",
"user_id": "5017", "sub": "testkey", "num": "(+39) 340.43.240.43", "activation_date":
"2012-04-12 08:13:29", "last_activation_date": "2012-04-12 08:13:29", "id": 1,
"sd_date": ""}, {"status": "SUSPENDED", "last_pay": "2012-04-12 11:56:01", "user_id":
"5017", "sub": "testkey2", "num": "(+39) 340.43.240.43", "activation_date": "2012-04-12
11:56:01", "last_activation_date": "", "id": 2, "sd_date": "2012-04-12 11:56:01"}]}
```

Indented

```
{
  "result": "0",
  "numbers": [
    {
```

```
      "status": "ACTIVE",
      "last_pay": "2012-04-12 08:13:31",
      "user_id": "5017",
      "sub": "testkey",
      "num": "(+39) 340.43.240.43",
      "activation_date": "2012-04-12 08:13:29",
      "last_activation_date": "2012-04-12 08:13:29",
      "id": 1,
      "sd_date": ""
    },
    {
      "status": "SUSPENDED",
      "last_pay": "2012-04-12 11:56:01",
      "user_id": "5017",
      "sub": "testkey2",
      "num": "(+39) 340.43.240.43",
      "activation_date": "2012-04-12 11:56:01",
      "last_activation_date": "",
      "id": 2,
      "sd_date": "2012-04-12 11:56:01"
    }
  ]
}
```


The following informations are reported for each registered keyword:

- **status**: indicates whether the keyword is “ACTIVE” or “SUSPENDED”. If the keyword is to be cancelled, the status may be "ALERTED" (see below “Activate, suspend and delete a keyword”);
- **last_pay**: the date of the last payment of the fee for this keyword;
- **user_id**: id of the user who registered the keyword;
- **sub**: the keyword;
- **num**: the number on which the keyword is registered;
- **activation_date**: the date of first activation for the keyword;
- **last_activation_date**: the date of the last activation for the keyword;
- **id**: unique identifier for the keyword;
- **sd_date**: if the keyword's status is “SUSPENDED”, this field specifies the suspension date otherwise its value is not relevant.

Managing received SMS messages

Once you have registered one or more keywords, you can start receiving SMS messages through Geodrop.

Suppose to send the following SMS:

- **sender**: +39 333 33 33 333;
- **recipient**: +39 340 43 240 43;
- **message text**: testkey first message.

You will find the message in the Inbox corresponding to the keyword “testkey” registered on the shared number “(+39) 340.43.240.43”.

Retrieving messages

To retrieve a list of the messages in a given Inbox, you must call the **Inbox** method in http GET, passing the shared number and the keyword (those informations uniquely identify the Inbox) as query string parameters. Accepted parameters are:

- **num** (mandatory): the number on which the keyword which identifies the Inbox is registered;
- **sub** (mandatory): the keyword the keyword which identifies the Inbox;
- **msg_id** (optional): the unique identifier of a message.

Here is an example of method call in which only mandatory parameters are specified:

```
curl -X GET -H "Authorization: Bearer <access-token>"
'https://api.geodrop.net/in/1/sms/1/Inbox?num=(%2B39)%20340.43.240.43&sub=testkey'
```

Both parameters are mandatory. Notice that the number representation must be urlencoded for the call to work properly.

If your message was successfully delivered to Geodrop, you will get the following response to the above method call:

Raw

```
{"messages": [{"text": "testkey first message", "msg_id": "10", "number_id": 1, "time":
"2012-05-03 08:41:49", "operator": "VODA", "sender_num": "+393333333333"}], "result":
"0"}
```

Indented

```
{
  "messages":[
    {
      "text":"testkey first message",
      "msg_id":"10",
      "number_id":1,
      "time":"2012-05-03 08:41:49",
```

