RingCentral Engage Digital REST API

Version	Date	Comment
1.2.16	August 12,2019	Add attachment_ids to reply assistant versions creation and update
1.2.15	August 12, 2019	Add offset maximum value in pagination and add information about negative offset being treated as 0
1.2.14	July 30, 2019	Change region example “en” to “us” for Time Sheets API
1.2.13	July 22, 2019	Rewrite content search examples
1.2.12	July 18, 2019	Add count_capped key for pagination
1.2.11	July 18, 2019	Add use_cobrowsing permission for Roles
1.2.10	July 16, 2019	Add source_ids filter for GET /1.0/users
1.2.9	July 09, 2019	URL encode “ to %22 for puma http parser

1.2.8	May 21, 2019	Add no_password field in user creation
1.2.7	Apr 24, 2019	Add first_categorization_at, first_content_id, first_content_author_id, last_content_id, intervention_user_ids, opened_intervention_user_ids fields for ContentThread
1.2.6	Apr 01, 2019	Add foreign_jwt_id and foreign_saml_id fields for Users
1.2.5	Feb 28, 2019	Add /1.0/tasks/:id/complete route
1.2.4	Feb 18, 2019	Add browser_notifications_disabled, display_only_unknown_bbcode, intervention_closing_period and use_two_letters_avatars parameters to Settings API
1.2.3	Feb 14, 2019	Add source filtering strategy to Webhooks
1.2.2	Feb 5, 2019	Add step field in task response and task filtering
1.2.1	Jan 10, 2019	Add Email/SMS custom parameters for content creation
1.2	Dec 17, 2018	Change throttling code from 503 to 429
1.1.14	Sept 25, 2018	Add the manage_users_of_my_teams permission
1.1.13	Sept 18, 2018	Update restriction params on Folders API
1.1.12	Sept 11, 2018	Add Attachments API
1.1.11	July 31, 2018	Add secret field to webhook API
1.1.10	July 27, 2018	Add thread_id to tasks API
1.1.9	July 27, 2018	Add foreign key and uuid filters on identities API
1.1.8	July 26, 2018	Add extra_values field on Identities API
1.1.7	July 11, 2018	Add display_name field on Identities API
1.1.6	July 4, 2018	Add expected status value for PUT /1.0/status/:agent_id
1.1.5	May 30, 2018	Add context_data for contents
1.1.4	May 3, 2018	Update timesheet format
1.1.3	April 24, 2018	Add activate topology endpoint
1.1.2	April 11, 2018	Update permissions list
1.1.1	March 19, 2018	Add delete user endpoint
1.1	February 5, 2018	Add API for Survey Responses

1.0.28	January, 9, 2017	Add email to user search params
1.0.27	December, 11, 2017	Add API for Folder
1.0.26	November, 6, 2017	Add custom status update to Status
1.0.25	October, 5, 2017	Add identity phone numbers
1.0.24	September, 26, 2017	Add user scoped endpoint to Status
1.0.23	September, 19, 2017	Add Task move endpoint
1.0.22	September, 6, 2017	Add in_reply_to_id and in_reply_to_author_id to contents
1.0.22	August, 10, 2017	Add position to Reply Assistant Groups
1.0.21	August, 9, 2017	Adding informations about the throttling
1.0.20	July, 10, 2017	Fixing typos in category and team documentation
1.0.19	June, 01, 2017	Added close and open endpoints to threads
1.0.18	May, 23, 2017	Added Reply Assistant endpoints
1.0.17	May, 18, 2017	Added informations to Threads
1.0.16	April, 25, 2017	Added/Removed informations to Sources
1.0.15	April, 19, 2017	Added informations to Sources
1.0.14	April, 10, 2017	Added nickname to Users
1.0.13	April, 6, 2017	Added PUT/POST/DELETE to Categories and Teams
1.0.12	March, 20, 2017	Added language to contents
1.0.11	March, 16, 2017	Added the “workbins” queue parameter for tasks
1.0.10	March, 6, 2017	Added API for topologies
1.0.9	March, 6, 2017	Added API for tasks
1.0.8	February, 16, 2017	Added approval_required, published and remotely_deleted to Content
1.0.7	January, 20, 2017	Added timezone to User
1.0.6	September, 8, 2016	Added sort extra parameter for Identity groups listing
1.0.5	September, 5, 2016	Added sort extra parameter for Identities and Interventions listing
1.0.4	September, 1, 2016	Added API for Presence Status

1.0.3	August, 10, 2016	Added custom_data to threads
1.0.2	July, 7, 2016	Added Domain endpoint (GET/PUT) + Locales and Timezones
1.0.1	July, 4, 2016	Added locale, spoken_languages and enabled fields to users api
1.0.0	June, 22, 2016	Added user_reply_in_average + bh to Intervention
0.9.8	may, 13, 2016	Added some fields to Intervention
0.9.7	may, 3, 2016	Added support of many identity_id filter on /1.0/interventions endpoint
0.9.6	april, 8, 2016	Added API for Channel
0.9.5	april, 6, 2016	Added API for TimeSheet
0.9.4	april, 5, 2016	Added API for Webhook
0.9.3	april, 4, 2016	Added custom_fields to Auth Identities (Dimelo Communities)
0.9.2	march, 15, 2016	Added GET and PUT for User sources_permissions
0.9.1	march, 8, 2016	Added POST and PUT for Role
0.9	march, 7, 2016	Added task view status API
0.8.4	november, 17, 2015	Added interventions extra attribute
0.8.3	july, 28, 2015	Added intervention/content/thread categorization and content/thread archival
0.8.2	june, 18, 2015	Add session.created/destroyed event
0.8.1	may, 27, 2015	Added info about attachments with viruses
0.8.0	november, 21, 2014	Added Intervention assign API
0.7.0	november, 13, 2014	Added contents attachments API
0.6.0	november, 12, 2014	Added content extra attributes for Facebook and twitter
0.5.1	may, 19, 2014	Added category_ids to interventions and added note to contents
0.5.0	may, 2, 2014	Add filtering query as HTTP params
0.4.4	april, 23, 2014	Added image_url field to content_sources and communities
0.4.3	april, 22, 2014	identity_group_id filter on /1.0/interventions endpoint.

0.4.2	april, 18, 2014	Getting all content sources is now paginated.
0.4.1	april, 18, 2014	Small change on errors format.
0.4.0	april, 16, 2014	Added /1.0/users/me API method and access token can now be given via Authorization request header.
0.3.3	april, 10, 2014	Updated response content type to application/json instead of application/x-javascript.
0.3.2	april, 10, 2014	Small refactoring on HTTP error codes.
0.3.1	march, 24, 2014	Added API for tags.
0.3.0	march, 21, 2014	Generic error responses. Added pagination everywhere.
0.2.0	march, 19, 2014	Add API call for events.
0.1.18	january, 24, 2014	Added identity & identity_group keywords on API.
0.1.17	january, 22, 2014	Added associated type on custom fields and custom fields on interventions.
0.1.16	january, 8, 2014	Added gender on identities and identities group APIs.
0.1.15	december, 19, 2013	Added API for custom fields and custom fields for identity groups API.
0.1.14	december, 13, 2013	Added closed_at to interventions API.
0.1.13	november, 26, 2013	Improved error messages.
0.1.12	november, 7, 2013	Added external_id to users API (on creation and update too).
0.1.11	august, 13, 2013	Added community_url to identities API.
0.1.10	august, 13, 2013	Added body_input_format in contents API.
0.1.9	august, 13, 2013	Added source_url in contents API.
0.1.8	may, 23, 2013	Add thread_category_ids in content threads API.
0.1.7	february, 26, 2013	Only intervention comments in token's user sources are now returned.
0.1.6	february, 26, 2013	Only identities in token's user sources are now returned.
0.1.5	february, 25, 2013	Only interventions in token's user sources are now returned.
0.1.4	february, 22, 2013	Only contents and threads in token's user sources are now returned.

0.1.3	february, 22, 2013	Open intervention is now mandatory to create a content with in_reply_to.
0.1.2	february, 20, 2013	Editing.
0.1.1	february, 19, 2013	Content source status.
0.1.0	february, 19, 2013	User and access token association. Create content API.
0.0.16	january, 23, 2013	Renamed disabled by enabled in users.
0.0.15	january, 11, 2013	Renamed Identity field: klout to klout_score.
0.0.14	december, 21, 2012	Added API to list on identity groups and filter them. Added note on identity_group creation.
0.0.13	december, 13, 2012	Added identity_group_id on identities and also added identity groups API.
0.0.12	november, 23, 2012	Added double quotes to content thread search parameters.
0.0.11	november, 20, 2012	Updated identities API according to identity merge feature.
0.0.10	august, 14, 2012	Refactored /contents/:id API method query parameter keywords.
0.0.9	july, 19, 2012	Removed visible_content_sources on users API.
0.0.8	july, 13, 2012	Removed category groups API and updated categories API.
0.0.7	june, 29, 2012	Added tags and Identity PUT.
0.0.6	june, 29, 2012	Added some facebook & twitter identities attributes.
0.0.5	june, 22, 2012	Added users write API methods (update, create, invite).
0.0.4	june, 20, 2012	Added intervention creation and deletion API methods.
0.0.3	june, 19, 2012	Added intervention comments read/write API methods.
0.0.2	june, 12, 2012	Added intervention_id keyword on contents query.
0.0.1	may, 23, 2012	First document release.

Table of contents

Introduction

RingCentral Engage Digital provides a REST JSON API to retrieve, create data and manipulate data from third party applications.

This document describes what can be done, under which conditions and the expected input and output formats and communication mechanism.

Note that API can easily be tested from any web browser or command line terminal.

Building HTTP request

Scheme and hostname

Request **must** be done with https scheme. Hostname is determined from your application name. If your application name is example, then the API hostname will be: example.api.engagement.dimelo.com.

HTTP method

As it is specified in API methods list, HTTP method can be GET, POST, PUT or DELETE.

URL path

All API paths is prefixed by /1.0. This is the version of RingCentral Engage Digital Rest API.

Access token

You **must** provide an access token on each of your API calls. Note that this access token must stay private. Don't publish it in a public website code. The access token is **unencrypted**.

You need to contact your project manager to have your own access token. Each access token is associated to an existing agent, all contents, interventions you make will be published as associated agent.

Note that some API methods requires authorization. It depends from the token's user permissions. Authorization is described on all API methods below.

As parameter

Access token can be specified in request parameter named access_token.

As request header

In order to be compliant with OAuth 2.0 standards access token can also be specified with an Authorization request header where value respects following format:

Bearer access_token_value

Example

We suppose you have a RingCentral Engage Digital instance accessible from https://test.engagement.RingCentral Engage Digital.com and an access token with value abc42.

With access token as parameter

To get all interventions on the source accessible by the token's users, URL will looks like:

https://test.api.engagement.dimelo.com/1.0/interventions?access_token=abc42

With access token as request header

To get all interventions on the source accessible by the token's users, you'll need to build your request with Authorization request header with proper value. HTTP request will looks like:

GET /1.0/interventions HTTP/1.1
Host: test.api.engagement.dimelo.com
Authorization: Bearer abc42

Multiple parameters

Some API methods described below take extra parameters. Some of them are **multiple** (example: category_ids, tags_ids or some custom fields). You must add double brackets [], after the parameters name.

Examples:

- ?firstname=john&category_ids[]=4242&category_ids[]=2854
- tag_ids[]=1&tag_ids[]=2
- custom_field_values[multiple_custom_field_key][]=value1&custom_field_values[multiple_custom_field_key][]=value2&custom_field_values[multiple_custom_field_key][]=value3

Uploading a file

The Attachments API allows you to upload file in order to use it later on in another API call (e.g. to create a content).
In order to upload a file to our API you need to pass the **file** parameter as multipart form data.

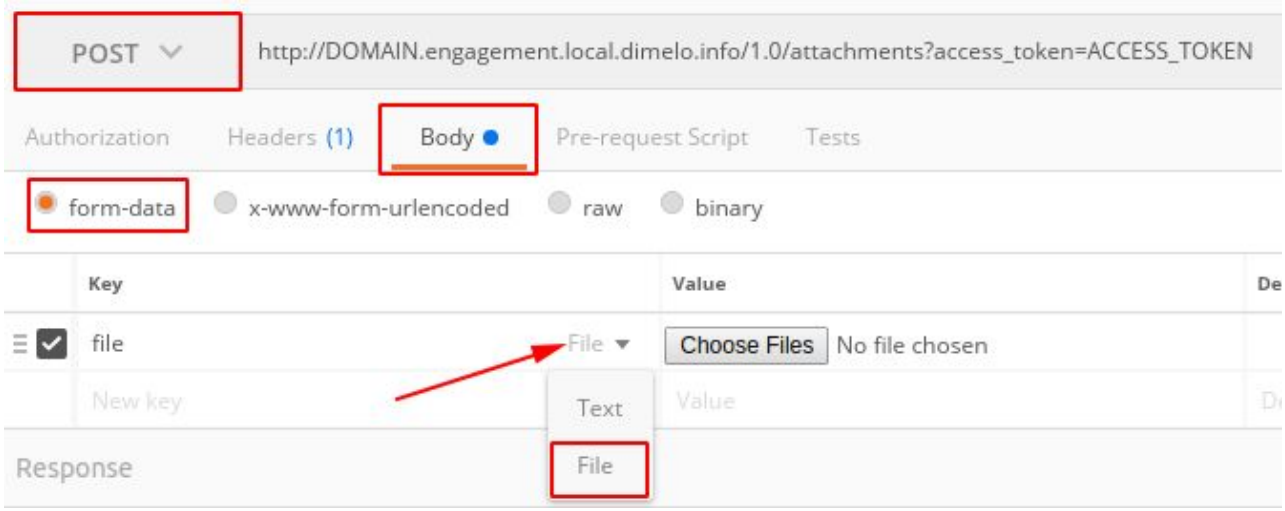
Using Curl

You can upload file via Curl by using the -F option with the path to your file, here's an example:

curl -X POST https://DOMAIN.api.engagement.local.dimelo.info/1.0/attachments?access_token=ACCESS_TOKEN -F 'file=@path/to/your/file'

Using Postman

Postman also allows you to upload files by doing adding a form-data parameter named **file**, and choose the file you want to upload:



Response

JSON

All responses are formatted in JSON (except some errors). Here is an example:

```
{
  "title": "Hello World",
  "created_at": "2012-05-21T01:19:49Z",
  "available": true,
  "comments_count": 4
}
```

Encoding

All responses are formatted using UTF-8 encoding.

