Messaging Workflow Dispatch API

LEKAB Communication Systems AB

Version 7.0.36, 2026-11-02

Messaging Workflow Dispatch API

Introduction . . . 1
Different authentication methods available for requests . . . 1
1. The `/listsavedjobs` endpoint . . . 3
1.1. GET request example e.g. from web browser . . . 3
1.2. POST request example, probably from an application . . . 3
1.2.1. Explanation of parameters . . . 3
1.2.2. HTTP response . . . 3
1.2.3. Explanation of response . . . 4
1.2.4. Example Python 3 code for the `/listsavedjobs` endpoint . . . 4
2. The `/startsavedjob` endpoint . . . 5
2.1. GET request example e.g. from web browser . . . 5
2.2. POST request example, probably from an application . . . 6
2.2.1. Explanation of parameters . . . 6
2.2.2. Parameter availability for different generic job types (`GMM` and `GMA`) . . . 8
2.2.3. HTTP response . . . 9
2.2.4. Explanation of response . . . 9
2.2.5. Example Python 3 code for the `/startsavedjob` endpoint . . . 9
2.2.6. Tag filter usage . . . 10
2.2.7. Handling of settings . . . 10
3. Job event callbacks . . . 12
3.1. The `POST` callback type for job events . . . 12
3.1.1. `POST` callback for job event `JOB_START` . . . 12
3.1.2. `POST` callback for job event `JOB_ASSIGN` . . . 12
3.1.3. `POST` callback for job event `JOB_FINISH` . . . 13

Introduction

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

This Web Service API accepts `HTTP GET` or `HTTP POST` requests to list and run saved workflow templates used for sending and monitoring messages and responses.

Smart messaging workflows communicates via SMS and other channels for tasks where responses need to be tracked and presented, e.g. emergency handling, extra work personnel assignments, and unplanned schedule change notifications. The workflows are generally started from dedicated web forms in the Web Portal, leading the user through the construction of a messaging workflow for a sendout job. Workflows can be saved from the starting page, to facilitate re-use of settings in situations where similar needs must be fulfilled correctly and rapidly.

To allow automated messaging under program control, this API allows listing and starting of saved job templates.

Currently, there are two endpoints: `/listsavedjobs` and `/startsavedjob`. The endpoint for listing saved jobs is intended to allow presentation of the available jobs to a human user for selecting the corresponding numeric id of the saved job. The numeric id is then used with the `/startsavedjob` endpoint to specify which saved job to start. The endpoint for starting a saved job starts a messaging job, similar to the saved one, but with a few parameters changed. In the current version of the API, the job name (as displayed e.g. on the dashboard) and the first message of the sendout can be changed. Some job types also allow changing recipients (see below),

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. `GET` parameters containing non-URL characters must be URL encoded based on the `UTF-8` encoding of the characters.

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 requests

Every request to the service must include authentication, i.e. username and password, or equivalent. For `POST` requests these can be given in in the corresponding fields in the `JSON` document which is sent in the (automatically `HTTPS = SSL/TLS` encrypted) `HTTP` body. Since `GET` requests cannot have a body, instead username and password can be sent in the `U` and `P` url parameters. Note that everything in the URL after the host name is also part of the encrypted request, so url parameters are as safe during transfer as parameters in the body.

We also offer three different alternative ways of supplying these username and password credentials available in both the `GET` and `POST` cases:

1. Username and password can be given as standard Basic authentication, in which 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=`
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. An API key generated in the Web Portal can be used as a query parameter `key`, as the value of the header `X-API-Key` or as the field `apikey` in the `POST` request body. The length of the key varies depending on the length of the username (which is contained within the key). `TUxVOmRHVnpkSFZ6WlhJOjlKUUczTXU2TVZVU1Exd3Y` is a possible example key for the username `testuser`. The key is independent of the account password. When an API key is used, username and password need not be additionally supplied.
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.

If any of the alternative methods of authentication are used, parameter values pertaining to other authentication methods should be omitted or set to the empty string "".