```
      "operator":"VODA",
      "sender_num":"+393333333333"
    }
  ],
  "result":"0"
}
```

The **messages** property of the JSON object contains the list of all messages received through the shared number +39 340 43 240 43 for the keyword testkey. For each message you will get the following informations:

- **text**: the message text;
- **msg_id**: the internal unique identifier of the message;
- **number_id**: the internal unique identifier for the keyword;
- **time**: date and time of receipt of the message;
- **operator**: note that thi is the operator which gives the shared number to Geodrop and not the operator of the sender's mobile number;
- **sender_num**: the mobile number from which the message was sent.

If you call this method passing the unique identifier of a message in the **msg_id** parameter, you will get the details of the specified message only.

Deleting messages

To delete messages from your Inbox, you must call the **Inbox** method in http DELETE. Accepted parameters are:

- **num** (mandatory): the number on which the keyword which identifies the Inbox is registered;
- **sub** (mandatory): the keyword the keyword which identifies the Inbox;
- **msg_id** (optional): the unique identifier of the message to delete

WARNING! If only the **num** and **sub** parameters are specified, all messages in the Inbox are deleted.

The following method call deletes only the message message with id “10” from the Inbox identified by the keyword “testkey” and registered on the shared number “(+39) 340.43.240.43”:

```
curl -X GET -H "Authorization: Bearer <access-token>"
'https://api.geodrop.net/in/1/sms/1/Inbox?
num=(%2B39)%20340.43.240.43&sub=testkey&msg_id=10'
```

In case of success, the response is like that shown above:

Raw

```
{"result": "0", "details": {"errors": "", "lines": "1"}}
```

Indented

```
{
  "result": "0",
  "details": {
    "errors": "",
    "lines": "1"
  }
}
```

Managing registered keywords

For each of your registered and active keyword, Geodrop requires the payment of a periodic fee. To avoid paying the fee, you have to either suspend or delete the keyword.

If you delete the keyword, of course, you will no longer pay the fee, but you lose its ownership and then someone else would register it[1].

WARNING! Deleting a keyword will remove the corresponding Inbox and all messages stored in it.

When a keyword is suspended, you are no longer able to receive SMS messages in the corresponding Inbox. You can still access the Inbox and read old messages and the keyword can be activated again within the next month[2], so you can start receiving SMS messages again. In the period in which a keyword is suspended, the fee is not due.

WARNING! If you don't re-activate the keyword within the maximum time allowed, Geodrop will delete it automatically an you will loose all messages stored in the corresponding Inbox. Before deleting the keyword, Geodrop will send you an alert e-mail and the keyword status will be set to "ALERTED".

Suspend and activate a keyword

To suspend or activate a keyword you must call the **keywords** method in http POST, passing the id of the keyword and the new status in a JSON string. Here is a template request to activate or suspend a keyword:

```
curl -X POST -H "Authorization: Bearer <access-token>" --data <request-body>
https://api.geodrop.net/in/1/sms/1/keywords
```

The "`<request-body>`" is a JSON string representing an object containing all the information needed to modify the keyword status. The JSON object has the following attributes:

- **id** (mandatory): the keyword id;
- **status** (mandatory): used to specify the keyword status. The accepted values are "ACTIVE" or "SUSPENDED".

A complete request body is shown below:

Raw

1 A deleted keyword can be registered again only after a quarantine period dependent on the type of contract in force
2 As for the quarantine, even this period may vary, depending on the current contract

```
{"id":1, "status":"active"}
```

Indented

```
{
  "id": 1,
  "status": "ACTIVE"
}
```

Finally, to activate the keyword with id 1, the complete request is the following:

```
curl -X POST -H "Authorization: Bearer <access-token>" --data '{"id": 1,"status":
"ACTIVE"}' https://api.geodrop.net/in/1/sms/1/keywords
```

If the keyword has been correctly activated, you will get the following response:

Raw

```
{"result": "0", "details": {"errors": "", "lines": "1", "id": 1}}
```

Indented