Content type

The returned content-type is : application/json; charset=utf-8.

Errors

In case of a fatal error, a response is sent in JSON (application/json; charset=utf-8 content type) with an HTTP status code different than 200.

All errors rendered respects following format:

```
{
  error: "Error identifier",
  message: "A text message that describes the error",
  status: The HTTP status code
}
```

Here is some errors responses example that may occur:

Code	Response body example	Reason
400	{ "error": "invalid_access_token", "message": "Invalid access token", "status": 400 }	When you provide an invalid access token.
403	{ "error": "authorization_required", "message": "Authorization required", "status": 403 }	When the token's user is unable to access to given resource.
403	{ "error": "conflicting_rights", "message": "Cannot update an user that has more rights", "status": 403 }	When the token's user has less rights than the given user.
400	{ "error": "impersonation_error", "message": "Impersonation denied", "status": 400 }	When impersonation is forbidden for token's user.
400	{ "error": "impersonation_error" "message": "Impersonation denied on this user", "status": 400 }	When token's user can't impersonate given user.
404	{ "error": "not_found", "message": "No such content with id: \"421\"", "status": 404 }	When a record can't be found from its id.
400	{ "error": "impersonation_error", "message": "Invalid impersonated user id", "status": 400 }	When given impersonated_user_id parameter is invalid or there is not user with given id.
409	{ "error": "intervention_locked", "message": "<Resource> is already locked for <action>", "status": 409 }	When a similar action is already taking place on the resource
412	{ "error": "routing_error", "message": "Request must be in https", "status": 412 }	When a request is made in HTTP instead of HTTPs.

422	{
	"error": "user_has_content_or_intervention",
	"message": "Cannot delete user with an existing Content/Intervention",
	"status": 422
	}
		When the user you are trying to delete has at least a Content or an Intervention.

422	{
	"error": "validation_error",
	"message": "Identity can't be saved",
	"errors": [
	{
	"attribute": "firstname",
	"type": "too_short",
	"message": "Firstname is too short"
	},
	...
	],
	"status": 422
	}
		Record is invalid when you tried to create or update it. Only this kind of error is formatted in JSON.

429	{
	"error": "rate_limit_exceeded",
	"message": "Rate limit exceeded",
	"status": 429
	}
		When you reached API requests rate limit (500 in a minute).

Throttling

The number of queries is limited, the maximum is set to 500 queries per minute, otherwise you will hit the limit.

In case you reach the limit the server responds with 429 and the following JSON will be returned:

```
{
  "error": "rate_limit_exceeded",
  "message": "Rate limit exceeded",
  "'status'" => 429
}
```

User impersonation

All API requests accepts a "impersonated_user_id" parameter. This parameter allows you to overwrite the access token user by given user id.

Then all methods that use token's user (interventions creation, contents creation, etc.) will use given impersonated user as user.

Note that if token's user is unable to impersonate given user, an error will be rendered (400 HTTP code).

Example:

You have an access token with value 60576643bec4b6bd903232416ce5efad associated to user « John Doe ». If you create an new intervention comment, it will be created as « John Doe » author.

[POST]
https://[domain].api.engagement.dimelo.com/1.0/intervention_comments?body=test&intervention_id=c157a79031e1c40f85931829bc5fc552

Then, if you want to create intervention comment as « Bill Murray », with id d3b07384d113edec49eaa6238ad5ff00, you just have to add impersonated_user_id parameter:

[POST]
https://[domain].api.engagement.dimelo.com/1.0/intervention_comments?body=test&intervention_id=c157a79031e1c40f85931829bc5fc552&impersonated_user_id=d3b07384d113edec49eaa6238ad5ff00

Pagination

All API methods that returns a collection of records are paginated. Requests parameters and responses type always respect same format.

Request parameters

API methods that supports pagination takes two (optional) parameters:

- offset: The record index to start. Default value is 0. Negative values are treated as 0. Maximum value is 10 000 000.
- limit: The max number of records to return. Default value is 30, max value is 150.

JSON response

```
{
  "count": 4320,
  "offset": 90,
  "limit": 30,
  "records": [
    {},
    {},
    ...
  ],
  "count_capped": true
}
```

- count: This is equal to total records count (capped at 50000).
- count_capped: Boolean which is present when count limit has been reached (**optional**).
- offset: This is equal to given offset parameter.
- limit: This is equal to given limit parameter.
- records: The array of returned records.

Search & Filtering parameters

Most API methods that returns a collection of records can be filtered if they accept a q parameters. Be informed that using search in API is much more resource intensive and may be subject to rate limiting.

Request parameters

API methods that supports search (filtering) can be used in the following way:

- **With a `q` parameter containing a query string** (equivalent to RingCentral Engage Digital search interface)
  Example: fetch all threads where source id equal c157a or b12ec:
  /1.0/content_threads?q=source:c157a%20source:b12ec
- **With the search query passed asUn URL parameters**
  Fetch all threads where source id equal c157a or b12ec:
  /1.0/content_threads?source[]=c157a&source[]=b12ec

Please refer to RingCentral Engage Digital search API for all details about available query parameters

API methods

Communities

Format

JSON response of a community respects following format:

```
{
  "id": "37054c1c679c94f0069e33a9",
  "type": "facebook",
  "name": "Facebook",
  “image_url”: “http://test.engagement.dimelo.dev/facebook.png”,
  "active": true,
  "created_at": "2012-03-21T10:13:20Z",
  "updated_at": "2012-05-23T01:12:49Z"
}
```

Getting all communities

This method renders communities ordered by creation date (ascending).

HTTP method: GET
Path: /1.0/communities
Pagination: yes.
Authorization: only users that view communities.

Getting a community from its id

This method renders a community from given id.

HTTP method: GET
Path: /1.0/communities/:id
Pagination: no.
Authorization: only users that view communities.

Sources

Format

JSON response of a source respects following format:


```
{
  "id": "27054c1c679c94f4069e33a9",
  "type": "facebook",
  "community_id": "27054c1c679c95f0069e33a9",
  “image_url”: “http://test.engagement.dimelo.dev/facebook.png”,
  "name": "Facebook help",
  "active": true,
  "status": "ok",
  "hidden_from_stats": false,
  "color": 0,
  "sla_expired_strategy": false,
  "intervention_messages_boost": 9,
  "transferred_tasks_boost": null,
  "auto_detect_content_language": false,
  "default_content_language":  "fr",
  "tagging_category_ids": [],
  "user_tagging_category_ids": [],
  "content_archiving": false,
  "content_archiving_period": 36000,
  "time_sheet_ids": [],
  "created_at": "2012-04-21T01:14:12Z",
  "updated_at": "2012-05-23T01:18:49Z",
  "error_message": null
}
```


Source colors

Default:	0
Blue:	1
Green:	2
Turquoise:	3
Purple:	4
Yellow:	5
Orange:	6
Red:	7
Asphalt:	8
Grey:	9

Getting all sources

This method renders sources ordered by creation date (ascending).

HTTP method: GET
Path: /1.0/content_sources
Pagination: yes.
Authorization: only users that can view all sources or it renders only sources where user have read permission.

Getting a source from its id

This method renders a source from given id.

HTTP method: GET
Path: /1.0/content_sources/:id
Pagination: no.
Authorization: only users that can view all sources or it renders only sources where user have read permission.

Updating a source

This method updates an existing source from given attributes and renders it in case of success.

HTTP method: PUT
Path: /1.0/content_sources/:id
Pagination: no.
Extra parameters:

- name: Source name
- active: Activate/deactivate the source (Boolean)
- channel_id: Channel
- color: Color of the icon (see Source colors) (Integer)
- sla_response: Response time (seconds)
- sla_expired_strategy: SLA expired strategy ("max", "half" or "base")
- intervention_messages_boost: Priority boost of messages with intervention (Integer)
- transferred_tasks_boost: Priority boost of transferred tasks (Integer)
- hidden_from_stats: Hide from statistics (Boolean)
- default_category_ids: Default categories
- user_thread_default_category_ids: Default categories (agent messages)
- default_content_language: Default content language
- auto_detect_content_language: Auto-detect content language (Boolean)
- content_archiving: Automatic archiving of old contents (Boolean)
- content_archiving_period: Archive contents older than (seconds)

Authorization: only users that can update sources and only on sources where user have write permission.

Folders

Format

JSON response of a folder respects following format:

```
{
  "id": "9e8a01f258ee60930a2271cc",
  "created_at": "2011-05-04T22:00:00Z",
  "updated_at": "2011-05-04T22:00:00Z",
  "label": "Assistance",
  "parent_id": "08a5564a79b3d56562ff091a",
  "position": 0,
  "query": "source:d18c81948c137d86dac77216",
  "render_threads_count": true,
  "role_restriction": {},
  "team_restriction": {}
}
```

Getting all folders

This method renders folders.

HTTP method: GET
Path: /1.0/folders
Pagination: yes.
Authorization: only users that can manage folders.

Getting a folder from its id

This method renders a role from give id.

HTTP method: GET
Path: /1.0/folders/:id
Pagination: no.
Authorization: only users that can manage folders.

Creating a folder

This method creates a new folder. In case of success it renders the created folder, otherwise an error (422 HTTP code).

HTTP method: POST
Path: /1.0/folders
Pagination: no.
Authorization: only users that can manage folders.
Extra parameters:

- label: Folder's label (mandatory).
- parent_id: ID of the parent folder.

- position: position of the folder.
- query: query of the folder as described in Search API documentation.
    - Example: “active_and_assigned_to_me:true”
- render_threads_count: boolean describing display of the number of threads.
- role_restriction: list of roles allowed to see this folder. This parameter has to be a hash otherwise it will raise a 400 error. The key should be "only". For example : &role_restriction[only][]=4e5596cdae70f677b5000002
- team_restriction: list of teams allowed to see this folder. Same thing as role_restriction : team_restriction parameter has to be a hash with the key "only".

Updating a folder

This method updates an existing folder from given attributes and renders it in case of success.

HTTP method: PUT
Path: /1.0/folders/:id
Pagination: no.
Authorization: only users that can manage folders.
Extra parameters:

- label: Folder’s label.
- parent_id: ID of the parent folder.
- position: position of the folder.
- query: query of the folder as described in Search API documentation.
    - Example: “active_and_assigned_to_me:true”
- render_threads_count: boolean describing display of the number of threads.
- role_restriction: list of roles’ ids allowed to see this folder. This parameter has to be a hash otherwise it will raise a 400 error. The key should be "only". For example : &role_restriction[only][]=4e5596cdae70f677b5000002
- team_restriction: list of teams’ ids allowed to see this folder. Same thing as role_restriction : team_restriction parameter has to be a hash with the key "only".

Deleting a folder

This method destroys an existing folder. It renders the folder itself. It renders a 404 if id is invalid.

HTTP method: DELETE
Path: /1.0/folders/:id
Pagination: no.
Authorization: only users that can manage folders.

Roles

Format

JSON response of a role respects following format:

```
{
  "id": "12dfa96fec20593566ab7569",
  "created_at": "2012-05-23T10:12:49Z",
  "updated_at": "2012-05-23T10:12:49Z",
  "read_event": false,
  "update_intervention": false,
  "read_stats": true,
  "read_user": true,
  ...
}
```

Getting all roles

This method renders roles ordered by creation date (ascending).

HTTP method: GET
Path: /1.0/roles
Pagination: yes.
Authorization: only users that can manage roles.

Getting a role from its id

This method renders a role from given id.

HTTP method: GET
Path: /1.0/roles/:id
Pagination: no.
Authorization: only users that can manage roles.

Creating a role

This method creates a new role. In case of success it renders the created role, otherwise, it renders an error (422 HTTP code).

HTTP method: POST
Path: /1.0/roles
Pagination: no.
Extra parameters:

- label: Role's label (mandatory).

The list of available permissions is given in the next section

Authorization: only users that can manage roles.

Updating a role

This method updates an existing role from given attributes and renders it in case of success.

HTTP method: PUT
Path: /1.0/roles/:id
Pagination: no.
Extra parameters:

- label: Role's label.