Chapter 1. The `/listsavedjobs` endpoint

This endpoint is intended to allow the presentation of available saved jobs to a human, who can thereby find the numeric id of the requisite saved job. The numeric id is then used with the other endpoints to specify which job to operate on. Each job is listed by its name, numeric id and type of the job, where the type generally corresponds to the original job starting web page, from which the job was saved. Only saved jobs available for starting by the calling user will be listed.

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

```
curl https://secure.lekab.com/dispatch/api/listsavedjobs?U=partyparty&P=testpass
```

1.2. POST request example, probably from an application

```
https://secure.lekab.com/dispatch/api/listsavedjobs
```

With the contents of the HTTP body:

```
{"username":"partyparty","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
apikey	key	string	string	API key of the API account in the service
jobtype	T	string	string	job type of saved jobs to be listed (optional, defaults to listing all accessible types)

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.

```
{
"savedJobs" : [
    { "id" : 325961493210222597,
      "name" : "Alarm",
      "type" : "QuickAssign" },
    { "id" : 352872112744058883,
      "name" : "Office Party",
      "type" : "MonitoredMessage" },
    { "id" : 352872253467898883,
      "name" : "Oslo Alert",
      "type" : "MonitoredMessage" } ]
}
```

while the alphanumeric recipient number was rejected.

1.2.3. Explanation of response

json key	json value (strings quoted)	
savedJobs	json list of json documents	list of saved job items (can be empty)
id	json long integer	numeric id of this saved job item
name	string	name of this saved job item
type	string	type of this saved job item

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

```
import json
import requests

jobsreq =  {"username" : "partyparty", "password": "testpass",
"jobtype":"MonitoredMessage"}
jobsreq_json = json.dumps(jobsreq)
jobsurl = 'https://secure.lekab.com/dispatch/api/listsavedjobs'
response = requests.post(jobsurl, data=jobsreq_json)
jobsresp = response.json()
for a in jobsresp["savedJobs"]:
    print(a["name"] + " (of type " + a["type"] + ") has numeric id " + a["id"])
```

will output

```
Office Party (of type MonitoredMessage) has numeric id 352872112744058883
Oslo Alert (of type MonitoredMessage) has numeric id 352872253467898853
```

Chapter 2. The `/startsavedjob` endpoint

Smart messaging workflows communicates via SMS and other channels for tasks where responses need to be tracked and presented, e.g. emergency handling, extra work personnel assignments, and unplanned schedule change notifications. The workflows are generally started from dedicated web forms in the Web Portal, leading the user through the construction of a messaging workflow for a sendout job. Workflows can be saved from the starting page, to facilitate re-use of settings in situations where similar needs must be fulfilled correctly and rapidly.

To allow automated sendouts under program control, the `/startsavedjob` endpoint offers an API for starting such saved jobs.

The job to start is identified by its id, which may be known beforehand or listed by means of the `/listsavedjobs` endpoint. A display name for reference in e.g. the dashboard, can be given to replace the name of the saved job.

Note that a job with many recipients may take quite some time to start, and the `/startsavedjob` does not return a response until the job has started or an error has occured.

The message text of the main sendout can be changed using a parameter.

When the job was saved, scheduled send-out time parameters were removed, because they had relevance only for that send-out, and the original scheduled time is likely in the past when a saved job is re-used. A parameter to this endpoint sets a new scheduled start time, which is the time in the future when the first messages in the job will be sent. Job expiry and warning time points are calculated from the scheduled start time, if given. Note that recipient address resolution happens at the time of the API call, and later changes to the address book are not taken into consideration at the time of message sending.

For some job types, is possible to replace or add to the recipient list in the saved job with a combination of parameters specifying tag filters or groups from the address book. A boolean parameter determines whether to keep the old recipients or replace them. Specifically, one custom tag filter can be specified as a string or json document, and in addition a list of names of saved tag filters and a list of names of recipient groups in the address book can be given. Currently this is only implemented for jobs of the generic type GenericMonitoredMessage, and other use will result in an error message and no job being started. The use of tags and tag filters is described in more detail below.

In future versions of this endpoint we plan parameters for overriding the message channel set by the user for this specific job type.

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

```
curl
https://secure.lekab.com/dispatch/api/startsavedjob?U=partyparty&P=testpass&J=35287
2112744058883&N=Xmas%20bash&S=Xmas2022,EngineeringConsultants
```

2.2. POST request example, probably from an application

