Addressbook Web Service

LEKAB Communication Systems AB

Version 7.0.36, 2026-11-02

Addressbook Web Service

Introduction . . . 1
Address book name and ID . . . 1
Access rights to address books . . . 1
Different authentication methods . . . 2
Disabling Basic Authentication to force API Key usage . . . 3
Metadata and contact properties . . . 3
1. Modifying contacts in the address book . . . 4
2. Typical use case for the address book API . . . 5
2.1. Opt-in to a group in Azure . . . 5
2.2. Opt-out from a group in Azure . . . 6
3. The `/listaddressbooks` endpoint . . . 7
3.1. GET request example e.g. from web browser . . . 7
3.2. POST request example, probably from an application . . . 7
3.2.1. Explanation of parameters . . . 7
3.2.2. HTTP response . . . 7
3.2.3. Explanation of response . . . 8
3.2.4. Example Python 3 code for the `/listaddressbooks` endpoint . . . 8
4. The `/listtags` endpoint . . . 9
4.1. GET request example e.g. from web browser . . . 9
4.2. POST request example, probably from an application . . . 9
4.2.1. Explanation of parameters . . . 10
4.2.2. HTTP response . . . 10
4.2.3. Explanation of response . . . 10
4.2.4. Example Python 3 code for the `/listtags` endpoint . . . 11
5. Finding existing contacts with filters . . . 12
5.1. The `/getcontact` endpoint . . . 12
5.2. The `/findcontactbyaddress` endpoint . . . 12
6. List all contacts created or modified after a specific time . . . 14
7. The `/addcontact` endpoint . . . 15
7.1. Explanation of parameters . . . 16
7.2. HTTP response . . . 16
8. The `/updatecontact` endpoint . . . 18
8.1. Explanation of parameters . . . 19
8.2. HTTP response . . . 19
9. The `/deletecontact` endpoint . . . 21
9.1. Explanation of parameters . . . 21
9.2. HTTP response . . . 21
10. The `/deleteallcontacts` endpoint . . . 23

Introduction

© 2006 - 2026 LEKAB Communication Systems AB. Version 7.0.36, 2026-11-02.

The messaging platform includes many features for keeping the address book up to date and in sync with external systems or directories. It is possible to import users and groups from Microsoft Azure Entra ID using the Graph API or import lists of contacts from .CSV or Excel files regularly from a shared folder via SFTP.

This API provides complete control of how the address book is kept up to date from an external system by providing a Restful API with full access to the contacts. The Web Service accepts `HTTP GET`, `HTTP POST`, `HTTP PUT` or `HTTP DELETE` requests when working with contacts and address books in the messaging platform. Standard status codes are returned in the response and in many cases a JSON document is returned with additional information. Deleting contacts use the DELETE verb and updating expects PUT.

Some 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.

The format of the input and output json documents and the input url parameters are described in this manual.

Address book name and ID

The endpoint `/listaddressbooks` will show which address books the API user has access to. The unique id of the address book is later used in other API calls to make sure that the correct address book is being modified. The name of an address book might be changed at any time by a user in the web portal. It is important to use the ID which is static and remains the same over time.

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.

Access rights to address books

The API user account can be granted rights to address books in the same way as users who login to the web portal.

- The API user always has full access to its private address book. No other user can access the content. If the API user should not be able to access the private address book at all, enable the role `Hide the users personal address book`.

- The Company address book is shared by all users belonging to the company. By default, users only have read access. Users with the Company Administrator role has modify rights to the company address book but it is not recommended to add this role to API accounts because it also grants other rights such as user administation. If the API user account needs modify right to the Company address book, enable the role called `Manage settings, address books, templates for the whole company`. If the API user should not be able to access the company address book at all, enable the role `Hide the company address book`.
- Team address books are shared by the members who belong to the team. It is possible to add the API user account to a team if the content should be managed using this address book web service. By default, the members only have read access to contacts and tags. Modify rights are assigned by the Company administrator on each team by editing the API user. It is also possible to enable the role `Manage settings, address books, templates for own teams` for the API user account. This will automatically grant read/write access to all team address books where the API user is a member.

When editing the API user account, the default address book can be selected under Settings. This setting will be used in API calls if the optional `addressbookid` is omitted in the request.