List of permissions:

- access_help_center
- access_previous_messages
- access_pull_mode
- admin_stamp_answer
- approve_content
- assign_intervention
- author_block_content
- close_content_thread
- create_and_destroy_extension
- create_community
- create_content_source
- create_user
- delay_export_content
- delete_content_thread
- impersonate_user
- invite_user
- manage_api_access_tokens
- manage_app_sdk_applications
- manage_automatic_exports_tasks *
- manage_categories
- manage_chat
- manage_custom_fields
- manage_custom_notifications
- manage_emails_templates
- manage_folders
- manage_ice
- manage_identities
- manage_own_notifications
- manage_reply_assistant *
- manage_roles
- manage_rules_engine_rules *
- manage_surveys *
- manage_tags
- manage_teams
- manage_topologies
- manage_users_of_my_teams

- monitor_tasks
- monitor_team_tasks
- mute_content
- open_content_thread
- publish_content
- read_community
- read_content_source
- read_event
- read_export
- read_identity
- read_own_stats
- read_presence
- read_stats
- read_surveys *
- read_user
- receive_tasks
- reply_with_assistant *
- search_contents
- search_event
- update_community
- update_content_source
- update_extension
- update_identity
- update_intervention
- update_own_intervention
- update_settings
- update_time_sheet
- update_user
- use_cobrowsing
- use_emoji

* permission only available with the corresponding extension enabled

Authorization: A user can't update roles with more permissions than himself and can't give a role a permission he doesn't have.
Any permission updated with a user that does not have this permission will be ignored (The update is done, just not the unallowed permission)

Categories

Format

JSON response of a category respects following format:


```
{
  "id": "60944e5702bdafb74ec96142",
  "parent_id": "eb3c62690ec9025845fd3495",
  "name": "Technical",
  "created_at": "2012-05-23T01:12:49Z",
  "updated_at": "2012-05-23T01:12:49Z",
  "color": 0,
  "mandatory": false,
  "multiple": true,
  "post_qualification": false,
  "source_ids": [],
  "unselectable": false,
}
```


Category colors

Blue:	0
Green:	1
Turquoise:	2
Purple:	3
Orange:	4
Red:	5
Grey:	6

Getting all categories

This method renders categories ordered by creation date (ascending).

HTTP method: GET
Path: /1.0/categories
Pagination: yes.
Extra parameters:

- parent_id: To filter categories on given category parent id.

Authorization: no.

Getting a category from its id

This method renders a category from given id.

HTTP method: GET
Path: /1.0/categories/:id
Pagination: no.
Authorization: no.

Creating a category

This method creates a new team. In case of success it renders the created tag, otherwise, it renders an error (422 HTTP code).

HTTP method: POST
Path: /1.0/categories
Pagination: no.
Extra parameters:

- name: Team name.
- parent_id: ID of parent category.
- color: displayed color for the category, see Category colors.
- mandatory: mandatory categorization (Boolean).
- multiple: allow to assign multiple child categories (Boolean).
- post_qualification: post qualification (Boolean).
- unselectable: root category is unselectable (Boolean).
- source_ids: List of source id.

Note: The fields *mandatory*, *multiple*, *post_qualification*, *source_ids* and *unselectable* are accounted for only if the Category has no parent.

Authorization: only users that can manage teams.

Updating a category

This method updates an existing team from given attributes and renders it in case of success.

HTTP method: PUT
Path: /1.0/categories/:id
Pagination: no.
Extra parameters:

- name: Team name.
- parent_id: ID of parent category.
- color: displayed color for the category, see Category colors.
- mandatory: mandatory categorization (Boolean).
- multiple: allow to assign multiple child categories (Boolean).
- post_qualification: post qualification (Boolean).
- unselectable: root category is unselectable (Boolean).
- source_ids: List of source id.

Note: The fields *mandatory*, *multiple*, *post_qualification*, *source_ids* and *unselectable* are accounted for only if the Category has no parent.

Authorization: only users that can manage teams

Deleting a category

This method destroys an existing team. It renders the team itself. It renders a 404 if id is invalid.

HTTP method: DELETE
Path: /1.0/categories/:id
Pagination: no.
Extra parameters:

- take_over_category_id: ID of a category to recategorize (optional).

Authorization: only users that can manage teams

Tags

Format

JSON response of a tag respects following format:

```
{
  "id": "4ff15e9da90ffb522c00008b",
  "name": "Client VIP",
  "created_at": "2012-05-05T10:12:42Z",
  "updated_at": "2012-05-08T09:10:19Z",
}
```

Getting all tags

This method renders tags ordered by name (ascending).

HTTP method: GET
Path: /1.0/tags
Pagination: yes.
Authorization: no.

Getting a tag from its id

This method renders a tag from given id.

HTTP method: GET
Path: /1.0/tags/:id
Pagination: no.
Authorization: no.

Creating a tag

This method creates a new tag. In case of success it renders the created tag, otherwise, it renders an error (422 HTTP code).

HTTP method: POST
Path: /1.0/tags
Pagination: no.
Extra parameters:

- name: Tag name (mandatory).

Authorization: only users that can create a tag.

Updating a tag

This method updates an existing tag from given attributes and renders it in case of success.

HTTP method: PUT
Path: /1.0/tags/:id
Pagination: no.
Extra parameters:

- name: Tag's label.

Authorization: only users that are able to update tags.

Deleting a tag

This method destroys an existing tag. It renders tag itself. It renders a 404 if id is invalid.

HTTP method: DELETE
Path: /1.0/tags/:id
Pagination: no.
Authorization: only users that are able to destroy tags.

Teams

Format

JSON response of a team respects following format:

```
{
  "id": "7f57205ac51bd9ee4217cc33",
  "name": "Managers",
  "leader_ids": [
    "4d0fb475b242128032cbdf6d",
  ],
  "user_ids": [
    "4d0fb475b242128032cbdf6d",
    "60944e5702bdbfb74ec96141"
  ],
  "created_at": "2012-05-23T08:09:22Z",
  "updated_at": "2012-05-23T09:12:49Z"
}
```

Getting all teams

This method renders teams ordered by creation date (ascending).

HTTP method: GET
Path: /1.0/teams
Pagination: yes.
Authorization: no.

Getting a team from its id

This method renders a team from given id.

HTTP method: GET
Path: /1.0/teams/:id
Pagination: no.
Authorization: no.

Creating a team

This method creates a new team. In case of success it renders the created tag, otherwise, it renders an error (422 HTTP code).

HTTP method: POST
Path: /1.0/teams
Pagination: no.
Extra parameters:

- name: Team name.
- leader_ids: List of user id as leaders.
- user_ids: List of user id as team members.

Authorization: only users that can manage teams.

Updating a team

This method updates an existing team from given attributes and renders it in case of success.

HTTP method: PUT
Path: /1.0/teams/:id
Pagination: no.
Extra parameters:

- name: Team name.
- leader_ids: List of user id as leaders.
- user_ids: List of user id as team members.

Authorization: only users that can manage teams

Deleting a team

This method destroys an existing team. It renders the team itself. It renders a 404 if id is invalid.

HTTP method: DELETE
Path: /1.0/teams/:id
Pagination: no.
Extra parameters:

- take_over_category_id: ID of a category to recategorize (optional).

Authorization: only users that can manage teams

Users

Format

JSON response of a user respects following format:


```
{
 "id": "d033e22af348aeb5660fc214",
 "role_id": "d034e22ae348aeb5660fc214",
 "firstname": "John",
 "lastname": "Doe",
 "locale": "fr",
 "nickname": "Johnny",
 "email": "john@example.com",
 "external_id": "28102312",
 "gender": "man",
 "user_ids": [
  "60944e5702bdbfc74ec96141"
 ],
 "category_ids": [
  "4d0fb475b242128032cbdf6d",
  "60944e5702bdbfb74ec96141"
 ],
 "team_ids": [
  "7f57205ac51bc9ee4217cc32"
 ],
 "enabled": true,
 "created_at": "2012-05-05T10:12:42Z",
 "updated_at": "2012-05-08T09:10:19Z",
 "timezone": "Paris",
 "spoken_languages": [
  "fr",
  "En"
 ],
 "foreign_jwt_id": "",
 "foreign_saml_id": "",
 "invitation_pending": false
}
```


Note: the **foreign_jwt_id** and **foreign_saml_id** fields will only appear in the response if the related SSO extension is enabled on the domain.

Getting current access token user

This method renders user associated to given access token.

HTTP method: GET
Path: /1.0/users/me
Pagination: no.
Authorization: no.

Getting all users

This method renders users ordered by creation date (descending).

HTTP method: GET
Path: /1.0/users
Pagination: yes.
Extra parameters:

- email: To filter users on given email.
- category_id: To filter users on given category id.
- external_id: To filter users on given external id.
- foreign_jwt_id: To filter users on given foreign jwt id.
- foreign_saml_id: To filter users on given foreign saml id.
- identity_id: To filter users on given identity id.
- role_id: To filter users on given role id.
- source_ids: To filter users on their read permissions on the given sources (multiple).
- team_id: To filter users on given team id.

Authorization: only users that can view users. If the user affiliated to the token has the manage_users_of_my_teams permission, only the users belonging to at least one of the teams he's the leader of will be returned.

Getting a user from its id

This method renders a user from given id.

HTTP method: GET
Path: /1.0/users/:id
Pagination: no.
Authorization: If the user affiliated to the token has the manage_users_of_my_teams permission, only the users belonging to at least one of the teams he's the leader of will be returned.

Creating a user

This method creates a new user. In case of success it renders the created user, otherwise, it renders an error (422 HTTP code).

HTTP method: POST
Path: /1.0/users
Pagination: no.
Extra parameters:

- category_ids: User list of category ids (multiple).
- email: User email (mandatory).
- enabled: Whether the user is enabled or not (boolean).
- external_id: User external id, used for SSO.
- foreign_jwt_id: User JWT id, used by JWT SSO.
- foreign_saml_id: User SAML id, used by SAML SSO.
- firstname: User firstname (mandatory).
- gender: User gender ("man" or "woman").
- identity_ids: User list of identity ids (multiple).
- lastname: User lastname (mandatory).
- locale: Language for the user interface.
- nickname: User nickname.
- no_password: Used to create user without a password (only available if there is at least one SSO enabled on the domain)
- password: User plain password (mandatory unless no_password is present).
- role_id: User role id (mandatory).
- team_ids: User list of team ids (multiple).
- timezone: Use the timezone endpoint to get the timezone name (String), default is empty for domain timezone.
- spoken_languages: List of locales corresponding to the languages spoken by the user (multiple).

Authorization: only users that can create a user. If the user affiliated to the token has the manage_users_of_my_teams permission, the created user will need to belong to at least one of the teams he's the leader of. It will not be possible to assign the user to other teams.

Updating a user

This method updates users from given attributes and renders it in case of success.

HTTP method: PUT
Path: /1.0/users/:id
Pagination: no.

Authorization: only users that can update users. If the user affiliated to the token has the manage_users_of_my_teams permission, the updated user will need to belong to at least one of the teams he's the leader of. The teams the user affiliated to the token is the leader of will be the only ones which can be added or removed.

Inviting a user

This method invites a new user. In case of success it renders the created user, otherwise, it renders an error (422 HTTP code).

HTTP method: POST
Path: /1.0/users/invite
Pagination: no.
Extra parameters:

- category_ids: User list of category ids (multiple).
- email: User email (mandatory).
- enabled: Whether the user is enabled or not (boolean).
- external_id: User external id, used for SSO.
- foreign_jwt_id: User JWT id, used by JWT SSO.
- foreign_saml_id: User SAML id, used by SAML SSO.
- firstname: User firstname (mandatory).
- gender: User gender ("man" or "woman").
- identity_ids: User list of identity ids (multiple).
- lastname: User lastname (mandatory).
- locale: Language for the user interface.
- nickname: User nickname.
- role_id: User role id (mandatory).
- team_ids: User list of team ids (multiple).
- timezone: Use the timezone endpoint to get the timezone name (String), default is empty for domain timezone.
- spoken_languages: List of locales corresponding to the languages spoken by the user (multiple).

Authorization: only users that can invite other users. If the user affiliated to the token has the manage_users_of_my_teams permission, the invited user will need to belong to at least one of the teams he's the leader of. It will not be possible to assign the user to other teams.

Deleting a user

This method deletes the given user. In case of success it renders the deleted user, otherwise, it renders an error (422 HTTP code).

HTTP method: DELETE
Path: /1.0/users/:id
Pagination: no.
Authorization: only users that can update users. The user affiliated to the token must have at least all the permissions of the other user. If the user affiliated to the token has the manage_users_of_my_teams permission, the deleted user will need to belong to at least one of the teams he's the leader of.

Users - Sources_permissions

Format

JSON response of a user's sources permissions respects following format where the keys correspond to a source_id:


```
{
  "4f97fbea7aa58d073900344f": [
    "read",
    "reply",
    "initiate_discussion",
    "approval_required",
    "destroy",
    "reply_with_html"
  ],
  "54e1fa99776562f20a510300": [
    "read",
    "reply"
  ],
  "54b7e337776562be07840200": [
    "read",
    "reply",
    "initiate_discussion",
    "approval_required"
  ],
  "53bfe0760f4ca191fb000048": [
    "read",
    "reply",
    "initiate_discussion",
    "approval_required",
    "destroy"
  ],
  "53bfd78e7aa58d9d9e0000b6": [
    "read",
    "reply",
    "initiate_discussion",
    "approval_required",
    "destroy"
  ]
}
```


Get a user permissions by source

This method gets the permissions the user has on each source.