```
{
  "result": "0",
  "details": {
    "errors": "",
    "lines": "1",
    "id": 1
  }
}
```

Delete a keyword

To permanently delete a keyword and all received SMS messages stored in the corresponding Inbox, you must call the **keywords** method in http DELETE, passing the id as a querystring parameter as follow:

```
curl -X DELETE -v -H "Authorization: Bearer <access-token>"
https://api.geodrop.net/in/1/sms/1/keywords?id=3
```

In case of success, the response is like that shown above:

Raw

```
{"result": "0", "details": {"errors": "", "lines": "1"}}
```

Indented

```
{
  "result": "0",
  "details": {
    "errors": "",
    "lines": "1"
  }
}
```

Advanced features: using rules

In this section we describe in detail the different types of rules provided by the API of SMS In module. After a first part which describes the issues common to all types of rules, we punctually analyze the peculiarities of each action available, describing:

- the set of informations needed to set up the specific rule;
- how the rule uses the informations above when executing the action.

Adding new rules

To add a new rule you must call the **filters** method in http PUT. The main parameters accepted are:

- **action** (mandatory): identifies the operations to be performed once a SMS message is stored in the Inbox corresponding to the keyword for which the rule is defined. Is one of the following values:
  - **RESPONDER**: Geodrop sends a reply to the mobile phone number from which the SMS message is sent. In the **extra** object can be specified the text of the message and a personalized sender;
  - **FORWARD**: Geodrop sends a message to a specified number for each incoming message to which the rule can be applied;
  - **RELAY**: Geodrop calls the URL specified in the extra object for each incoming message to which the rule can be applied
  - **GROUP_ADD**: Geodrop adds the mobile number from which the SMS message is sent to the group of contacts specified in the **extra** object;
  - **GROUP_REMOVE**: Geodrop removes the mobile number from which the SMS message is sent from the group of contacts specified in the **extra** object;
- **number_id** (mandatory): the unique identifier of the **keyword** for which the rule is defined. Geodrop will try to apply the rule only on messages stored in the Inbox corresponding thi this keyword;
- **extra** (mandatory): is an object containing a set of informations that may vary according to the value of the **action** parameter value and are useful to the operations that the action involves. This parameter will be explained in detail in the next paragraphs;
- **description** (optional): brief description of the purpose for which the rule is designed;
- **condition** (optional): defines the checks to be performed to determine whether to apply the rule or not;
- **status** (optional): used to specify the filter status. The accepted values are 1 (active) or 0 (inactive). If a rule's status is inactive, Geodrop will never apply it to incoming messages.

The general method call has the following schema:

```
curl -X PUT -v -H "Authorization: Bearer <access-token>" --data <request-body>
https://api.geodrop.net/in/1/sms/1/filters
```

Where the `<request-body>` is a json string that will be detailed later in this section for each kind of rule.

Conditions

This document is intended to be a quick start guide and conditions may be really complex, so here we describe only the simplest cases, based on equality checks on the words contained in the message and on the sender's mobile phone number.

In the following examples, condition will be equalities of the form **<element>** = **<value>** (read: “the element equals the specified value”), where:

- **<element>** is a placeholder that identifies a specific word within the message text, the whole text or the sender's mobile phone number. In detail:
  - a placeholder that identifies a word is a number rounded by curly brackets: **{<num>}**. The number indicates the position of the word in the message. For example, the first word after the keyword is referred by the placeholder **{1}**;
  - the placeholder that indicates the whole message text, without the initial keyword, is **{text}**;
  - the placeholder that indicates the sender's mobile phone number is **{sender}**.
- **<value>** is a string or a number.

To give some examples of conditions, let us consider the following messages received by Geodrop:

- **message #1:**
  - **from**: +393333333333
  - **text**: testkey subscribe message
- **message #2:**
  - **from**: +394444444444
  - **text**: testkey first message

Def. 1: *A condition is **verified** for a message if, replacing the placeholders inside it with the corresponding elements of the message, the resulting equality is true.*

Def. 2: *An empty condition is always verified for any message.*

According to the above definitions, we can establish which of the following conditions are verified for each message.

Condition	Message #1	Message #2
{1} = subscribe	“subscribe = subscribe” is true, so the condition is **VERIFIED**	“first = subscribe” is false, so the condition is **NOT VERIFIED**
{SENDER} = +394444444444	+393333333333 = +394444444444 is false, so the condition is **NOT VERIFIED**	+ 394444444444 = +394444444444 is true, so the condition is **VERIFIED**
{2} = message	“message = message” is true, so the condition is **VERIFIED**	“message = message” is true, so the condition is **VERIFIED**

A given rule is applied only to messages for which its condition is verified. A rule with the first condition would be applied only to the message #1. A rule with the third condition would be applied to both messages.

Automatic responder

Geodrop can send a response to the mobile number from which a SMS message comes. The value of the **action** parameter for this kind of rules is “RESPONDER” and the **extra** object contains the following informations:

- **texts**: this could be a string or an array of strings. Geodrop will send a SMS for each string specified in this field;
- **sender**: personalized sender used for each message. It could be an alphanumeric string or a mobile phone number in E164 format;

A sample `<request-body>` to create a rule that acts as a simple responder is shown below:

Raw

```
{"action":"RESPONDER","number_id":1,"description":"Automatic responder","extra":
{"texts":["Your message has been received!","Thank you"],"sender":"MYRESPONDER"}}
```

Indented

```
{
  "action": "RESPONDER",
  "number_id": 1,
  "description": "Automatic responder",
  "extra": {
    "texts": [
      "Your message has been received!",
      "Thank you"
    ],
    "sender": "MYRESPONDER"
  }
}
```

In this example the optional parameters **condition** and **status** are missing, so:

1. the rule is created in status inactive (the default);
2. once activated, the rule will be applied to each received message;

3. if you send a SMS to the number on which the keyword with id 1 (corresponding to “testkey” in the previous examples), you will receive two SMS from “MYRESPONDER” with the following texts:
    - “Your message has been received!”;
    - “Thank you”.

Message forward

Received SMS messages can be forwarded to one or more mobile numbers. The value of the **action** parameter for this kind of rules is “FORWARD” and the **extra** object contains the following informations:

- **texts**: this could be a string or an array of strings, like for the automatic responder;
- **numbers**: this could be a mobile phone number or an array of mobile phone numbers. Geodrop will send a SMS for any string specified in the field text to each one of this mobile phone numbers;
- **sender**: personalized sender used for each message. It could be an alphanumeric string or a mobile phone number in E164 format;

A sample `<request-body>` to create a rule that forwards messages to a set of mobile phone numbers is shown below:

Raw

```
{"action":"FORWARD","number_id":1,"description":"Notice of receipt","status":1,"extra":
{"texts":"You got a new message!","numbers":
["+393341117125","+393666266294"],"sender":"SMSINBOX"}}
```

Indented

```
{
  "action": "FORWARD",
  "number_id": 1,
  "description": "Notice of receipt",
  "status": 1,
  "extra": {
    "texts": "You got a new message!",
    "numbers": [
      "+393341117125",
      "+393666266294"
    ],
    "sender": "SMSINBOX"
  }
}
```

Message relay

Received SMS messages can be forwarded to one or more URLs. The value of the **action** parameter

for this kind of rules is “RESPONDER” and the **extra** object contains the following informations:

- **urls**: a single object or an array of objects, each one structured as follows:
  - **url** (mandatory): the URL to be contacted;
  - **username** (optional): username used for basic authentication;
  - **password** (optional): password used for basic authentication;
  - **method** (optional): HTTP method to use for calling the url. The default value is “GET”;
  - **content_type** (optional): the mime type of the body of the request (used with POST and PUT requests). The default value is “application/x-www-form-urlencoded”;
  - **body** (optional): then body of the request.

A sample `<request-body>` to create a rule that calls a URL protected with basic authentication is shown below:

Raw