If the API user tries to add, delete och update contacts and does not have modify rights in the address book, error 401 Unauthorized is returned.

Different authentication methods

Requests can be authenticated using the username and password of a user account with the API role enabled. User administration is available in the web portal. As an alternative, one or more API Keys can be generated for the user account. Using API Keys is the preferred method because different keys can be generated (or revoked) for development, test and production usage. If the API-key is supplied in the request header, the username and password are not required.

The API Key can be sent as a value in HTTP header parameter 'x-api-key'.

```
curl --location 'secure.lekab.com/addressbook/api/listaddressbooks' \
--header 'x-api-key: sdflk38jsdm1jdFlYQoim3mdoi4JOjZtMTZQVmRMMZjclc'
```

Please note that username and password authentication remains primarily for compatibility with older applications. It is recommended to change to API Keys. See also section about disabling Basic Authentication to force API Key usage below.

The `POST` requests can authenticate 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. If the `x-api-key` is used in the header, the username and password are not needed in the JSON payload.

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=`.
4. Oauth authentication is available from a separate endpoint By supplying the username and password as Basic authentication to the `/token` endpoint, in an `HTTP POST` call of type `application/x-www-form-urlencoded` containing the body "grant_type=client_credentials", one obtains a token valid for 10 minutes. This token can then be supplied as a `Bearer` token in the HTTP request, so that the header `Authorization` shall have the value `Bearer + one space + token`. More information about this in the **AuthenticationAPI** documentation.

Disabling Basic Authentication to force API Key usage

The use of Basic authentication can be disabled by activating a role called `Disallow Basic API Authentication` on the API user account. Enabling this role will prevent the API user from authenticating using the username and password (HTTP headers as well as in the JSON body). Only API keys assigned to the user will be allowed, and the time-limited OAuth2 bearer tokens retrieved from the special authentication service. Please refer to the OAuth2 service manual on the API documentation page for additional information.

Metadata and contact properties

The endpoint `/listtags` can be used to present metadata used in the address book to users of the custom application you are building. These tags can later be used when sending messages by calling the SMS API (REST).

For example, the application you are building could retrieve the tags and let users choose for example a tag type called Group and values A, B and C. The user can easily send messages to all contacts that belong to group A when your application combines this Address Book API with a simple call to the /send endpoint described in the SMS API manual (see `toaddressbooktagfilterjson`). It is possible to combine multiple tags to offer powerful and flexible addressing.

Chapter 1. Modifying contacts in the address book

It is possible to List, Add, Delete and Update contacts in the address book.

When the external application is the master data source, it usually keeps a unique ID for each contact. This ID is used when saving a contact in the address book for the first time and later, to find, update or delete it.

A complete contact object with all properties should be provided by the calling application when using endpoints `/addcontact` and `/updatecontact`. It is not possible to update only specific parts of a contact, for example one of the addresses or one tag value. Applications should retrieve the whole contact, change whatever is needed and send it back to update. The complete contact object is returned in the JSON body when using `/getcontact` and `/findcontactbyaddress`. This makes it easy to modify the existing object before sending it back.

The process is simple:

- Get a contact using the unique id (or by iterating through a search result).
- Compare the content against your master data and change whatever is needed in the JSON structure.
- Call `/updatecontact` and pass the modified JSON in the body. The same external id is used to find and update the existing contact.

Chapter 2. Typical use case for the address book API

One scenario could be that the security policy prevents granting read access to users and groups in Azure which is required by the Graph API. That means the polling for changes in Entra ID initiated from the messaging platform cannot be used. Instead, the address book API can be used to push changes initiated from the other directory. The customer builds their own integration and updates the address book in the messaging platform without sharing any access tokens or privileges to their Azure tenant.

The call to Address Book API can be added in a custom authentication flow where users request access to the Azure Groups used to subscribe to messages from the messaging platform. The Office 365 Groups connector for Power Platform and Azure Logic Apps has the OnGroupMembershipChange event that could be used to trigger events that update the address book.

Any SQL database or directory source can be integrated using the API in a similar way.