HTTP method: GET
Path: /1.0/users/:id/sources_permissions
Pagination: no.

Authorization: only users that can update users. The user affiliated to the token must have at least all the permissions of the other user. If the user affiliated to the token has the manage_users_of_my_teams permission, the updated user will need to belong to at least one of the teams he's the leader of.

Updating a user permissions by source

This method updates the permissions

HTTP method: PUT
Path: /1.0/users/:id/sources_permissions
Pagination: no.
Extra parameters: parameters are the source_id you want to update and value is a comma separated list of the different permissions you want to set on this source for this user.

<source_id_x>=<string of permission list>: To update the permission for source X identified by its id source_id_x with the permissions <string of permission_list>.

Possible permissions are : read, reply, approval_required, initiate_discussion, reply_with_html, destroy ...

In the example below:

- The permissions read, reply, initiate_discussion and destroy will be added for the source with id 7e0cb507bb45d106ccb4dd02.
- The permissions read, reply, approval_required will be added for the source with id d18c81948c137d86dac77216

Example: 7e0cb507bb45d106ccb4dd02=read,reply,initiate_discussion,destroy&
d18c81948c137d86dac77216=read,reply,approval_required

Authorization: only users that can update users. The user affiliated to the token must have at least all the permissions of the other user. If the user affiliated to the token has the manage_users_of_my_teams permission, the updated user will need to belong to at least one of the teams he's the leader of.

Identities

Format

JSON response of an identity respects following format:


```
{
  "id": "d033e22ae248aeb5660fc214",
  "community_id": "cea4e71ae44b2c1f110cf9bb",
  "community_url": "http://twitter.com/test",
  "type": "auth",
  "uuid": "john@example.com",
  "firstname": "John",
  "gender": "man",
  "identity_group_id": "bea2e71ae543b2cf140cf42a",
  "lastname": "Doe",
  "email": "john@example.com",
  “home_phone”: “0136656565”,
  “mobile_phone”: “0636656565”,
  "user_ids": [
    "d033e22ae348afb5660fc214",
    "315f166c5aca64a157f7d410"
  ],
  "avatar_url": "http://gravatar.com/avatar/205e460b479e2e5b48aec07710c08d50",
  "created_at": "2012-05-09T12:20:10Z",
  "updated_at": "2012-05-09T14:22:44Z",
  "display_name": "@JohnDoe",
  "extra_values": {
    "customer_id": "712cce0cae48b4e1ff9bad1b"
  }
}
```


Notes:

- The identity group id references the identity group that contains all informations (phone, notes, etc.). Many identities may belong to this group. If the identity group id is null, it means that identity does not have a group and any extra information. Please refer to identity groups API for more informations.

Extra attributes for **facebook** identities:


```
{
  ...
  "fb_bio": "My name is john",
  "fb_category": "Product/service",
  "fb_locale": "fr_FR",
  ...
}
```

Extra attributes for **twitter** identities:

```
{
  ...
  "klout_score": 28.1,
  "tw_description": "My name is doe, John Doe",
  "tw_followers_count" => 42,
  "tw_following_count" => 48,
  "tw_statuses_count" => 402,
  ...
}
```

Extra attributes for **auth** (Dimelo Communities) identities:

```
{
  ...
  "cutsom_field_1": "My First Field",
  "custom_field_2": "My Second Field",
  "custom_field_3": null,
  ...
}
```

Getting all identities

This method renders identities ordered by creation date (descending). Only identities in sources where token's user has "read" permission are returned.

HTTP method: GET
Path: /1.0/identities
Pagination: yes.
Extra parameters:

- community_id: To filter identities on given community id.
- identity_group_id: To filter on given group id.
- user_id: To filter identities on given user id.
- sort: To change the criteria chosen to sort the identities. The value can be "created_at" or "updated_at".
- foreign_id: To filter identities on given user id
- uuid: To filter identities on given uuid

Authorization: no.

Getting an identity from its id

This method renders an identity from given id. If token's user does not have "read" on identity's source community a 404 HTTP response will be returned.

HTTP method: GET
Path: /1.0/identities/:id
Pagination: no.

Authorization: no.

Identity groups

Format

JSON response of an identity group respects following format:


```
{
  "id": "50c93389421f4924360000aa",
  "created_at": "2012-12-13T01:46:49Z",
  "updated_at": "2012-12-13T01:46:49Z",
  "company": "Dimelo, a RingCentral Company",
  "custom_field_values" => {
    "customer_id" => 32409
  },
  "emails": [
    "john@example.com"
  ],
  "firstname": "John",
  "gender": "man",
  "home_phones": [
    "+33 3 23 96 97 98"
  ],
  "identity_ids": [
    "4f0aa52d656a3d75867f784b"
  ],
  "lastname": "Doe",
  "mobile_phones": [
    "+33 6 82 83 84 85",
    "+687 81 82 83"
  ],
  "notes": "Hello world",
  "tag_ids": [
    "cea4e71ae44b2c1f110cf9bb"
  ],
  "avatar_url": "http://gravatar.com/avatar/205e460b479e2e5b48aec07710c08d50"
}
```

Getting all identity groups

This method renders identity groups ordered by creation date (descending). Note that identity_group are created in a lazily only when data are manually added to an identity OR a two identity are merged altogether. That means that some identity DON'T have identity_group, and identity_group do not cover all identities.

HTTP method: GET
Path: /1.0/identity_groups
Pagination: yes.
Extra parameters:

- firstname: To filter groups on given firstname.
- lastname: To filter groups on given lastname.
- email: To filter groups that have given email.
- uuid: To filter groups that have given uuid.
- sort: To change the criteria chosen to sort the identities. The value can be "created_at" or "updated_at".

Authorization: no.

Getting an identity group from its id

This method renders an identity group from given id.

HTTP method: GET
Path: /1.0/identity_groups/:id
Pagination: no.
Authorization: no.

Updating an identity group

This method updates an identity group from given attributes and renders it in case of success.

HTTP method: PUT
Path: /1.0/identity_groups/:id
Pagination: no.
Extra parameters:

- company: Identity company.
- custom_field_values[custom_field_key]: Identity custom field with key « custom_field_key ». It can be multiple if custom field is multiple or is has multiple_choice type.
- emails: Identity emails (multiple).
- firstname: Identity firstname.
- gender: Identity's gender. It can be "man", "woman" or empty.
- home_phones: Identity home phones (mutiple).
- lastname: Identity lastname.
- mobile_phones: Identity mobile phones (multiple).
- notes: Identity notes.
- tag_ids: Identity tag ids (multiple).

Authorization: no.

Custom fields

Format

JSON response of a custom field respects following format:


```
{
  "id": "50c93389421f4924360000aa",
  "created_at": "2012-12-13T01:46:49Z",
  "updated_at": "2012-12-13T01:46:49Z",
  "associated_type_name": "IdentityGroup",
  "key": "customer_id",
  "label": "Customer ID",
  "multiple": false,
  "position": "2",
  "type": "integer"
}
```


Getting all custom fields

This method renders custom fields ordered by position (ascending).

HTTP method: GET
Path: /1.0/custom_fields
Pagination: yes.
Authorization: only users that can see custom fields in administration section.

Getting a custom field from its id

This method renders a custom field from given id.

HTTP method: GET
Path: /1.0/custom_fields/:id
Pagination: no.
Authorization: only users that can see custom fields in administration section.

Creating a custom field

This method creates a custom field. In case of success it renders the custom field, otherwise, it renders an error (422 HTTP code).

HTTP method: POST
Path: /1.0/custom_fields
Pagination: no.
Extra parameters:

- associated_type_name: The associated type of custom field. It can be IdentityGroup or Intervention.
- label: The label of the custom field (mandatory).
- key: The key of the custom field (example: customer_id). This is used to determine how it is stored on identity groups.
- type: The type of the custom field. It can be string, boolean, text, integer, float, single_choice, or multiple_choice (default: string).
- choices: A list of choices to be for single_choice, or multiple_choice types. This must be given as array.
- multiple: true or false, this as no effect on single_choice, multiple_choice or boolean types (default: false).
- position: an integer that indicates custom field's position between others (default: -1).

Authorization: only users that can create custom fields.

Updating a custom field

This method updates an existing custom field from given attributes and renders it in case of success.

HTTP method: PUT
Path: /1.0/custom_fields/:id
Pagination: no.
Extra parameters:

- label: Custom field's label.
- choices: Custom field's choices (multiple).
- position: Custom field's position.

Authorization: only users that are able to update custom fields.

Deleting a custom field

This method destroys an existing custom field. It renders custom field itself. It renders a 404 if id is invalid.

HTTP method: DELETE
Path: /1.0/custom_fields/:id
Pagination: no.
Authorization: only users that are able to destroy custom fields.

Threads

Format

JSON response of a thread respects following format:


```
{
  "id": "9c9903dc3d559a6801ec5441",
  "source_id": "d19c81948c137d86dac77216",
  "title": "ADSL modem issue",
  "interventions_count": 1,
  "contents_count": 4,
  "closed": false,
  "category_ids": [
    "4d0fb475b242228a32cbdf6d",
    "59248c4dae276a041cb296d2"
  ],
  "thread_category_ids": [
    "4d0fb475b242228a32cbdf6d",
  ],
  "extra_data": {
    "custom_my_number": 123456,
    "trigger_id"=>"foo"
  },
  "foreign_id": "ab-2031",
  "created_at": "2012-05-18T14:24:44Z",
  "updated_at": "2012-05-21T18:10:12Z",
  "first_categorization_at": "2016-07-05 15:14:07 +0200",
  "first_content_id": "4d0fb475b242228a32cbdf6d",
  "First_content_author_id": "4d0fb475b242228a32cbdf6d",
  "last_content_id": "4d0fb475b242228a32cbdf6d",
  "intervention_user_ids": [
    "4d0fb475b242228a32cbdf6d",
    "59248c4dae276a041cb296d2"
  ],
  "opened_intervention_user_ids": [
    "4d0fb475b242228a32cbdf6d",
    "59248c4dae276a041cb296d2"
  ]
}
```


The extra_data field depends on the source type. For more informations, please refer to the exports documentation.

Getting all threads

This method renders threads ordered by last content date (descending). Only threads in sources where token's user has "read" permission are returned.

HTTP method: GET
Path: /1.0/content_threads
Pagination: yes.
Extra parameters:

- q: A search query to filter threads. Please refer to Search & filtering parameters for more details.

Authorization: no.

Getting a thread from its id

This method renders a thread from given id. If token's user does not have "read" on thread's source a 404 HTTP response will be returned.

HTTP method: GET
Path: /1.0/content_threads/:id
Pagination: no.
Authorization: no.

Categorizing a thread

This method updates the categories of a thread. If token's user does not have "read" on thread's source a 404 HTTP response will be returned.
If the thread is already being categorized, a 409 HTTP response will be returned.

HTTP method: PUT
Path: /1.0/content_threads/:id/update_categories
Pagination: no.
Authorization: no.
Extra parameters:

- thread_category_ids: An array containing the new categories to set on the thread.

Archiving a thread

Archives the contents of a thread. If token's user does not have "read" on thread's source a 404 HTTP response will be returned.
If the thread is already being archived, a 409 HTTP response will be returned.

HTTP method: PUT
Path: /1.0/content_threads/:id/ignore
Pagination: no.
Authorization: no.

Close a thread

Notice:

Thread closure/opening is only available for the following sources:

- Emails
- Answers
- Ideas
- Facebook Messenger
- Google+
- Lithium
- Mobile Messaging

Starts a job to close a thread. It returns the thread but as the job is asynchronous, the state of the “close” attribute in the returned object do not is the one when the job started.
If token’s user does not have “read” on thread’s source a 404 HTTP response will be returned.
Returns a 403 if the thread cannot be closed or if the user does not have the permission to close a thread.

HTTP method: PUT
Path: /1.0/content_threads/:id/close
Pagination: no.
Authorization: no.

Open a thread

Notice:

Thread closure/opening is only available for the following sources:

- Emails
- Answers
- Ideas
- Facebook Messenger
- Google+
- Lithium
- Mobile Messaging

Starts a job to open a thread. It returns the thread but as the job is asynchronous, the state of the “close” attribute in the returned object is the one when the job started.
If token’s user does not have “read” on thread’s source a 404 HTTP response will be returned.
Returns a 403 if the thread cannot be opened or if the user does not have the permission to open a thread.

HTTP method: GET
Path: /1.0/content_threads/:id/open

Pagination: no.
Authorization: no.