```
https://secure.lekab.com/dispatch/api/startsavedjob
```

with the contents of the HTTP body:

```
{
  "username" : "partyparty",
  "password" : "testpass",
  "jobid" : 352872112744058883,
  "jobname" : "Xmas bash",
  "toaddressbooksavedtagfilters" : ["Xmas2022","EngineeringConsultants"]
}
```

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
password	P	string	string	password of the API account in the service
apikey	key	string	string	API key of the API account in the service
jobid	J	json long integer	number	id of the saved job to start
jobname	N	string	string	name to distinguish the started job (optional)
message	M8	string UTF-8	string URL-encoded UTF-8	Message to send (optional)
assigned message	AM8	string UTF-8	string URL-encoded UTF-8	Message sent to assigned recipients when assigned (optional, GMA only)
tonumbers	T	json list of string	comma separated strings	recipient phone number list (optional, GMM only)
toaddressbookcustomtagfilter	A	string containing tag filter	string URL-encoded UTF-8 containing tag filter	filter specifying combination of tags in the user's address book (optional)

POST json key	GET query param	json value (strings quoted)	query param value (strings without quotes)	
toaddressbooktagfilterjson	(n/a)	json list of filter parts (see below)	(n/a)	filter specifying combination of tags in the user's address book (optional)
toaddressbooksavedtagfilters	S	json list of saved tag filter names	comma separated strings each URL-encoded UTF-8 containing saved filter name	names of saved tag filters from the user's address book (optional, GMM only)
toaddressbookgroups	G	json list of group/distribution list names	comma separated strings each URL-encoded UTF-8 containing group/distribution list name	names of groups/distribution lists from the user's address book (optional, GMM only)
excludenumbers	XT	json list of string	comma separated strings	recipient phone number list, which should not receive messages (optional, GMM only)
excludecustomtagfilter	XA	string containing tag filter	string URL-encoded UTF-8 containing tag filter	filter specifying combination of tags in the user's address book, which should not receive messages (optional, GMM only)
excludetagfilterjson	(n/a)	json list of filter parts (see below)	(n/a)	filter specifying combination of tags in the user's address book, which should not receive messages (optional, GMM only)
tokeep	K	boolean (default false)	T, TRUE, Y or YES	add the to-parameters to the existing recipients, groups or tag filters in the saved job (default is to replace, GMM only)
responsegoal	R	json integer > 0	integer > 0	number of confirmed YES-responders needed to finish the callout (optional for GMM, for GMA only with new tag filter)
scheduledstart	AT	string containing time	string URL-encoded UTF-8 containing time	Job where first messages are sent at the given future time. ISO 8601 standard format like "2022-02-14T15:00:00Z" or "2022-02-14T15:00Z" or "2022-02-14T17:00:00+02:00 or "2022-02-14T13:00-02:00" (where a url encoded plus sign is %2B)

2.2.2. Parameter availability for different generic job types (GMM and GMA)

Currently, every saved job type that can be started is derived from one of the two generic types, GenericMonitoredMessage (GMM) and GenericMultiAssign (GMA). These have a different look in the web portal interface and different ways of selecting recipients.

GMM derived pages start jobs that expect a response (usually Yes or OK to acknowledge receipt of the message) from every recipient. The GMM jobs have a single web page for the sendout and recipients are specified by adding to the recipient table a mixture of any number of contacts, tag filters (saved or custom), groups/distribution lists and non-contact numbers.

The GMA jobs, on the other hand, are started from a multi-step wizard page which select recipients from how they are tagged in the address book, using tag filters with different values of a single tag type, for example specifying a tag type Role and prescribing a response goal of 1 where the Role is Surgeon, and another response goal of 2 where the Role is Registered Nurse. The example sendout goes to all Surgeons and RNs and finishes when 1 Surgeon and 2 RNs have responded with Yes. The settings page for GMA jobs further allows for setting a global tag filter, so that only those Surgeons and RNs who also have the tag value OnCall:Yes will get the sendout.

The difference in how recipients are selected will make different parameters to this API work in different ways, or not at all, with the different generic job types. If a disallowed parameter is specified when starting a job of a certain type, this will result in a 400 error and no messages being sent.

Changing job name, scheduled start time and main message text is possible for both GMM and GMA.

GMA allows the "assigned message", i.e. the saved message to be sent at the time of assignment to each recipient who is assigned to the job, to be changed by supplying the parameter assignedmessage to the API. The corresponding message in GMM is taken from the currently active job type settings and cannot be changed from the API, resulting in a 400 error with no messages being sent if a non-blank value is supplied.

GMM also allows all the to-parameters and exclude-parameters. The parameter tokeep determines whether the to-parameter recipients are added to those in the saved job, or if they are replaced (the default). The exclude-parameters exclude recipients from both saved and new recipient lists. While changing the GMM responsegoal is not currently possible from the web page, it works fine from this API.

GMA only selects from one tag type, and keeps different response goals for different values for this tag. It can also do escalation using another tag type and set of values, where recipients with different values receive their message at different times. There is a global tag filter from the settings which is added onto the original filter. If this API is called, it is possible to specify a toaddressbookcustomtagfilter or a toaddressbooktagfilterjson, which may be complex, like Role:Surgeon;OnCall:Yes. When this feature is used, a responsegoal must also be given, while a responsegoal cannot be used without a new filter. When a complex new filter is used, it is divided into a first filter used as the tag type and tag value in standard GMA, while the rest of the tag filter goes into the global tag filter, replacing the value from settings. All filters, escalations and response goals from the saved job are ignored in that case. Calling this API without specifying a new filter and response goal will use the filters, escalations and response goals

from the saved job, as well as the global tag filter from the settings. The other `to`-parameters, `tokeep` and all `exclude`-parameters are disallowed for the `GMA` generic job type.

2.2.3. HTTP response

A successful request will, after some waiting time, return 200 OK and a json document of the following format.