```
{"action":"RELAY","number_id":1,"description":"Notification on
website","status":1,"extra":{"urls":
{"url":"http://ilmiosito.com/notifications","username":"myuser","password":"mypwd"}}}
```

Indented

```
{
  "action": "RELAY",
  "number_id": 1,
  "description": "Notification on website",
  "status": 1,
  "extra": {
    "urls": {
      "url": "http://ilmiosito.com/notifications",
      "username": "myuser",
      "password": "mypwd"
    }
  }
}
```

Interaction with the contacts book

Linked to your Geodrop account you may have a set of contact books in which you can store and organize the mobile phone numbers of your contacts. For more informations on how the contact books module works, please refer to [BASE_GUIDE].

Geodrop can automatically modify the content of those contact books, adding or removing the mobile phone numbers from which SMS messages are sent.

To add the sender of a SMS message to your contacts book, you must define a rule whose **action** has the value “GROUP_ADD”. The extra object for this kind of actions contains the following

informations:

- **groups**: a string or an array of strings, each corresponding to one of the groups in which want to place the number

The following `<request-body>` defines a rule to insert the sender into two groups, named “All contacts” and “Authorized contacts”:

Raw

```
{"action":"GROUP_ADD","condition":"{1} = subscribe","number_id":1,"description":"Add to my contacts","status":1,"extra":{"groups":["All contacts","Authorized contacts"]}}
```

Indented

```
{
  "action": "GROUP_ADD",
  "condition": "{1} = subscribe",
  "number_id": 1,
  "description": "Add to my contacts",
  "status": 1,
  "extra": {
    "groups": [
      "All contacts",
      "Authorized contacts"
    ]
  }
}
```

To remove the sender of a SMS message from your contacts book, you must define a rule whose **action** has the value “GROUP_REMOVE”. The extra object is identical to that used in the case of addition.

The following `<request-body>` defines a rule to remove the sender from the group named “Authorized contacts”:

Raw

```
{"action":"GROUP_REMOVE","condition":"{1} = unsubscribe","number_id":1,"description":"Remove from authorized contacts","status":1,"extra":{"groups":"Authorized contacts"}}
```

Indented

```
{
  "action": "GROUP_REMOVE",
  "condition": "{1} = unsubscribe",
  "number_id": 1,
  "description": "Remove from authorized contacts",
  "status": 1,
  "extra": {
    "groups": "Authorized contacts"
  }
}
```

Modifying existing rules

To modify an existing rule you must call the **filters** method in http PUT, like for a new insertion, adding the mandatory **id** parameter to the **<request-body>**. If the id parameter is missing, the request is regarded as an attempt to add a new rule. The following examples shows how to set the status of a rule to active (1):

Raw

```
{"id":4,"status":1}
```

Indented

```
{
  "id": 4,
  "status": 1
}
```

Deleting rules

To delete a rule you must call the **filters** method in http DELETE, passing the unique identifier of the rule to be removed as the value of the **id** query string parameter.

```
curl -X DELETE -H "Authorization: Bearer <access-token>"
https://api.geodrop.net/in/1/sms/1/filters?id=4
```

In case of success, the response is like the following:

Raw

