Addressbook Web Service

LEKAB Communication Systems AB

Version 5.1.161, 2024-06-10

Addressbook Web Service

Introduction

This Web Service accepts `HTTP GET` or `HTTP POST` requests to update address books in the messaging platform.

In the initial implementation, there are two endpoints: `/listaddressbooks` and `/listtags`. Not all address books visible to a user have proper names, and may instead be identified only as a certain user's or organizational unit's or company's address book. The endpoint for listing addressbooks is intended to allow presentation of the available addressbooks to a human user for selecting the corresponding numeric id of the addressbook. The numeric id is then used with the other endpoints to specify which addressbook to operate on. The endpoint for listing tags gives a comprehensive list of all tag types and tag values in the address book, with counts for the number of occurrences.

Each of the endpoints supports the same function with `GET` and `POST`, but in the `GET` case, parameters are given in the calling `URL` (after a `?` sign, separated by `&` signs) , while in the `POST` case the parameters are given in a json document in the `HTTP POST` request body. `UTF-8` encoding is assumed in all `HTTP` bodies.

Both the `GET` and the `POST` versions return responses in the `HTTP` response body as a json document.

The format of the input and output json documents and the input url parameters are described below.

Different authentication methods available for GET requests

The `POST` requests all use authentication by giving username and password in the corresponding fields in the `JSON` document which is sent in the (automatically `HTTPS = SSL/TLS` encoded) `HTTP` request body.

For `GET` requests we offer three different ways of supplying these credentials:

1. Username and password can be sent in the `U` and `P` url parameters
2. Username and password can be given in the HTTP headers, `X-Lekab-Userid` and `X-Lekab-Password`, respectively. The values have to be the `Base64` encoding of (a `UTF-8` byte array representation of) the username or password to allow non-US-ASCII characters. Here testuser will be encoded as `dGVzdHVzZXI=` and testpass as `dGVzdHBhc3M=`
3. Username and password can be given as Basic authentication, i.e, the header `Authorization` should have the value `Basic token`, where the token is the `Base64` encoding of (a `UTF-8` byte array representation of) `username:password`. Here `testuser:testpass` will be encoded as `dGVzdHVzZXI6dGVzdHBhc3M=` and the `Authorization` header will have the value `Basic dGVzdHVzZXI6dGVzdHBhc3M=`

Chapter 1. The `/listaddressbooks` endpoint

Not all address books, to which a user has reading rights, have proper names. They may instead be identified only as a certain user's or organizational unit's or company's address book. This endpoint is intended to allow the presentation of available address books to a human user, who can thereby find the numeric id of the requisite address book. The numeric id is then used with the other endpoints to specify which address book to operate on. If no numeric id is given in the other endpoints, the default address book of the user calling the web service is assumed.

1.1. GET request example e.g. from web browser

```
curl
https://secure.lekab.com/addressbook/api/listaddressbooks?U=doggykennel&P=testpass
```

1.2. POST request example, probably from an application

```
https://secure.lekab.com/addressbook/api/listaddressbooks
```

With the contents of the HTTP body:

```
{"username":"doggykennel","password":"testpass"}
```

1.2.1. Explanation of parameters

POST json key	**GET query param**	**json value (strings quoted)**	**query param value (strings without quotes)**	
username	U	string	string	username of the API account in the service
password	P	string	string	password of the API account in the service

1.2.2. HTTP response

A successful request will return 200 OK and a json document of the following format. If the user does not present proper login credentials or cannot read any address books, a 401 Unauthorized will be returned, with no json document.

```
{ "addressbooks" : [
     { "id" : 576978287400472577,
       "name" : "Kennel Company - Shared address book" },
     { "id" : 646802963557457921,
       "name" : "Doggy Kennel - Private address book" },
     { "id" : 646802966493470722,
       "name" : "Kennel Club Members" }]
}
```

while the alphanumeric recipient number was rejected.