```
{
  "result": "OK",
  "runtimeid": 383228819873509376,
  "jobname": "Xmas bash",
  "errormessage": ""
}
```

In the case that a job start request is formally successful, but the job could not be started because of problems in the saved job, there will still be a 200 OK response, but with the result code in the JSON response different from "OK". `HTTP` results other than 200 OK, e.g. 400 Bad request or 401 Unauthorized, will result from formal errors in the call to the web service, like missing non-optional data or invalid credentials.

2.2.4. Explanation of response

json key	**json value (strings quoted)**	
result	string	"OK" if successful, "ERROR" in most other cases
runtimeid	json long integer	the id of the runtime instance of the job
jobname	string	name to distinguish the started job
errormessage	string	error message, or empty string if OK

2.2.5. Example Python 3 code for the `/startsavedjob` endpoint

```
import json
import requests

startreq =  {"username" : "partyparty", "password" : "testpass", "jobid" :
352872112744058883, "jobname" : "Xmas bash", "toaddressbooksavedtagfilters" :
["Xmas2022","EngineeringConsultants"]}
startreq_json = json.dumps(startreq)
starturl = 'https://secure.lekab.com/dispatch/api/startsavedjob'
response = requests.post(starturl, data=startreq_json)
startresp = response.json()
print("Started job " + startresp["jobname"] +" with runtime id " +
str(startresp["runtimeid"]))
```

will output