```
{"result": "0", "details": {"errors": "", "lines": "1"}}
```

Indented

```
{
  "result": "0",
  "details": {
    "errors": "",
    "lines": "1"
  }
}
```

Appendix A: status code and charset

Status codes

Status	Code	H.S.C.[3]	Description
OK	0	200	Operation successfully completed
GENERIC_FAILURE	-1	500	An unexpected error has occurred
ALREADY_IN	-2		The requested operation has violated a uniqueness constraint and cannot be completed
CONSTRAINT_FAILURE	-3	406	Some constraints on the method's parameters are not met
DB_FAILURE	-4		Operation failed due to database problems
NO_EFFECT	-5	200	Operation successfully completed, but no changes are made to the system status
NOT_OWNED_BY_USER	-6	401	The requested operation tried to modify an object not owned by the current user
UNABLE_TO_PAY	-7		The current user has not enough balance to complete the operation
UNAUTHORIZED	-8	401	The current user is not authorized to call the method

Calling a non-existent resource, you will get back an HTTP status code 404.

Calling a non-implemented method of an existing resource, you will get back an HTTP status code 405.

3 HTTP Status Code

GSM 03.38 charset

3GPP TS 23.038 / GSM 03.38																
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	@	£	$	¥	è	é	ù	ì	ò	Ç	*LF*	Ø	ø	*CR*	Å	å
1x	Δ	_	Φ	Γ	Λ	Ω	Π	Ψ	Σ	Θ	Ξ	*ESC*	Æ	æ	ß	É
2x	*SP*	!	"	#	¤	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	¡	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	Ä	Ö	Ñ	Ü	§
6x	¿	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	ä	ö	ñ	ü	à
1B 0x											*FF*					
1B 1x					^							*ESC2*				
1B 2x									{	}						\
1B 3x													[	~	]	
1B 4x	|															
1B 5x																
1B 6x						€										
1B 7x																

Appendix B: JSON Schema Definitions

LANES endpoint

GET http method reponse

```
{
  "description": "Lanes GET method reponse",
  "type": "object",
  "properties": {
    "result": {
         "description": "Error code. If 0, there has been no errors",
         "type": "number",
         "required":true
       },
    "msisdns": {
         "description": "List of available numbers",
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "id": {"type": "number", "required":true},
          "lane": {"type": "number", "required":true},
          "shared": {"type": "number", "required":true},
          "active": {"type": "number", "required":true},
          "msisdn": {"type": "string", "required":true},
          "user_id": {"type": "number", "required":true},
        }
      }
    },
    "details": {
         "type": "object",
      "properties": {
        "id": {
          "type": "number"
        },
        "lines": {
          "type": "number",
           },
         "error": {
          "type": "string",
           }
      }
     }
}
```

```
}
```

KEYWORDS endpoint

PUT http method request body

```
{
  "description": "Keywords PUT method request body",
  "type": "object",
  "properties": {
    "sub": {
        "type": "string",
        "required":true
      },
    "status": {
        "type": "string",
      },
    "num": {
        "type": "string",
      }
  }
}
```

GET http method response

```
{
  "description": "Keywords GET method response",
  "type": "object",
  "properties": {
    "result": {
        "description": "Error code. If 0, there has been no errors",
        "type": "number",
        "required":true
      },
    "numbers": {
        "description": "List of registered keyword for the user",
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "id": {"type": "number", "required":true},
          "sub": {"type": "string", "required":true},
          "num": {"type": "string", "required":true},
          "status": {"type": "string", "required":true},
          "user_id": {"type": "number", "required":true},
          "last_pay": {"type": "string", "format":"date-time", "required":true},
          "activation_date": {"type": "string", "format":"date-time", "required":true},
```

```
          "last_activation_date": {"type": "string", "format":"date-time",
"required":true},
          "sd_date": {"type": "string", "format":"date-time"}
        }
      }
    },
    "details": {
        "type": "object",
      "properties": {
        "id": {
          "type": "number"
        },
        "lines": {
          "type": "number",
          },
         "error": {
          "type": "string",
          }
      }
      }
  }
}
```

POST http method request body

```
{
  "description": "Keywords POST method request body",
  "type": "object",
  "properties": {
    "id": {
        "type": "number",
        "required":true
      },
    "status": {
        "type": "string",
      }
  }
}
```

PUT, POST and DELETE http method response