2.1. Opt-in to a group in Azure

- A user request to join "Group A" in Azure and a flow is triggered.
- A manager grants access for the user. The unique Azure user object id and group name are known (if only group id is known, lookup the group name so it can be used as tag value later).
- The Address Book API is called:
- Check if a contact already exist with the Azure user object id using `/getcontact`.
- If the object id exists in the address book:
  - Update the tag type "Groups" and add the value "Group A".
  - Optionally, update the name and the mobile number in the SMS address, as well as the EMAIL address type.
  - Optionally, update other tag types such as City, Country, Street, etc. This could be useful when filtering and addressing.
  - Save the contact with a call to `/updatecontact`. It will overwrite the existing becaues it has the same contactid.
- Else, if the object id does not exist in the address book:
  - Create a new contact with a contact id equal to the Azure user object id.
  - Add the name and the mobile phone as SMS address (also add the EMAIL address type if needed).
  - Add "Group A" as the value to the "Groups" tag type.
  - Optionally, add other tag types such as City, Country, Street, etc. This could be useful when filtering and addressing.

    - Save the contact with a call to `/addcontact`.

Note: An alternative to /getcontact followed by either update or add could be to first try /updatecontact and provide the complete JSON body based on the external data source. If the response is 404 Not Found, do /addcontact. One advantage with first calling /getcontact is that your application would receive additional data that may have been updated manually by users logged in to the web portal (or modified by the contact using the opt-in/opt-out service). The application could then merge these changes with the information that has changed in the external data source.

2.2. Opt-out from a group in Azure

- Someone is removed from "Group A" and a flow is triggered. The unique Azure user object id and group name are known (if only group id is known, lookup the group name so it can be used as tag value later).
- The Address Book API is called:
- Check if a contact already exist with this object id using `/getcontact`.
- If the contact with the id exists it is returned as a JSON object in the response:
    - Locate and update the tag type "Groups" and remove the value "Group A".
    - If other values remain in the "Groups" tag, keep them.
        - Save the changes to the existing contact using `/updatecontact`. It will overwrite the existing becaues it has the same contactid.
    - if "Group A" was the only value in "Groups":
        - Delete the whole contact using `/deletecontact` (unless the address book is used also for other use cases).

Chapter 3. 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.

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

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

3.2. 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"}
```

3.2.1. 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

3.2.2. 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" }]
}
```

3.2.3. 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

3.2.4. 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 4. 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.

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

```
GET
https://secure.lekab.com/addressbook/api/listtags?U=doggykennel&P=testpass&A=646802
966493470722
```

4.2. POST request example, probably from an application

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

with the contents of the HTTP body (username and password for compatibility with older integrations):

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

with the contents of the HTTP body where the API key is used instead of username and password:

```
{
  "apikey" : "sdflk38jsdm1jdFlYQoim3mdoi4JOjZtMTZQVmRMMZjclc",
  "addressbookid" : 646802966493470722
}
```

Note that the API Key can also be added to the header as `x-api-key`.

4.2.1. 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
addressbookid	A	json long integer	number	id of the addressbook where tags are counted (optional, defaults to user's default address book)

4.2.2. 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" }
```

4.2.3. 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

POST json key	json value (strings quoted)	
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

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

```
import json
import requests

tagsreq =  {"username" : "doggykennel", "password" : "testpass", "addressbookid" :
6468029664934707 22}
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
```

Chapter 5. Finding existing contacts with filters

5.1. The `/getcontact` endpoint

Find a contact with a specific external id, for example Azure Object ID:

```
GET https://secure.lekab.com/addressbook/api/getcontact?contactid=f32ca8c1-63ed-
4a1e-ad2c-e88157c7e991
```

It is possible to also include the addressbookid as a parameter. This is important if the contact with the same external unique id might be added to multiple address books available to the API user account.