Contents

Format

JSON response of a content respects following format:


```
{
 "id": "73f1cb2938229d7fa222d096",
 "source_id": "d19c81948c137d86dac77216",
 "source_url": "http://domain-test.answers.dimelo.com/questions/42",
 "source_type": "answers",
 "thread_id": "26c56bc5b71c5193b6f8c656",
 "in_reply_to_id": "58bc74bc920026b30196fdf4",
 "in_reply_to_author_id": "57ea9a7f13047d506e65289d",
 "author_id": "4f0aa52d656a3d75867f784c",
 "creator_id": "ac24dc966bc7ecb74017c0cd",
 “foreign_id”: “7789”,
 "type": "question",
 "created_from": "synchronizer",
 "private_message": false,
 "status": "replied",
 "intervention_id": "7f946431b6eebffafae642cc",
 "language": "fr",
 "body": "Hello,\n\nHow to unlock my nokia 3210?\n\nThanks!",
 "body_formatted": {
  "text": "Hello,\n\nHow to unlock my nokia 3210?\n\nThanks!",
  "html": "<p>Hello,</p>\n\n<p>How to unlock my nokia 3210?</p>\n\n<p>Thanks!</p>"
 },
 "body_input_format": "text",
 "title": "Nokia 3210 unlocking",
 “attachments_count”: 1,
 “attachments”: [{
  "id":"5464b5c04d61639684110000",
  "created_at":"2011-05-05T22:00:00Z",
  "updated_at":"2011-05-05T22:00:00Z",
  “content_type”: “application/pdf”,
  “size”: 174784,
  "filename":"sso.pdf",
  "foreign_id":"123",
  "embed":"false",
  "public?":"true",
  "url":"http://domain-test.engagement.dimelo.dev/attachments/5464b5c04d61639684110000"
 }],
 "synchronization_status": "success",
 "category_ids": ["4d0fb475b242228033cbdf6d", "60944e5702bdafb74ac96141"],
 "created_at": "2012-05-24T04:00:44Z",
 "updated_at": "2012-05-24T04:00:44Z",
```

```
 "approval_required": false,
 "remotely_deleted": false,
 "published": true,
"context_data": {
  "customer_id": 1214
}
}
```


Notes:

- body_input_format values can be: text or html.
- body_formatted always contains text **and** html versions.
- foreign_id is the id of the content on its corresponding source.
- category_ids are content categories if none, they are default to intervention categories or thread categories
- Context_data is present only if the content has context_data associated. The context_data hash keys are the custom fields keys.
- You can download the attachments by using an API access token with the following URL : https://[your-domain].engagement.dimelo.com/attachments/[attachment_id]?access_token=[your_access_token]

Format (extra attributes)

Certain type of content : Facebook video, photos, tweets may return extra attributes

Attachments with viruses

In the eventuality that an attachment contains a virus, said attachment will be returned with an extra virus_signature, describing the virus found.
In addition, the url field will be for security reasons.

Facebook Album

Attribute Name	Type	Description
fb_album_type	String	Type of the album, can be : app, cover, profile, mobile, wall, normal or album.
fb_description	String	The description of the album
fb_link	String	A link to this album on facebook
fb_name	String	Title of the album

Facebook Photo

Attribute Name	Type	Description

fb_link	String	A link to the photo on facebook
fb_name	String	The user provided caption given to this photo

Facebook Video

Attribute Name	Type	Description
fb_description	String	The description of the video
fb_link	String	A link to the video on facebook
fb_name	String	The video title or caption
fb_source	String	An URL to the raw, playable video file.

Facebook Link

Attribute Name	Type	Description
fb_caption	String	Caption extracted by facebook from target page e.g: blog.b-and-you.fr
fb_description	String	A description of the link (appears beneath the link caption)
fb_link	String	The URL that was shared
fb_name	String	The name of the link
fb_source	String	A URL to the thumbnail image used in the link post

Twitter tweet

Attribute Name	Type	Description
tw_entities	Hash	Entities provide structured data from Tweets including resolved URLs, media, hashtags and mentions without having to parse the text to extract that information. https://dev.twitter.com/overview/api/entities-in-twitter-objects#tweets

Getting all contents

This method renders contents ordered by creation date (descending). Only contents in sources where token's user has "read" permission are returned.

HTTP method: GET
Path: /1.0/contents
Pagination: yes.
Extra parameters:

- q: To filter contents on given query. Query works exactly like threads query but only have those keywords: intervention,identity, identity_group, source, status_in, thread or text. Order can be created_at.desc (default) or created_at.asc.
  e.g. q=intervention:7f946431b6eebffafae642cc%20source:d19c81948c137d86dac77216
  Please refer to Search & filtering parameters for more details.

Authorization: no.

Getting a content from its id

This method renders a content from given id. If token's user does not have "read" on content's source a 404 HTTP response will be returned.

HTTP method: GET
Path: /1.0/contents/:id
Pagination: no.
Authorization: no.

Creating a content

This method allows you to create a new content. It can be a reply to another content or a content that initiate discussion. It uses the token's user as content user if he is authorized. Content will be created in RingCentral Engage Digital and pushed asynchronously to the source.

Replying to a customer content is usually possible (unless the source/conversation is read only).

Composing a content on the contrary depend on the source itself :
- The source may not support it (and be purely reactive like Instagram, Messenger ...)
- Some sources (usually public account) like Twitter or Facebook page allows to publish content without targeting specific individuals.
- Some sources (usually non public media) require specific targeting (phone number for SMS, email address for email, customer_id ...) to be able to create a content. This is source specific and detailed under the generic parameters.

HTTP method: POST
Path: /1.0/contents
Pagination: no.
Extra parameters:

- author_id: The identity id of content. This parameter is not mandatory, by default it uses the token's user first identity on source.
- body: The content's body. This parameter is mandatory.
- in_reply_to_id: The content's id you want to reply to. If omitted, a new discussion will be created. If source does not support to initiate discussion this parameter is mandatory.
- private: Created contents are public by default, set this parameter to "1" in order to create a private reply. It is NOT supported on every source.
- source_id: The source to create content to. If you specify in_reply_to_id parameter, source will be determined from it. Otherwise, this parameter is mandatory.
- attachment_ids: An array containing the attachments' ids that need to be attached to this content.

Authorization: only users that can reply or initiate discussion (= compose) on given source. it also renders an error if in_reply_to isn't synchronized or if in_reply_to content is not under an **open** intervention.

Source specific parameters :

Emails

When creating a content on an email source, some other parameters will be required/available.

Extra parameters:

- title: The subject of the email. This parameter is mandatory when initiating a discussion.
- to: An array containing the email addresses used in the "To" section of the email. This parameter is mandatory when initiating a discussion.
- cc: An array containing the email addresses used in the "Cc" section of the email.
- bcc: An array containing the email addresses used in the "Bcc" section of the email.

SMS

When creating a content on an SMS source, some other parameters will be required/available.

Extra parameters:

- to: The number the SMS will be sent to. It must start with 00 or +, example: +33634231224 or 0033634231224. This parameter is mandatory when initiating a discussion.

Categorizing a content

This method updates the categories of a content. If token's user does not have "read" on this content's source a 404 HTTP response will be returned.

HTTP method: PUT
Path: /1.0/contents/:id/update_categories
Pagination: no.
Authorization: no.
Extra parameters:

- category_ids: An array containing the new categories to set on the content.

Ignoring a content

Ignores a content. If token's user does not have "read" on content's source a 404 HTTP response will be returned.

HTTP method: PUT
Path: /1.0/contents/:id/ignore
Pagination: no.
Authorization: no.

Attachments

Format

JSON response of an attachment respects following format:


```
{
  "id": "5b87a6b2f042de5f94fabf8d",
  "created_at": "2018-08-30T08:11:30Z",
  "updated_at": "2018-08-30T08:11:30Z",
  "content_type": "image/jpeg",
  "filename": "cat.jpeg",
  "foreign_id": null,
  "size": 700754,
  "url": "http://domain-test.engagement.dimelo.dev/attachments/5b87a6b2f042de5f94fabf8d",
  "video_metadata": [],
  "embed": false,
  "public?": true
}
```


Getting all attachments

This method renders attachments ordered by creation date (descending).

HTTP method: GET
Path: /1.0/attachments
Pagination: yes.
Authorization: no.

Getting an attachment from its id

This method renders an attachment from given id.

HTTP method: GET
Path: /1.0/contents/:id
Pagination: no.
Authorization: Only user that have “read” permission on the attachment’s attached content’s source, if the attachment is not attached to any content it will check the “Publish contents” permission.

Creating an attachment

This method allows you to create a new attachment.

HTTP method: POST
Path: /1.0/contents
Pagination: no.
Extra parameters:

- file: The file you want to upload. See the “Uploading a file” section.
- private: You can create an attachment as private by setting this parameter to “true” or “1”. This parameter IS NOT mandatory, by default all attachment will be created as public.

Authorization: only users that can publish contents.

Events

Format

JSON response of an event respects following format:

```
{
  "id": "5314a71d64204fa46c00000a",
  "created_at": "2014-03-03T16:00:29Z",
  "updated_at": "2014-03-03T16:00:29Z",
  "name": "content.replied",
  "user_id": "4ee91f197aa58d01b500000f",
  "extra_infos": {
    "content_id": "806f6f44bf47db6c5e166cf3",
    "content_thread_id": "9c9903dc3d559a6801eb5441",
    "foreign_id": null
  }
}
```

Notes:

- attributes in extra_infos are optional and unspecified/not guaranteed, don’t rely on it for critical tasks, this is provided as information only
- foreign_id values can be: null

Getting all events

This method renders events ordered by creation date (descending)

HTTP method: GET
Path: /1.0/events
Pagination: yes
Extra parameters:

- q: To filter events on given query. Query works exactly like threads query but only have those keywords: content, content_thread, name_in, created_before, created_after, user. Order can be created_at.desc (default) or created_at.asc.
  e.g.
  q=name_in:”content.replied”%20content_thread:"7f946431b6eebffafae642cc"%20created_after:"2014-03-20"%20user:”4ee91f197aa58d01b500000f”%20order:"created_at.asc"
  - DateTime parameters should be iso8601
  - you can specify multiple value for a given keyword : q=name_in:’content.replied’ name_in:’content.ignored’

  Please refer to Search & filtering parameters for more details.

Authorization: Only users whose role can search event permission.

The following table describes all possible events :

Event Name	Description
api_access_token.created	API access token created
api_access_token.destroyed	API access token destroyed
api_access_token.updated	API access token updated
automatic_exports_task.created	Automatic export task created
automatic_exports_task.destroyed	Automatic export task destroyed
automatic_exports_task.failed	Automatic export task failed
automatic_exports_task.succeed	Automatic export task succeed
automatic_exports_task.updated	Automatic export task updated
category.created	Category created
category.destroyed	Category deleted
category.updated	Category updated
community.created	Community created
community.destroyed	Community deleted

community.updated	Community updated
content.admin_stamped	Answer admin stamped
content.approved	Message approved
content.author_blocked	Identity blocked
content.author_stamped	Answer author stamped
content.author_unblocked	Identity unblocked
content.auto_categorization_infirmed	ICE message categorization corrected
content.auto_categorization_not_precise	ICE message categorization failed
content.auto_categorized	Message categorized by ICE
content.auto_ignored	Message auto archived
content.categorized	Message categorized
content.destroyed	Message deleted
content.discussion_initiated	Discussion initiated
content.discussion_planned	Discussion planned
content.ice_nth_content_ignored	ICE ignored a message
content.ignored	Message ignored
content.imported	Message imported
content.liked	Message liked
content.moderated_ban	Identity blocked
content.moderated_modif	Message edited
content.moderated_nok	Message unpublished
content.moderated_ok	Message approved
content.planned_discussion_synchronized	Planned discussion synchronized
content.published	Message published
content.recategorized	Message recategorized
content.replied	Message replied
content.reply_assistant_used	Reply assistant used to reply to a message

content.retried_synchronization	Message synchronization retried
content.retweeted	Message retweeted
content.source_changed	Source changed
content.stared	Message starred
content.thread_closed	Thread closed
content.thread_opened	Thread opened
content.unliked	Message unliked
content.unpublished	Message unpublished
content.unstamped	Message unstamped
content.unstared	Message unstarred
content.updated	Message updated
content_source.created	Content source created
content_source.destroyed	Content source destroyed
content_source.updated	Content source updated
content_thread.categorized	Thread categorized
content_thread.destroyed	Thread destroyed
content_thread.recategorized	Thread recategorized
custom_field.created	Custom field created
custom_field.destroyed	Custom field deleted
custom_field.updated	Custom field updated
expired_data_purge.deleted	Expired data deleted
extension.created	Extension added
extension.destroyed	Extension removed
extension.updated	Extension updated
folder.created	Folder created
folder.destroyed	Folder destroyed
folder.updated	Folder updated

identity.followed	Identity followed
identity.unfollowed	Identity unfollowed
identity.updated	Identity updated
intervention.assigned	Message assigned
intervention.canceled	Intervention canceled
intervention.closed	Message solved
intervention.deferred	Intervention deferred
intervention.opened	Message engaged
intervention.recategorized	Intervention recategorized
intervention.reopened	Intervention reopened
intervention.updated	Intervention updated
intervention.user_updated	Intervention's user updated
intervention_comment.created	Intervention commented
intervention_comment.destroyed	Intervention destroyed
reply_assistant_knowledge_base_entry.created	Reply assistant knowledge base entry created
reply_assistant_knowledge_base_entry.destroyed	Reply assistant knowledge base entry destroyed
reply_assistant_knowledge_base_entry.updated	Reply assistant knowledge base entry updated
reply_assistant_knowledge_base_version.created	Reply assistant knowledge base version created
reply_assistant_knowledge_base_version.destroyed	Reply assistant knowledge base version destroyed
reply_assistant_knowledge_base_version.updated	Reply assistant knowledge base version updated
reply_assistant_sentence_entry.created	Reply assistant sentence entry created
reply_assistant_sentence_entry.destroyed	Reply assistant sentence entry destroyed
reply_assistant_sentence_entry.updated	Reply assistant sentence entry updated
reply_assistant_sentence_version.created	Reply assistant sentence version created
reply_assistant_sentence_version.destroyed	Reply assistant sentence version destroyed

reply_assistant_sentence_version.updated	Reply assistant sentence version updated
reply_assistant_version_permalink.created	Reply assistant permalink version created
reply_assistant_version_permalink.destroyed	Reply assistant permalink version destroyed
reply_assistant_version_permalink.updated	Reply assistant permalink version updated
role.created	Role created
role.destroyed	Role destroyed
role.updated	Role updated
security.updated	Security settings updated
session.created	A user signed in.
session.destroyed	A user logout.
settings.updated	Settings updated
survey.created	Survey created
survey.destroyed	Survey destroyed
survey.updated	Survey updated
tag.created	Tag created
tag.destroyed	Tag destroyed
tag.updated	Tag updated
team.created	Team created
team.destroyed	Team destroyed
team.updated	Team updated
user.created	Agent created
user.destroyed	Agent destroyed
user.disconnected	Agent disconnected
user.impersonated	Agent impersonated
user.invited	Agent invited
user.notifications_updated	Agent's notifications updated
user.updated	Agent updated

Getting an event from its id

This method renders an event from given id. If token's user role does not have "search event" permission a 404 HTTP response will be returned.