```
{
  "description": "Keywords PUT, POST and DELETE methods response",
  "type": "object",
  "properties": {
    "result": {
        "description": "Error code. If 0, there has been no errors",
```

```
        "type": "number",
        "required":true
      },
    "details": {
        "type": "object",
      "properties": {
        "id": {
          "type": "number"
        },
        "lines": {
          "type": "number",
          },
         "error": {
          "type": "string",
          }
      }
      }
  }
}
```

Inbox endpoint

GET http method response

```
{
  "description": "Inbox GET method response",
  "type": "object",
  "properties": {
    "result": {
        "description": "Error code. If 0, there has been no errors",
        "type": "number",
        "required":true
      },
    "messages": {
        "description": "List of messages in the requested Inbox",
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "masg_id": {"type": "number", "required":true},
          "sender_num": {"type": "string", "required":true},
          "text": {"type": "string", "required":true},
          "number_id": {"type": "number", "required":true},
          "operator": {"type": "string", "required":true},
          "time": {"type": "string", "format":"date-time", "required":true}
        }
```

```
      }
    },
    "details": {
        "type": "object",
      "properties": {
        "id": {
          "type": "number"
        },
        "lines": {
          "type": "number",
          },
         "error": {
          "type": "string",
          }
      }
      }
  }
}
```

FILTERS endpoint

PUT http method request

```
{
  "description": "Filters PUT method request",
  "type": "object",
  "properties": {
    "action": {"type": "string", "required":true},
      "number_id": {"type": "number", "required":true},
      "description": {"type": "string"},
      "condition": {"type": "string"},
      "status": {"type": "number"},
    "extra": {
        "description": "Object describing extra properties for the specified action",
      "type": [
        {
              "description": "Object for the RESPONDER action",
              "type": "object",
          "properties": {
            "texts": {
                  "type": [
                    {"type": "string"},
                {"type": "array", "items": {"type": "string"}}
              ],
              "required":true
            },
```

```
      "sender": {"type": "string"}
    }
  },
  {
    "description": "Object for the FORWARD action",
    "type": "object",
    "properties": {
      "texts": {
        "type": [
          {"type": "string"},
          {"type": "array", "items": {"type": "string"}}
        ],
        "required":true
      },
      "numbers": {
        "type": [
          {"type": "string", "format":"phone"},
          {"type": "array", "items": {"type": "string", "format":"phone"}}
        ],
        "required":true
      },
      "sender": {"type": "string"}
    }
  },
  {
    "description": "Object for the RELAY action",
    "type": "object",
    "properties": {
      "urls": {
        "type": [
          {
            "type": "object",
            "properties": {
              "url": {"type": "string", "format":"url", "required":true},
              "username": {"type": "string"},
              "password": {"type": "string"},
              "method": {"type": "string"},
              "content_type": {"type": "string"},
              "body": {"type": "string"},
            }
          },
          {
            "type": "array",
            "items": {
            }
          }
```

```
            ],
            "required":true
          }
        }
      },
      {
        "description": "Object for the GROUP_ADD and GROUP_REMOVE action",
        "type": "object",
        "properties": {
          "groups": {
            "type": [
              {"type": "string"},
              {"type": "array", "items": {"type": "string"}}
            ],
            "required":true
          }
        }
      }
    ]
  }
 }
}
```

PUT, POST and DELETE http method response

```
{
  "description": "Filters PUT, POST and DELETE methods response",
  "type": "object",
  "properties": {
    "result": {
      "description": "Error code. If 0, there has been no errors",
      "type": "number",
      "required":true
    },
    "details": {
      "type": "object",
      "properties": {
        "id": {
          "type": "number"
        },
        "lines": {
          "type": "number",
          },
         "error": {
          "type": "string",
```

```
            }
        }
    }
  }
}
```

Related documents

Code	Description
[BASE_GUIDE]	SpringBase user guide