```
GET
https://secure.lekab.com/addressbook/api/getcontact?addressbookid=925406391282233345&contactid=f32ca8c1-63ed-4a1e-ad2c-e88157c7e991
```

If the contact is not found, Status 404 Not Found is returned with details in the response body:

```
{
    "result": "ERROR",
    "error": "Contact not found: (addressbookid=925406391282233345,
contactid=f32ca8c1-63ed-4a1e-ad2c-e88157c7e991)"
}
```

5.2. The `/findcontactbyaddress` endpoint

Find contacts with a specific mobile number in the SMS address (usually only one but there could be duplicates):

```
GET
https://secure.lekab.com/addressbook/api/findcontactbyaddress?addresstype=SMS&address=46701234567
```

Find all contacts that can receive SMS messages because they have that address type:

```
GET https://secure.lekab.com/addressbook/api/findcontactbyaddress?addresstype=SMS
```

Find all contacts that can receive VOICE calls with Text-to-speech conversion or using MP3 playback:

```
GET https://secure.lekab.com/addressbook/api/findcontactbyaddress?addresstype=VOICE
```

Find all contacts that can receive E-mails:

```
GET https://secure.lekab.com/addressbook/api/findcontactbyaddress?addresstype=EMAIL
```

Find all contacts that are configured with a custom iOS or Android app address for push notifications:

```
GET https://secure.lekab.com/addressbook/api/findcontactbyaddress?addresstype=APP
```

Find contacts with a specific address, regardless of type. For example, check if a phone number exist either as SMS or VOICE type

```
GET
https://secure.lekab.com/addressbook/api/findcontactbyaddress?address=46701234567
```

Chapter 6. List all contacts created or modified after a specific time

Sometimes it is useful to keep track of modifications since the last time a scheduled task was performed. Contacts in the address book have properties for created time and updated time. Note that a contact can be updated by a user in the web portal. This will change the updated timestamp. There is also a field with the information about who made the update.

The time format used in GET is YYYY-MM-DDTHH:MM:SS+0100 and must be URL-encoded. A time zone can be added, for example: YYYY-MM-DDTHH:MM:SS+0100

It is possible to specify the specific address book to search, by adding the id to the request. The `/listaddressbooks` endpoint is used to find the id.

Find all contacts created after a specific date/time. Search in all address books the api user has access to (no address book id appended to the uri path):

```
GET https://secure.lekab.com/addressbook/api/contacts/createdafter/2025-03-
30T00%3A00%3A00%2B0100/
```

Find all contacts created after 2025-03-30 in a specific address book. The addressbook id 1086241428642066431 is added at the end:

```
GET https://secure.lekab.com/addressbook/api/contacts/createdafter/2025-03-
30T00%3A00%3A00%2B0100/1086241428642066431
```

Find all contacts modified after 2025-03-30, in all address books the api user has access to:

```
GET https://secure.lekab.com/addressbook/api/contacts/modifiedafter/2025-03-
30T00%3A00%3A00%2B0100/
```

Find all contacts modified after 2025-03-30 in a specific address book. The addressbook id 1086241428642066431 is added at the end:

```
GET https://secure.lekab.com/addressbook/api/contacts/modifiedafter/2025-03-
30T00%3A00%3A00%2B0100/1086241428642066431
```

Chapter 7. The `/addcontact` endpoint

Adding a contact uses the POST verb. The contact object is sent in the body as a JSON document. The contact has some general properties such as Name and Company. The unique contactid is important when you plan to search for and update the contact at later time. The Addresses and Tags are arrays and can contain a list of values.

The address types can be SMS, EMAIL, VOICE or APP. When using SMS and VOICE, make sure that the phone number in the address field is a complete number including international country code. Do not add a plus sign.

For the SMS address only, we will remove some punctuation, like hyphen, point, parenthesis and plus "+" sign for your convenience, and if your company has the setting for validating mobile numbers, or the `validatemobile` query parameter is `TRUE`, we will validate the number against a frequently updated mobile number series table. But it is important that the resulting number sequence contains the country code, we only support international numbers. Please refer to the E.164 standard for more information about phone number formats.

To add a contact with a specific external reference in `contactid`, in this case a Azure Object ID:

```
POST https://secure.lekab.com/addressbook/api/addcontact
```

Or, if mobile number validation is needed for the SMS type address:

```
POST https://secure.lekab.com/addressbook/api/addcontact?validatemobile=TRUE
```

```
{
  "addressbookid": "9254063912822333345",
  "contactid": "f32ca8c1-63ed-4a1e-ad2c-e88157c7e991",
  "contactname": "Eva Nilsson",
  "companyname": "Test company",
  "addresses": [
    {
      "addresstype": "SMS",
      "address": "46701234567"
    },
    {
      "addresstype": "VOICE",
      "address": "46701234567"
    },
    {
      "addresstype": "EMAIL",
      "address": "eva.nilsson@test.company.xyz"
    }
  ],
  "tags": [
    {
```