HTTP method: GET
Path: /1.0/events/:id
Pagination: no.
Authorization: Only users who's role can search event permission.

Interventions

Format

JSON response of an intervention respects following format:


```
{
 "id": "3f55c8330da4144afd1c6728",
 "created_at": "2012-05-21T01:15:28Z",
 "updated_at": "2012-05-21T01:19:49Z",
 "source_id": "f18c81948c137d86dac77216",
 "thread_id": "9c9903dc3d559a8801eb5441",
 "content_id": "c93e3586250ff60181b6c2f0",
 "deferred_at": "2012-05-21T01:18:49Z",
 "identity_id": "8a8deed44623a4c44268c266",
 "comments_count": 1,
 "closed": false,
 "closed_at": "2012-05-24T02:00:32Z",
 "custom_fields": {
  "external_id": "342901"
 },
 “category_ids: ["4d0fb475b242228032cbdf6d", "59248c4dae276a021cb296d2"],
 "user_id": "d033e22ae348feb5660fc214",
 "user_replies_count": 1,
 "user_reply_in_average": 84959,
 "user_reply_in_average_bh": 63000,
 "user_reply_in_average_count": 1,
 "first_user_reply_id": "573446514379728247000001",
 "first_user_reply_in": 0,
 "first_user_reply_in_bh": 0,
 "last_user_reply_in": 0,
 "last_user_reply_in_bh": 0,
 "status": "Fermée"
```

}

Getting all interventions

This method renders interventions ordered by creation date (descending). Only interventions in sources where token's user has "read" permission are returned.

HTTP method: GET
Path: /1.0/interventions
Pagination: yes.
Extra parameters:

- thread_id: To filter interventions on given thread id.
- user_id: To filter interventions on given user id.
- identity_group_id: To filter interventions on given identity_group_id. This will return interventions associated to any identity in the indentity_group.
- identity_id: To filter interventions on given identity_id(s). Can be a single value or an array.
- sort: To change the criteria chosen to sort the interventions. The value can be "created_at" or "updated_at".

Authorization: no.

Getting an intervention from its id

This method renders an intervention from given id. If token's user does not have "read" on intervention's source a 404 HTTP response will be returned.

HTTP method: GET
Path: /1.0/interventions/:id
Pagination: no
Authorization: no.

Creating an intervention

This method creates a new intervention or reopens it. In case of success it renders the intervention, otherwise, it renders an error (422 HTTP code). This method opens intervention as access token's user.

HTTP method: POST
Path: /1.0/interventions
Pagination: no.
Extra parameters:

- content_id: The content to create intervention on (mandatory).

Authorization: no, but it renders an error if intervention can't be created or reopened (already opened, etc.).

Cancelling an intervention

This method cancels (destroys) an intervention. It renders intervention itself. If token's user does not have "read" on intervention's source a 404 HTTP response will be returned.

Caveats:

- If the intervention is already being canceled, it will return a 409 error.
- To be able to close an intervention, it must meet the following criteria otherwise a 403 will be raised:
    - Intervention MUST NOT already be closed
    - Intervention MUST NOT have agent replies
    - Access-Token agent MUST have read access on the source

HTTP method: DELETE
Path: /1.0/interventions/:id/cancel
Pagination: no.
Authorization: no, but it renders an error if intervention can't be destroyed (see caveats).

Updating an intervention

This method updates intervention from given attributes and renders it in case of success.

HTTP method: PUT
Path: /1.0/interventions/:id
Pagination: no.
Extra parameters:

- custom_field_values[custom_field_key]: Intervention custom field with key « custom_field_key ». It can be multiple if custom field is multiple or is has multiple_choice type.

Reassigning an intervention

This method updates the user in charge of the intervention

HTTP method: PUT
Path: /1.0/interventions/:id/reassign
Pagination: no.
Authorization: Only users who can update interventions.
Extra parameters:

- user_id: The ID of the new user in charge of the intervention (mandatory).

Closing an intervention

This method closes an intervention.
Caveats:

- If the intervention is already being closed, it will return a 409 error.
- To be able to close an intervention, it must meet the following criteria otherwise a 403 will be raised:
    - Intervention MUST NOT already be closed

- Intervention MUST have agent replies
- Access-Token agent MUST be the owner of the intervention or have the permission to edit permissions
- Access-Token agent MUST have read access on the source

HTTP method: PUT
Path: /1.0/interventions/:id/close
Pagination: no.
Authorization: no, but it renders an error if intervention can't be closed (see caveats)

Categorizing an intervention

This method updates the categories of an intervention. If token's user does not have "read" on the intervention's source a 404 HTTP response will be returned.

HTTP method: PUT
Path: /1.0/interventions/:id/update_categories
Pagination: no.
Authorization: no.
Extra parameters:

- category_ids: An array containing the new categories to set on the intervention.

Intervention comments

Format

JSON response of an intervention comment respects following format:


```
{
  "id": "1514156ae26c48b001c20b2f",
  "user_id": "d033e22ae448aeb5660fc214",
  "videntity_id": "a51dda7c7ff50b61eaea0444",
  "thread_id": "04eac1eee93f793d924ca4a6",
  "intervention_id": "83393e80ae3ae5fed7a35775"
  "body": "Problème récurrent chez nos clients. Il faudrait le mettre dans la notice sur ce modèle de téléphone",
```

"created_at": "2012-06-17T13:00:45Z",
"updated_at": "2012-06-17T13:00:45Z"
}

Getting all intervention comments

This method renders interventions comments ordered by creation date (descending). Only comments in sources where token's user has "read" permission are returned.

HTTP method: GET
Path: /1.0/intervention_comments
Pagination: yes.
Extra parameters:

- intervention_id: To filter comments on given intervention id.
- thread_id: To filter comments on given thread id.
- user_id: To filter comments on given user id.
- identity_id: To filter comments on given identity id.

Authorization: no.

Getting an intervention comment from its id

This method renders an intervention comment from given id. If token's user does not have "read" on comment's source a 404 HTTP response will be returned.

HTTP method: GET
Path: /1.0/intervention_comments/:id
Pagination: no.
Authorization: no.

Creating an intervention comment

This method creates a new intervention comment. In case of success it renders the created comment, otherwise, it renders an error (422 HTTP code). It creates comment as token's user. If token's user does not have "read" on given intervention's source a 404 HTTP response will be returned.

HTTP method: POST
Path: /1.0/intervention_comments
Pagination: no.
Extra parameters:

- body: The comment body (mandatory).
- intervention_id: The comment intervention id (mandatory).
- user_id: The comment user id (mandatory).

Authorization: no.

Deleting an intervention comment

This method destroys an intervention comment. It renders comment itself. If token's user does not

have “read” on comment’s source a 404 HTTP response will be returned.

HTTP method: DELETE
Path: /1.0/intervention_comments/:id
Pagination: no.
Authorization: no.

Status (task view)

This API allows to read and modify agent's availability and busyness while in push mode (task view).

Format

JSON response of an agent's status respects the following format:

```
[{
  "agent_id":"4f4f3a08a90ffb27ee000583",
  "channels":[
    {
      "id":"55794bd9416472d4e8050000",
      "name":"Async",
      "status":"available",
      "busyness":"busy"
    },
    {
      "id":"55794bda416472d4e8060000",
      "name":"Realtime",
      "status":"available",
      "busyness":"unoccupied"
    }
  ],
  "custom_status":null
}]
```

Possible values for the "status" attribute are:

1. null — *the agent is not connected on this channel*
2. "available" — *The agent is available (green/yellow/orange dot)*
3. "away" — *The agent is away (red dot)*

Possible values for the "busyness" attribute are:

1. "unoccupied" — *The agent don't have any task in progress on this channel*
2. "ok" — *The agent has less task in progress than the soft limit allows (green dot)*
3. "busy" — *The agent has more task than the soft limit allows (yellow dot)*
4. "full" — *The agent is at full capacity / hard limit (orange dot)*

The "custom_status" attribute represent the custom "away" status selected, it can either be:

1. null — *The agent is available or in the generic "Away" status*
2. { "id":"5582b1f4776562af9b000008" } — *The id of the custom status*

Get all connected agents status

This method get all currently connected agents & their status.

HTTP method: GET
Path: /1.0/status
Pagination: no.
Authorization: only users that have the right to monitor the task view.

Get a connected agent status

This method get the status of a connected agent.
Returns a 404 if the user does not exist (not_found) or if he's not connected (disconnected).

HTTP method: GET
Path: /1.0/status/:agent_id
Pagination: no.
Authorization: only users that have the right to monitor the task view.

Changing an agent's status

This method updates an agent's availability. Can be used to set either channels statuses OR custom status. If both parameters are provided, ignores custom status. The status parameter **MUST** be either "away" or "available".

HTTP method: PUT
Path: /1.0/status/:agent_id
Pagination: no.
Authorization: only users that have the right to monitor the task view.
Extra parameters:

- status: A hash of channel_id => availability (must contain all channels).
- custom_status_id: id of presence status (optional)

Example data:

status[55794bd9416472d4e8050000]=away&status[55794bda416472d4e8060000]=away

Webhook

This API allows to list, read, modify and delete a webhook.

Format

JSON response of a webhook respects following format:

```
{
  "id": "de6f4800e6e9c6bf875cf396",
  "active": false,
  "staging_use": true,
  "url": "https://example-webhook.staging.com/endpoint",
  "verify_token": "GKaCilcA2DDA0Y",
  "source_filtering_strategy": "all_sources_except",
  "source_filtering_ids": [

  ],
  "registered_events": [
    "intervention.assigned""intervention.canceled",
    "intervention.closed",
    "intervention.deferred",
    "intervention.opened",
    "intervention.reopened",
    "intervention.updated",
    "intervention.user_updated",
    "intervention.reactivated",
    "intervention.recategorized"
  ],
  "api_access_token"=>{
    "id"=>"55794bd9416472d4e8050000",
    "created_at"=>"2011-05-02T22:00:00Z",
    "updated_at"=>"2011-05-02T22:00:00Z"
  },
  "sources_filtering_strategy": "no_source_except",
  "sources_filtering_ids": [
    "4f97fbea7aa58d073900344f",
    "54e1fa99776562f20a510300"
  ]
}
```

Getting all webhooks

This method renders webhooks ordered by active and staging_use (descending).

HTTP method: GET
Path: /1.0/webhooks
Pagination: yes.
Authorization: users having manage_api_access_tokens permission can see all webhooks / users

don't having the manage_api_access_tokens permission can see only the webhooks belonging to access token created by them.

Getting a webhook from its id

This method renders a webhook from given id.

HTTP method: GET
Path: /1.0/webhooks/:id
Pagination: no.
Authorization: users having manage_api_access_tokens permission can see any webhook / users don't having the manage_api_access_tokens permission can see only the webhook in case the webhook is associated to an access token created by them.

Creating a webhook

This method creates a webhook. In case of success it renders the webhook, otherwise, it renders an error (422 HTTP code).

HTTP method: POST
Path: /1.0/webhooks
Pagination: no.
Extra parameters:

- active: true or false, this field is used to enable/disable a webhook.
- label: The label of the webhook (mandatory).
- staging_use: true or false, this field is used to determine if a webhook will be run in staging or production.
- url: The url of a webhook. This is used to determine the endpoint of your webhook, where the data will be submitted.
- verify_token: The token used in your webhook.
- secret: The secret key that will be served as a *X-Dimelo-Secret* header in every request.
- registered_events: An array containing all the events that your webhook wants to subscribe.
- sources_filtering_strategy: String that determines the source filtering strategy, can be "all_sources_except" or "no_source_except". If not specified on webhook creation, will be "all_sources_except".
- sources_filtering_ids: An array of source ids filtering the webhook perimeter depending on the source_filtering_strategy.

Authorization: All users having the manage_api_access_tokens permission or all users having an api access token.

Updating a webhook

This method updates an existing webhook from given attributes and renders it in case of success.

HTTP method: PUT
Path: /1.0/webhooks/:id
Pagination: no.
Extra parameters:

- active: true or false, this field is used to enable/disable a webhook.