1.2.3. Explanation of response

POST json key	json value (strings quoted)	
addressbooks	json list of json documents	list of address book items never empty because no accessible address books gives HTTP 401
id	json long integer	numeric id of this address book item
name	string	name of this address book item

1.2.4. Example Python 3 code for the `/listaddressbooks` endpoint

```
import json
import requests

bookreq =  {"username" : "doggykennel", "password": "testpass"}
bookreq_json = json.dumps(bookreq)
bookurl = 'https://secure.lekab.com/addressbook/api/listaddressbooks'
response = requests.post(bookurl, data=bookreq_json)
bookresp = response.json()
for a in bookresp["addressbooks"]:
    print(a["name"] + " has numeric id " + a["id"])
```

will output

```
Kennel Company - Shared address book has numeric id 576978287400472577
Doggy Kennel - Private address book has numeric id 646802963557457921
Kennel Club Members has numeric id 646802966493470722
```

Chapter 2. The /listtags endpoint

A user, team or a company, can have an address book in the messaging platform, with contacts that each has a phonenumber that can receive SMS messages. These contacts can be marked with `tags` consisting of a tag type and a tag value, here often written separated by a colon character. For instance, a contact can be marked with the tag `Base:STO` if the person is based in Stockholm. Here `Base` is the tag type and `STO` is the tag value. The same contact can have many tags, for instance `Group:Management` or `On call:Yes`. Which tags types and tag values are used is up to the user/company, but they should preferably consist of letters and numbers, and can especially not contain the characters `;`, `|` or `:`. The tags are used in several parts of the service to select addressees for messages using, so called, tag filters.

The `/listtags` endpoint is used to retrieve a comprehensive listing of all tags in a given address book, with counts for the number of occurrences.

2.1. GET request example e.g. from web browser

```
curl
https://secure.lekab.com/addressbook/api/listtags?U=doggykennel&P=testpass&A=646802966
493470722
```

2.2. POST request example, probably from an application

```
https://secure.lekab.com/addressbook/api/listtags
```

with the contents of the HTTP body:

```
{
  "username" : "doggykennel",
  "password" : "testpass",
  "addressbookid" : 646802966493470722
}
```

2.2.1. Explanation of parameters

POST json key	GET query param	json value (strings quoted)	query param value (strings without quotes)	
username	U	string	string	username of the API account in the service

POST json key	GET query param	json value (strings quoted)	query param value (strings without quotes)	
password	P	string	string	password of the API account in the service
addressbookid	A	json long integer	number	id of the addressbook where tags are counted (optional, defaults to user’s default address book)

2.2.2. HTTP response

A successful request will return 200 OK and a json document of the following format.

```
{ "tags" : [ { "type" : "Breed",
               "typecount" : 250,
               "values" : [ { "value" : "German shepherd", "count" : 150 },
                            { "value" : "Labrador retriever", "count" : 57 },
                            { "value" : "Mutt", "count" : 43 } ] },
             { "type" : "Dog show newsletter",
               "typecount" : 104,
               "values" : [ { "value" : "Yes", "count" : 104 } ] } ],
  "addressbookid" : 646802966493470722,
  "addressbookname" : "Kennel Club Members" }
```

2.2.3. Explanation of response

POST json key	json value (strings quoted)	
addressbookid	json long integer	the id of the address book searched
addressbookname	string	name of the address book searched
tags	json list of json documents	one type item per tag type
type	string	tag type in this type item
typecount	integer	number of tags with this tag type in the addressbook
values	json list of json documents	one value item per tag value
value	string	tag value in this value item
count	integer	number of tags with this tag value and tag type in the addressbook

2.2.4. Example Python 3 code for the `/listtags` endpoint

```
import json
import requests

tagsreq =  {"username" : "doggykennel", "password" : "testpass", "addressbookid" :
646802966493470722}
tagsreq_json = json.dumps(tagsreq)
tagsurl = 'https://secure.lekab.com/addressbook/api/listtags'
response = requests.post(tagsurl, data=tagsreq_json)
tagsresp = response.json()
print("Listing tags in " + tagsresp["addressbookname"])
for a in tagsresp["tags"]:
    print(a["type"] + " occurs " + str(a["typecount"]) + " times, with " +
str(len(a["values"])) + " different values")
```

will output

```
Listing tags in Kennel Club Members
Breed occurs 250 times, with 3 different values
Dog show newsletter occurs 104 times, with 1 different values
```