```
      "tagtype": "Groups",
      "tag": "Group A"
    },
    {
      "tagtype": "City",
      "tag": "Stockholm"
    }
  ]
}
```

7.1. Explanation of parameters

JSON key	Type	Description
addressbookid	Integer	Numeric id of the address book where the contact should be added. The id has previously been retrieved using /listaddressbooks. The id is optional. The default address book of the API user account is used if the parameter is missing.
contactid	String	The unique id of the contact to update. This id usually comes from the corresponding entity in the external data source such as Azure, CRM or HR systems.
contactname	String	The fullname of the contact displayed in the address book.
companyname	String	The company name of the contact displayed in the address book. This field is optional.
addresses	Array	One or more addresses of type SMS, EMAIL or VOICE. Use SMS also for RCS messaging addresses.
tags	Array	Tags store metadata that can be used in recipient filters when sending messages. Tags are key value pairs where tagtype is the key and tag contais the value. The tags are optional and can be omitted in the JSON body. For more information about tags, please refer to the /listtags endpoint.
validatemobile	HTTP query parameter	Validate SMS address against table of valid mobile number series

The `validatemobile` parameter checks if the given phone number (which must include the country code) is listed as belonging to a valid mobile number series for the country in question. If not, an error will occur and the contact will not be saved. This functionality can also be turned on in the company settings for your company. Only for SMS type address (VOICE can be to a landline phone).

7.2. HTTP response

If the contact was successfully added, the status 200 OK is returned and the response contains the `addressbookid` and `contactid` in the JSON body:

```
{
    "addressbookid": "9254063912822333345",
    "contactid": "f32ca8c1-63ed-4a1e-ad2c-e88157c7e991"
}
```


If the user does not present proper login credentials using the x-api-key in the header or does not have write access to the address books, a 401 Unauthorized will be returned. json document.


```
{
    "result": "ERROR",
    "error": "Unauthorized"
}
```

Chapter 8. The `/updatecontact` endpoint

Updating a contact uses the PUT verb. The contact object is sent in the body as a JSON document. For information about the JSON body, please refer to `/addcontact`. The query parameter `validatemobile` works the same as for `/addcontact`.

To update a contact with a specific external reference in `contactid`, in this case a Azure Object ID:

```
PUT https://secure.lekab.com/addressbook/api/updatecontact
```

Or, if mobile number validation is needed for the SMS type address:

```
PUT https://secure.lekab.com/addressbook/api/updatecontact?validatemobile=TRUE
```

```
{
  "addressbookid": "925406391282233345",
  "contactid": "f32ca8c1-63ed-4a1e-ad2c-e88157c7e991",
  "contactname": "Eva Nilsson",
  "companyname": "Test company",
  "addresses": [
    {
      "addresstype": "SMS",
      "address": "46701234567"
    },
    {
      "addresstype": "VOICE",
      "address": "46701234567"
    },
    {
      "addresstype": "EMAIL",
      "address": "eva.nilsson@test.company.xyz"
    }
  ],
  "tags": [
    {
      "tagtype": "Groups",
      "tag": "Group A"
    },
    {
      "tagtype": "City",
      "tag": "Stockholm"
    }
  ]
}
```

8.1. Explanation of parameters

JSON key	Type	Description
addressbookid	Integer	Numeric id of the address book containing the contact to be updated. The id has previously been retrieved using /listaddressbooks. The id is optional. The default address book of the API user account is used if the parameter is missing.
contactid	String	The unique id of the contact to update. This id was set when the contact was created and refers to the corresponding entity in the external data source such as Azure, CRM or HR systems.
contactname	String	The fullname of the contact displayed in the address book.
companyname	String	The company name of the contact displayed in the address book. This field is optional.
addresses	Array	One or more addresses of type SMS, EMAIL or VOICE. Use SMS also for RCS messaging addresses.
tags	Array	Tags store metadata that can be used in recipient filters when sending messages. Tags are key value pairs where tagtype is the key and tag contais the value. Sending an empty array "tags": [] will remove all tags from the contact, if they exist. For more information about tags, please refer to the /listtags endpoint.
validatemobile	HTTP query parameter	Validate SMS address against table of valid mobile number series

The `validatemobile` parameter checks if the given phone number (which must include the country code) is listed as belonging to a valid mobile number series for the country in question. If not, an error will occur and the contact will not be updated. This functionality can also be turned on in the company settings for your company. Only for SMS type address (VOICE can be to a landline phone).

8.2. HTTP response

If the contact was successfully updated, the status 200 OK is returned and the response contains the `addressbookid` and `contactid` in the JSON body:

```
{
    "addressbookid": "925406391282233345",
    "contactid": "f32ca8c1-63ed-4a1e-ad2c-e88157c7e991"
}
```

If the contact is not found, status 404 Not Found is returned with details in the response body:

```
{
    "result": "ERROR",
    "error": "Contact not found: contactid=f32ca8c1-63ed-4a1e-ad2c-e88157c7e991"
}
```

If the user does not present proper login credentials using the x-api-key in the header or does not have write access to the address books, a 401 Unauthorized will be returned. json document.

```
{
    "result": "ERROR",
    "error": "Unauthorized"
}
```

Chapter 9. The `/deletecontact` endpoint

Deleting a contact uses the DELETE verb. The `contactid` and `addressbookid` are sent as query parameters.

To delete a contact with a specific external reference in `contactid` from a specific address book:

```
DELETE https://secure.lekab.com/addressbook/api/deletecontact?contactid=f32ca8c1-63ed-4a1e-ad2c-e88157c7e991&addressbookid=925406391282233345
```

Note: The address Book ID is optional but recommended. The default address book of the API user is used if the parameter is missing. To list all address books the api user has access to, please refer to the `/listaddressbooks` endpoint.

To delete a contact with a specific external ID from the API user's default address book:

```
DELETE https://secure.lekab.com/addressbook/api/deletecontact?contactid=f32ca8c1-63ed-4a1e-ad2c-e88157c7e991
```

9.1. Explanation of parameters

DELETE query param	Type	Description
addressbookid	Integer	Numeric id of the address book containing the contact to be deleted. The id has previously been retrieved using /listaddressbooks. The id is optional. The default address book of the API user account is used if the parameter is missing.
contactid	String	The unique id of the contact to delete. This id was set when the contact was created and refers to the corresponding entity in the external data source such as Azure, CRM or HR systems.

9.2. HTTP response

If the contact was successfully deleted, the status 200 OK is returned and the response contains the `addressbookid` and `contactid` in the JSON body:

```
{
    "addressbookid": "925406391282233345",
    "contactid": "f32ca8c1-63ed-4a1e-ad2c-e88157c7e991"
}
```

If the user does not present proper login credentials using the x-api-key in the header or does not have write access to the address books, a 401 Unauthorized will be returned. json document.

```
{
    "result": "ERROR",
    "error": "Unauthorized"
}
```

Chapter 10. The `/deleteallcontacts` endpoint

Warning – use with caution. This request will empty the address book by deleting all contacts. In many scenarios, the application using the API is keeping track of changes and updates only specific contacts based on their id or date/time last modified. However, in some cases the application might just want to empty the address book and add all contacts at once. For this purpose, the delete all endpoint was added.

Note that tag filters in the address book remain after deleting all contacts. For example, if users in the web portal have prepared addressing filters for contacts matching tag type City=Stockholm and Department=Sales, these filters remain and will become useful again when the new contacts are added. Usually the deleteallcontacts call is followed by calls to `/addcontact` within a few seconds and will not interfer with users who are logged in and sending messages. However, if the process is adding thousands of conects and takes some time, it should be scheduled at night-time. Updating individual contacts instead of "delete all - add all" always cause less disturbance if users are expecteds to be active 24/7.

To delete all contacts in the address book with id 925406391282233345:

```
DELETE
https://secure.lekab.com/addressbook/api/deleteallcontacts?addressbookid=9254063912
82233345
```

Note: The address Book ID is mandatory when deleting all contacts. To list all address books the api user has access to, please refer to the `/listaddressbooks` endpoint.