- label: The label of the webhook (mandatory).
- staging_use: true or false, this field is used to determine if a webhook will be run in staging or production.
- url: The url of a webhook. This is used to determine the endpoint of your webhook, where the data will be submitted.
- verify_token: The token used in your webhook.
- secret: The secret key that will be served as a *X-Dimelo-Secret* header in every request.
- registered_events: An array containing all the events that your webhook wants to subscribe.
- sources_filtering_strategy: String that determines the source filtering strategy, can be “all_sources_except” or “no_source_except”. If not specified on webhook creation, will be “all_sources_except”.
- sources_filtering_ids: An array of source ids filtering the webhook perimeter depending on the source_filtering_strategy.

Authorization: All users having the manage_api_access_tokens permission or all users having an api access token belonging to the webhook you’re updating.

Deleting a webhook

This method destroys an existing webhook. It renders webhook itself. It renders a 404 if id is invalid.

HTTP method: DELETE
Path: /1.0/webhooks/:id
Pagination: no.
Authorization: All users having the manage_api_access_tokens permission or all users having an api access token belonging to the webhook you’re deleting.

Time sheet

This API allows to list, read and modify a time sheet.

Format

JSON response of a time sheet respects following format:


```
{
 "id": "de6f4800e6e9c6bf875cf396",
 "active": false,
 "label": "Facebook (Time sheet)",
 "created_at": "2011-05-02T22:00:00Z",
 "updated_at": "2011-05-02T22:00:00Z",
 "source_ids": [ "4d0fb475b242128032cbdf6d", "60944e5702bdbfb74ec96141" ],
 "holidays": [
  {
   "name": "Christmas Day",
   "date": "12-25-2011"
  },
  {
        "name": "Easter Sunday",
   "date": "04-01-2011"
  }
 ],
 "monday_hours": “28800-72000”,
 "tuesday_hours": “28800-72000”,
 "wednesday_hours": “28800-72000”,
 "thursday_hours": “28800-72000”,
 "friday_hours": “28800-72000”,
 "saturday_hours": “”,
 "sunday_hours": “”,
}
```


Getting all time sheets

This method renders time sheets ordered by active and label.

HTTP method: GET
Path: /1.0/time_sheets
Pagination: yes.
Authorization: only users that can see time sheets in administration section.

Getting a time sheet from its id

This method renders a time sheet from given id.

HTTP method: GET
Path: /1.0/time_sheets/:id

Pagination: no.
Authorization: only users that can see time sheets in administration section.

Creating a time sheet

This method creates a time sheet. In case of success it renders the time sheet, otherwise, it renders an error (422 HTTP code).

HTTP method: POST
Path: /1.0/time_sheets
Pagination: no.
Extra parameters:

- active: true or false, this field is used to enable/disable a time sheet.
- label: The label of the time sheet (mandatory).
- source_ids: An array containing id of each source using your time sheet.
- holidays_region: A string containing the first two letters of your country (example: "fr"/"us"/"es"), useful to bootstrap default holidays following to a country.
- holidays: An array containing one or more hash of holidays, a holiday must contain a name (string) and a date (string), the date must be in a valid format, a valid format is a format corresponding to your domain's locale).
- monday_hours: this field define the time intervals of the day (in secs). An empty string means that there are no business hours on this day.
  For example: "a-b,c-d":
  - "a" is the beginning of the first interval of the day
  - "b" is the ending of the first interval of the day
  - "c" is the beginning of the second interval of the day
  - "d" is the ending of the second interval of the day
- ...

Authorization: only users that can create time sheet.

Updating a time sheet

This method updates an existing time sheet from given attributes and renders it in case of success.

HTTP method: PUT
Path: /1.0/time_sheets/:id
Pagination: no.
Extra parameters:

- active: true or false, this field is used to enable/disable a time sheet.
- label: The label of the time sheet (mandatory).
- source_ids: An array containing id of each source using your time sheet.
- holidays: An array containing one or more hash of holidays, a holiday must contain a name (string) and a date (string), the date must be in a valid format, a valid format is a format corresponding to your domain's locale).
- monday_hours: this field define the time intervals of the day (in secs). An empty string means that there are no business hours on this day.
  For example: "a-b,c-d":
  - "a" is the beginning of the first interval of the day
  - "b" is the ending of the first interval of the day

    - “c” is the beginning of the second interval of the day
    - “d” is the ending of the second interval of the day
- ...

Authorization: only users that are able to update time sheet.

Deleting a time sheet

This method destroys an existing time sheet. It renders time sheet itself. It renders a 404 if id is invalid.

HTTP method: DELETE
Path: /1.0/time_sheets/:id
Pagination: no.
Authorization: only users that are able to destroy time sheet.

Channels

This API allows to list, read and modify channels.

Format

JSON response of a channel respects following format:


```
{
  "id": "55794bda416472d4e8060000",
  "Name": "Realtime",
  "label": "realtime",
  "created_at": "2011-05-02T22:00:00Z",
  "updated_at": "2011-05-02T22:00:00Z",
  "source_ids": [ "4d0fb475b242128032cbdf6d", "60944e5702bdbfb74ec96141" ],
  "hard_capability": 4,
  "soft_capability": 2,
  "task_timeout_seconds": 3600
}
```


Getting all channels

This method renders all channels ordered by date of creation.

HTTP method: GET
Path: /1.0/channels
Pagination: yes.
Authorization: only users that can see channels.

Getting a channel from its id

This method renders a channel from given id.

HTTP method: GET
Path: /1.0/channels/:id
Pagination: no.
Authorization: only users that can see channels.

Updating a channel

This method updates an existing channel from given attributes and renders it in case of success.

HTTP method: PUT
Path: /1.0/channels/:id
Pagination: no.
Extra parameters:

- name: The name of the channel.
- source_ids: An array containing id of each source assigned to a channel (multiple).

- soft_capability: Number of tasks that can be assigned to agent by the routing before they are considered "occupied".
- hard_capability: Maximum number of tasks that can be assigned to agents.
- task_timeout_seconds: this field defines the time before a task expires (in seconds).

Authorization: only users that are able to update channels.

Settings

This API allows to list, read and modify settings.

Format

JSON response of the settings respects following format:


```
{
 "id": "48cc6703bdae1462ce06a555",
 "created_at": "2010-12-13T09:25:31Z",
 "updated_at": "2016-07-06T08:35:35Z",
 "activity_presence_threshold": 45,
 "activity_tracking": true,
 "beginning_of_week": "monday",
 "category_tagging": true,
 "content_languages": [
  "en",
  "fr",
  "nl"
 ],
 "dashboard": true,
 "deny_iframe_integration": false,
 "disable_password_autocomplete": true,
 "dump_in_preprod": true,
 "expire_password_after": 2592000,
 "expire_password_enabled": true,
 "export_in_seconds": false,
 "fold_useless_contents": true,
 "fte_duration": 7,
 "identity_merge": true,
 "intervention_defer_rates": [
  1800,
  10800,
  86400
 ],
 "intervention_defer_threshold": 43200,
 "intervention_rates": [
  30,
  300,
  600,
  7200,
  86400,
  259200,
  604800
 ],
 "locale": "fr",
 "multi_lang": true,
 "name": "domain-test",
 "password_archivable_enabled": false,
```

```
 "password_archivable_size": 5,
 "password_min_length": 6,
 "password_non_word": false,
 "password_numbers": false,
 "password_recovery_disabled": false,
 "push_enabled": true,
 "reply_as_any_identity": true,
 "rtl_support": true,
 "self_approval_required": true,
 "session_timeout": 360,
 "sharding_key": "domain_test",
 "spellchecking": true,
 "style": "dimelo",
 "third_party_services_disabled": false,
 "timezone": "Paris",
 "track_js": true,
 "type": "demo",
 "urgent_task_threshold": 60,
 "use_system_font": false,
 "browser_notifications_disabled": false,
 "display_only_unknown_bbcode": true,
 "intervention_closing_period": null,
 "use_two_letters_avatars": true
}
```


Getting all settings

This method renders all settings of your domain.

HTTP method: GET
Path: /1.0/settings
Pagination: no.
Authorization: only users that can update settings.

Updating settings

This method updates the current domain settings.

HTTP method: PUT
Path: /1.0/settings
Pagination: no.
Extra parameters:

- activity_presence_threshold: (in hours).
- activity_tracking: Enable activity tracking (Boolean)
- beginning_of_week: (Day of week)
- browser_notifications_disabled: (En/Dis)able the browser notifications (Boolean)
- category_tagging: Activate the forced categorization by source. (Boolean)
- content_languages: (See format)
- dashboard: Activate the dashboard (Boolean)
- deny_iframe_integration: Prevent the DD to be embed by other websites (Boolean)

- display_only_unknown_bbcode: Control whether or not all BBCodes are display in RingCentral Engage Digital (Boolean)
- disable_password_autocomplete: (Boolean)
- expire_password_after: password expiration delay (in seconds)
- expire_password_enabled: enable password expiration (Boolean)
- export_in_seconds: provide durations in seconds in export (Boolean)
- fold_useless_contents: fold archived contents (Boolean)
- fte_duration: FTE data period (in hours)
- identity_merge: enable identity merge (Boolean)
- intervention_closing_period: Threshold time from which interventions will be automatically closed, leave blank to disable automatic intervention closing (time in seconds)
- intervention_defer_rates: (Array of times in seconds)
- intervention_defer_threshold: (in seconds)
- intervention_rates: (Array of times in seconds)
- locale: locale code (String)
- multi_lang: activate multi language support for messages (Boolean)
- name: Name of the RingCentral Engage Digital (String)
- password_archivable_enabled: prohibit reusing olf passwords (Boolean)
- password_archivable_size: number of archived passwords
- password_min_length: minimum character length
- password_non_word: should contain at least 1 non alphanumeric char (Boolean)
- password_numbers: should contain at least 1 number (Boolean)
- password_recovery_disabled: disable password recovery by email (Boolean)
- push_enabled: Enable push mode (Boolean)
- reply_as_any_identity: Enable reply as any identity (Boolean)
- rtl_support: Enable right to left support (Boolean)
- self_approval_required: Allow authors to ask approval of their messages (Boolean)
- session_timeout: Session timeout (in minutes)
- spellchecking: Enable spellchecking (Boolean)
- style: Defines the DD's design (String)
- third_party_services_disabled: Disable third-party services (tracking...) (Boolean)
- timezone: Use the timezone endpoint to get the timezone name (String)
- track_js: Track JS errors (Boolean)
- type: Can be 'demo', 'production' or 'archived'
- urgent_task_threshold: Chat max response time (in seconds)
- use_system_font: Experimental (Boolean)
- use_two_letters_avatars: (En/Dis)able two letter avatar usage (Boolean)

Authorization: only users that are able to update settings.

Locales

This API allows to list all available locales. The interface field determine if the locale is available as an interface configuration.

Format

JSON response of locales respects following format:

```
[
 {
  "name":"Afrikaans",
  "code":"af",
  "interface":false
 },
 {
  "name":"English",
  "code":"en",
  "interface":true
 },
 {
  "name":"Français",
  "code":"fr",
  "interface":true
 },
 {
  "name":"עברית",
  "code":"iw",
  "interface":false
 },
 {
  "name":"日本語",
  "code":"ja",
  "interface":false
 },
 {
  "name":"Polski",
  "code":"pl",
  "interface":false
 }
]
```

Getting all locales

This method renders all available locales.

HTTP method: GET
Path: /1.0/locales
Pagination: no.

Authorization: No.

Timezones

This API allows to list all available timezones.

Format

JSON response of timezones respects following format:


```
[
 {
  "name":"American Samoa",
  "utc_offset":-39600
 },
 {
  "name":"Eastern Time (US & Canada)",
  "utc_offset":-18000
 },
 {
  "name":"UTC",
  "utc_offset":0
 },
 {
  "name":"Paris",
  "utc_offset":3600
 },
 {
  "name":"Seoul",
  "utc_offset":32400
 },
 {
  "name":"Tokyo",
  "utc_offset":32400
 },
 {
  "name":"Melbourne",
  "utc_offset":36000
 }
]
```


Getting all timezones

This method renders all available timezones.

HTTP method: GET
Path: /1.0/timezones
Pagination: No.

Authorization: No.

Presence statuses

This API allows to list, read and modify presence statuses.

Format

JSON response of a presence status respects following format:

```
{
  "id": "5775200c1b5321672cd42529",
  "created_at": "2011-05-02T22:00:00Z",
  "updated_at": "2011-05-02T22:00:00Z",
  "name": "Meeting"
}
```

Getting all presence statuses

This method renders all presence statuses ordered by name (in alphabetical order).

HTTP method: GET
Path: /1.0/presence_status
Pagination: no.
Authorization: only users that have the right to monitor the task view.

Getting a presence status from its id

This method renders a presence status from given id.

HTTP method: GET
Path: /1.0/presence_status/:id
Pagination: no.
Authorization: only users that have the right to monitor the task view.

Creating a presence status

This method creates a presence status. In case of success it renders the presence status, otherwise, it renders an error (422 HTTP code).

HTTP method: POST
Path: /1.0/presence_status
Pagination: no.
Extra parameters:

- name: The name of the presence status.

Authorization: only users that have the right to monitor the task view.

Updating a presence status

This method updates an existing presence status from given attributes and renders it in case of success.