```
Started job Xmas bash with runtime id 383228819873509376
```

2.2.6. Tag filter usage

A user, or a company with many users, can have address books in the service, with contacts that each contain a phone number that can receive SMS messages and/or an email address and/or an address to the contact's address in a mobile app. These contacts can be marked with `tags` consisting of a tag type and a tag value. For instance, a contact can be marked with the tag `Base:STO` if the person is based in Stockholm. 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 `:`. A tag filter in this interface consists of a list of filter parts, which are connected by logical AND (a contact is selected only if it fulfills every part of the filter). A filter part consists of a tag type and a list of tag values. A contact fulfills the filter part criterion if it has a tag with the given tag type and one of the given tag values (logical OR). Filters can be given either as a string in GET or POST or as a json document in the POST input. Using the string notation, the example contact would fulfill a `Base:STO|OSL|CPH;On call:Yes` filter, which consists of two filter parts, one saying that the contact needs to have a `Base:STO` or a `Base:OSL` or a `Base:CPH` tag, and the other that the contact needs to have an `On call:Yes` tag. Note that filter parts are separated by a semicolon, tag type and tag values are separated by a colon and tag values are separated by vertical bars (UNIX pipe symbols).

Which address book to use for the tag filter selections will in most cases have been set when setting the user's settings for the job type in question (discussed in more detail below).

When a tag filter or group/distribution list is part of the saved job, or added as a to-parameter, the address book will be searched for the correct recipients at the time of starting the job from this API, so that any recent changes in the address book will affect the resulting recipient list. That means that, for instance, employees that left the company will not get the new message, provided that they have been cleared from the company address book. There is no way of accessing the exact recipients that would have been found at the point of time when the job was saved when address book searching of groups and tag filters is employed. In contrast, if the saved job explicitly lists individual recipients or phone numbers, the address book will not be searched for whether they are still present, and any employees that left the company will still get the message. This is a major advantage of only using tag filters and groups/distribution lists in saved jobs.

2.2.7. Handling of settings

The web form for starting a job of the type in question, which is also where the job was originally saved, has access to a settings page. There, certain aspects of the job are controlled. Note that the settings are not saved with the job, but instead the most up-to-date settings for the job type are used when starting a job using this API. This can lead to unexpected results if settings are often changed, but for a typical user of the Dispatch API, a standard job of the job type is saved and settings for that job type are input at one point, and thereafter the API is used

repeatedly with the same saved job and settings. In future versions, additional API parameters for overriding some of the current settings are planned.

Chapter 3. Job event callbacks

When a sendout job is started, either from this API or from a web portal job starting page, it is possible to receive callback to a user-controlled URL when events happen during the job life cycle: "job has started", "recipient xxx has accepted and been assigned", "job has finished", etc.

A job is started by the user account who logs in with their credentials for this API or the web portal, but the owner of the job is a special "system user" account which can be shared by many login accounts within a company. In every user account that can start jobs, it is specified which "system user" account will own any jobs started.

The callback url is not specified per individual job, but instead is common for all jobs owned by the same system user account. In the account settings page for the system user, there is a selector for callback type (currently NONE or POST) and an input field for the callback url. The `NONE` callback type, obviously, means that no callback is sent.

3.1. The `POST` callback type for job events

When a job event occurs, a HTTP POST request is sent to the specified url, with the HTTP body containing a JSON document describing the event. The receiving service at the specified url is expected to respond with a standard HTTP `200 OK` response. Any response body content is ignored.

All timestamps are standard UNIX milliseconds after the epoch (Jan 1 1970 00:00:00.000 UTC)

3.1.1. `POST` callback for job event `JOB_START`

This callback is sent when a starting account (`agent`) starts a job with an `id` owned by an account (with id `userid`) of a certain `jobtype`.

```
{
"jobid" : "777903613308588032",
"userid" : 113495,
"eventtype" : "JOB_START",
"timestamp" : 1644928276273,
"jobtype" : "GroundHandling",
"sendtaskid" : "",
"sendname" : "",
"contactid" : "",
"agent" : "md-adm"
}
```

3.1.2. `POST` callback for job event `JOB_ASSIGN`

This callback is sent when the system assigns a recipient with a name `sendname`, an address book contact id (`contactid`) and a `sendtaskid` (internal to the system, mainly for support) to a task within the job with an `id` owned by an account (with id `userid`) of a certain `jobtype`.

```
{
"jobid" : "777903613308588032",
"userid" : 113495,
"eventtype" : "JOB_ASSIGN",
"timestamp" : 1644928296110,
"jobtype" : "",
"sendtaskid" : "777903613518303232",
"sendname" : "Tomas Hansson - Lekab",
"contactid" : "519898635249405952",
"agent" : "SYSTEM"
}
```

3.1.3. `POST` callback for job event `JOB_FINISH`

```
{
"jobid" : "777903613308588032",
"userid" : 113495,
"eventtype" : "JOB_FINISH",
"timestamp" : 1644928299193,
"jobtype" : "",
"sendtaskid" : "",
"sendname" : "",
"contactid" : "",
"agent" : "SYSTEM"
}
```