HTTP method: PUT
Path: /1.0/presence_status/:id
Pagination: no.
Extra parameters:

- name: The name of the presence status.

Authorization: only users that have the right to monitor the task view.

Deleting a presence status

This method destroys an existing presence status. It renders presence status itself. It renders a 404 if id is invalid.

HTTP method: DELETE
Path: /1.0/presence_status/:id
Pagination: no.
Authorization: only users that have the right to monitor the task view.

Tasks

This API allows to list, read and transfer tasks.

Format

JSON response of the tasks respects following format:


```
{
  "id": "583d895f21b1347296ce7c82",
  "created_at": "2016-11-29T13:57:51Z",
  "updated_at": "2017-03-01T13:42:46Z",
  "channel_id": "55794bd9416472d4e8050000",
  "priority": 10,
  "content_id": "583d896221b1347296ce7ca3",
  "intervention_id": "583d898e21b13468ac3988e6",
  "thread_id": "04eac1eee93e793d924ca4a6",
  "agent_ids": [
    "55eea23d7765624299000002",
    "572721c5776562278b0003bb"
  ],
  "accepted_at": "2017-03-01T13:42:46Z",
  "completed_at": "2016-11-29T13:58:38Z",
  "expire_at": "2017-03-01T14:42:42Z",
  "locked_at": null,
  "transfered_at": "2017-03-01T13:42:42Z",
  "step": "realtime_default_target"
```

}

Getting all tasks

This method renders tasks ordered by priority (highest first) and then by creation date (latest first).

HTTP method: GET
Path: /1.0/tasks
Pagination: yes.
Extra parameters:

- queue: To filter tasks on given queue name (filters on the “global” queue by default).The most commonly used queues are:
  - global : contains every task pending assignation
  - workbin_{agent_id}: contains every tasks assigned to the {agent_id} agent
  - history: contains every processed tasks
  - undelivered: contains every undelivered tasks

  If queue is set to “workbins” all the tasks currently in a workbin will be returned.
- channel_id: To filter tasks on given channel id.
- step: To filter tasks on the step they’re currently in.

Authorization: only users that can read tasks.

Getting a task from its id

This method renders a task from given id.

HTTP method: GET
Path: /1.0/tasks/:id
Pagination: no.

Transferring a task

This method transfers an existing task and renders it in case of success.

HTTP method: POST
Path: /1.0/tasks/:id/transfer
Pagination: no.
Extra parameters:

- agent_ids: List of agents to transfer the task to (multiple).
- bypass: Force the transfer to the first agent in agent_ids if set. When bypass is used, agent_ids need to be provided and the other parameters will be ignored.
- category_ids: Filter agents receiving the task depending on their categories.
- language: Filter agents receiving the task depending on their spoken languages.
- team_ids: Filter agents receiving the task depending on their teams.
- comment: Add a comment to the task.

Authorization: only users that have the right to monitor the task view.

Move a task to another queue

This method changes a task queue and renders it in case of success. Only accepts “undelivered” and special queue defined in topology (e.g. triage)

HTTP method: POST
Path: /1.0/tasks/:id/move
Pagination: no.
Extra parameters:

- queue: Name of the queue task has to be moved in.

Authorization: only users that have the right to monitor the task view.

Complete a task

This method completes a task.

HTTP method: POST
Path: /1.0/tasks/:id/complete
Pagination: no.
Authorization: for that call to succeed several conditions are required:

- the task **must be in an agent’s workbin**.
- the access token user **must** either **own the task** or **be able to monitor all tasks** or **be able to monitor his team’s tasks** if the task is owned by one of his team member.

Topologies

This API allows to list, create, activate, read, modify and delete topologies.

Format

JSON response of a topology respects following format:


```
{
  "id": "58bd903121b1340ed87cf545",
  "created_at": "2017-03-06T16:37:05Z",
  "updated_at": "2017-03-06T16:37:05Z",
  "config": {
    "inputs": {
      "inbound": "import_step",
      "transfered": "transfer_step"
    },
    "steps": [
      {
        "links": {
          "async": "triage",
          "realtime": "triage"
        },
        "_type": "Routing::ImportStep",
        "label": "import_step"
      },
      {
        "links": {
          "async": "triage",
          "realtime": "triage"
        },
        "_type": "Routing::TransferStep",
        "label": "transfer_step"
      },
      {
        "links": {},
        "_type": "Routing::Enqueue",
        "label": "triage",
        "queue": "triage"
      }
    ]
  },
  "name": "Default Topology"
}
```


Getting all topologies

This method renders all topologies ordered by name (in alphabetical order).

HTTP method: GET

Path: /1.0/topologies
Pagination: yes.
Authorization: only users that have the right to manage topologies

Getting a topology from its id

This method renders a topology from given id.

HTTP method: GET
Path: /1.0/topologies/:id
Pagination: no.
Authorization: only users that have the right to manage topologies.

Creating a topology

This method creates a topology. In case of success it renders the topology, otherwise, it renders an error (422 HTTP code).

HTTP method: POST
Path: /1.0/topologies
Pagination: no.
Extra parameters:

- name: The name of the topology.
- Json_config: The json describing the topology.

Authorization: only users that are techteam.

Updating a topology

This method updates an existing topology from given attributes and renders it in case of success.

HTTP method: PUT
Path: /1.0/topologies/:id
Pagination: no.
Extra parameters:

- name: The name of the topology.
- Json_config: The json describing the topology.

Authorization: only users that have the right to manage topologies.

Activating a topology

This method activates an existing topology from given attributes and renders it in case of success.

HTTP method: PUT
Path: /1.0/topologies/:id/activate
Pagination: no.
Authorization: only users that are techteam.

Deleting a topology

This method destroys an existing topology. It renders topology itself. It renders a 404 if id is invalid.

HTTP method: DELETE
Path: /1.0/topologies/:id
Pagination: no.
Authorization: only users that are techteam.

Reply Assistant

Entries

Format

JSON response of a reply assistant entry respects following format:


```
{
  "id": "513e5ca00f4ca191470027df",
  "created_at": "2013-03-11T22:37:20Z",
  "updated_at": "2017-05-05T16:51:48Z",
  "label": "Attente au bureau de poste",
  "version_ids": [
    "513e5ca00f4ca191470027e0"
  ],
  "source_ids": [
    "513ecb6465dbc56b86d45cc"
  ],
  "foreign_id": "2",
  "shortcuts": "fermé",
  "category_ids": [
    "89bc4984bce684bce486bc8"
  ],
  "entry_group_id": "562f784277656257df000021"
}
```


Getting all entries

This method renders all entries ordered by creation date (ascending).

HTTP method: GET
Path: /1.0/reply_assistant/entries
Pagination: yes.
Authorization: only users that have the right to manage reply assistant

Getting an entry from its id

This method renders an entry from given id.

HTTP method: GET
Path: /1.0/reply_assistant/entries/:id
Pagination: no.
Authorization: only users that have the right to manage reply assistant.

Creating an entry

This method creates a reply assistant entry. In case of success it renders the entry, otherwise, it renders an error (422 HTTP code).

HTTP method: POST
Path: /1.0/reply_assistant/entries
Pagination: no.
Authorization: only users that have the right to manage reply assistant.

Updating an entry

This method updates an existing entry from given attributes and renders it in case of success.

HTTP method: PUT
Path: /1.0/reply_assistant/entries/:id
Pagination: no.
Extra parameters:

- label: The name of the topology.
- foreign_id: The internal/company id of the entry. This is used to match RingCentral Engage Digital entry's id with the company one. Example: KB042.
- category_ids: To restrict the entry to a set of RingCentral Engage Digital categories. Then, KB entries that do not match message's categories to which you are replying will not be suggested.
- shortcuts: entry shortcuts
- entry_group_id: Entry group id
- source_ids: Source ids (array)

Authorization: only users that have the right to manage topologies.

Deleting an entry

This method destroys an existing entry. It renders the entry itself. It renders a 404 if id is invalid.

HTTP method: DELETE
Path: /1.0/reply_assistant/entries/:id
Pagination: no.
Authorization: only users that have the right to manage reply assistant.

Versions

Format

JSON response of a version respects following format:


```
{
    "id": "513e5ca00f4ca191470027e0",
    "created_at": "2013-03-11T22:37:20Z",
    "updated_at": "2017-05-05T16:51:48Z",
    "body": "Nous vous invitons à consulter les heures de fréquentation affichées dans votre bureau de poste.",
    "entry_id": "513e5ca00f4ca191470027df",
    "source_ids": [],
    "format": "text",
    "attachments_count": 0,
    "language": "fr"
  }
```


Getting all versions

This method renders all reply assistant versions ordered by creation date (ascending).

HTTP method: GET
Path: /1.0/reply_assistant/versions
Pagination: yes.
Authorization: only users that have the right to manage reply assistant.

Getting a version from its id

This method renders a version from given id.

HTTP method: GET
Path: /1.0/reply_assistant/versions/:id
Pagination: no.
Authorization: only users that have the right to manage reply assistant.

Creating a version

This method creates a reply assistant version. In case of success it renders the version, otherwise, it renders an error (422 HTTP code, 404 if the entry_id is invalid).

HTTP method: POST
Path: /1.0/reply_assistant/versions
Pagination: no.
Extra parameters:

- body: Body of the version
- entry_id: Reply assistant entry id (mandatory)
- source_ids: Source ids (array)
- attachment_ids: Attachments ids (array)
- format: Either “text” or “html”

- language: Language (ex: “fr”)

Authorization: only users that have the right to manage reply assistant.

Updating a version

This method updates an existing version from given attributes and renders it in case of success.

HTTP method: PUT
Path: /1.0/reply_assistant/versions/:id
Pagination: no.
Extra parameters:

- body: Body of the version
- entry_id: Reply assistant entry id
- source_ids: Source ids (array)
- attachment_ids: Attachment ids (array) **Warning: you must include already linked attachments if you want to keep them**
- format: Either “text” or “html”
- language: Language (ex: “fr”)

Authorization: only users that have the right to manage reply assistant.

Deleting a version

This method destroys an existing version. It renders the version itself. It renders a 404 if id is invalid.

HTTP method: DELETE
Path: /1.0/reply_assistant/versions/:id
Pagination: no.
Authorization: only users that have the right to manage reply assistant.

Groups

Format

JSON response of an entry group respects following format:


```
{
    "id": "562f784277656257df000021",
    "created_at": "2015-10-27T13:12:34Z",
    "updated_at": "2015-10-27T13:12:34Z",
    "name": "Technical",
    "entry_ids": [
      "513e5ca00f4ca191470027df",
      "513e5f677aa58d93710025b0",
      "513e5f8fa90ffb6736002717"
    ],
    "autocomplete": true,
    "position": 0
  }
```


Getting all groups

This method renders all groups ordered by creation date (ascending).

HTTP method: GET
Path: /1.0/reply_assistant/groups
Pagination: yes.
Authorization: only users that have the right to manage reply assistant

Getting a group from its id

This method renders an entry group from given id.

HTTP method: GET
Path: /1.0/reply_assistant/groups/:id
Pagination: no.
Authorization: only users that have the right to manage reply assistant.

Creating a group

This method creates an entry group. In case of success it renders the group, otherwise, it renders an error (422 HTTP code).

HTTP method: POST
Path: /1.0/reply_assistant/groups
Pagination: no.
Extra parameters:

- name: The name of the group.
- entry_ids: List of the entry in this group.
- autocomplete: Used for autocompletion in chat.

- position: Used to determine the order of the groups in the interface, in ascending order.

Authorization: only users that have the right to manage reply assistant.

Updating a group

This method updates an existing group from given attributes and renders it in case of success.

HTTP method: PUT
Path: /1.0/reply_assistant/groups/:id
Pagination: no.
Extra parameters:

- name: The name of the group.
- entry_ids: List of the entry in this group.
- autocomplete: Used for autocompletion in chat.
- position: Used to determine the order of the groups in the interface, in ascending order.

Authorization: only users that have the right to manage reply assistant.

Deleting a group

This method destroys an existing group. It renders the group itself. It renders a 404 if id is invalid.

HTTP method: DELETE
Path: /1.0/reply_assistant/groups/:id
Pagination: no.
Authorization: only users that have the right to manage reply assistant.

Survey Response

Format

JSON response of an intervention respects following format:


```
{
 "id":"5a60e5e0eb70256ab08a8d21",
 "created_at":"2018-01-18T18:22:24Z",
 "updated_at":"2018-01-18T18:22:24Z",
 "submitted_at":"2018-01-18T18:17:23Z",
 "url":{
  "i":"brqjhp2qlbg3up79gv",
  "t":"5a60de91d6cb0093319d45c1",
  "slanguage":"French",
  "sglocale":"fr"
 },
 "main_indicator":1,
 "main_indicator_scaled":0.2,
 "intervention_id":"5a60dec4d6cb00501aa54cba",
 "survey_id":"55c9f46d776562784b000438",
```

```
 "source_id":"54e1fa99776562f20a510300",
 "user_id":"5a5dfffe7a8beb217d87ad5e",
 "foreign_id":139,
 "answers":{
  "Are you satisfied with the response we provided ?":"Very unsatisfied"
 }
}
```


Getting a survey response from its id

This method renders a survey response from given id. If token's user does not have "read" or "manage" role on surveys, a 403 HTTP response will be returned.

HTTP method: GET
Path: /1.0/survey_responses/:id
Pagination: no
Authorization: yes.