Use the KnowledgeOwl API

Last modified on 11/02/2026 11:29 am EST

An introduction to using our API.

Introducing the KnowledgeOwl API

Feature availability
This feature is available on select plans.

The KnowledgeOwl REST API provides a large collection of endpoints, allowing you to perform many of the same tasks that can be done through the app. This can be helpful in a variety of scenarios, including:

- Automating repeated tasks
- Bulk operations, such as extracting custom data reports
- Integrating with other tools. For example, you can use the API with Webhooks by Zapier to integrate KnowledgeOwl with other software.

On this page, learn how to query the API, including authentication and filtering. It is specific to the KnowledgeOwl API, and assumes you have some familiarity with REST APIs. If you are new to working with APIs, you may want to take a look at Working with APIs first.

Refer to Endpoint reference for details of available endpoints.

Authentication

The API uses basic authentication. This means all API requests need to include a username and password. For the KnowledgeOwl API, the username must be an API key. The password can be any dummy value, such as x. Refer to API keys for information on creating, editing, and deleting your API keys.

Our API is an HTTPS-only API. Any non-secure requests will return an error.

API keys

Authors with **Full Admin permissions** can view and manage API keys, regardless of their author role.

To view and manage your API keys, go to **Account > API**.

Copyright © 2015 – 2025 Silly Moose, LLC. All rights reserved.
Powered by KnowledgeOwl knowledge base software.

Feature availability
This feature is available on select plans.

Create a new key

To generate a new API key:

1. Go to **Account** > **API**. The **API keys** page opens.
2. Select **+ Add new API key**. The **Create API key** modal opens.
3. Enter a **Key purpose**. The purpose should identify any integrations, processes, and so on that would help someone know what this key is used for and whether it is still used.
4. Use the **Knowledge base access** dropdown to select a knowledge base you want this key to have access to.
5. Once you select a knowledge base, use the checkboxes to select which **Resources** and which Actions the key has permission to complete on those resources. Refer to API key permission-endpoint mapping for a detailed breakdown of which resources map to which endpoint and which actions map to which call type.

   Only use what you need
   Limit permissions to only those actions needed for the tasks you want to complete. To pull information from KnowledgeOwl, only **Read** permissions are needed.

6. If you'd like this key to have access to other knowledge bases, select **+ Add another knowledge base** and repeat the resource/action permissions process for that knowledge base.
7. Use the **Account-level access** action boxes to grant the key access to resources at the account, rather than knowledge base, level. These checkboxes grant access to much of the information available in the **Account** menu.
8. Once you've finished configuring your API key, select **Create**. The **API key created** modal opens.
9. Select **Copy** to copy your new API key. Be sure to save this somewhere, as it's the only time you'll be able to access the full key.
10. Select **Done** to close the modal.

The **Active keys** list updates to display your new key. It is now ready to use.

Edit an existing API key

You may want to change the purpose listed for an API key or change the actions allowed on certain resources.

Copyright © 2015 – 2025 Silly Moose, LLC. All rights reserved.
Powered by KnowledgeOwl knowledge base software.

To update an existing API key:

1. Go to **Account > API**. The **API keys** page opens.
2. Select the gear icon in the **Actions** column next to the API key you want to edit:

   The **Edit API key** modal opens.
3. Make any changes to the **Key purpose**, **Knowledge base access**, or **Account-level access** sections.
4. Be sure to **Save** your changes.

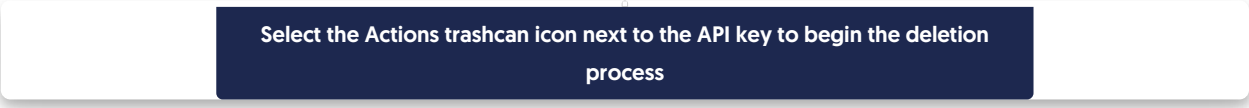
The key's permissions immediately update. If you removed existing permissions, those will no longer be available to the key. If you added them, they'll be immediately available.

Delete a key

Look before you leap
Deleted API keys are not recoverable. Be sure you no longer need a key before deleting it.

To delete an API key you no longer need:

1. Go to **Account > API**. The **API keys** page opens.
2. Select the red trashcan icon in the Actions column next ti the API key you want to delete:

   The **Delete API key** modal opens.
3. Deleting an API key cannot be undone. If you're sure you want to delete the API key, select **OK** to confirm deletion.

The key is now deleted and all calls made using it will fail.

Find your knowledge base ID or project ID

Many API calls require you to pass in a knowledge base ID, known as the `project_id` in the API documentation.

To find your knowledge base ID:

1. Go to **Pages**.
2. Your knowledge base/project ID is in the URL on this page, the string of letters and numbers that appears after `/articles/id/`.

Copyright © 2015 – 2025 Silly Moose, LLC. All rights reserved.
Powered by KnowledgeOwl knowledge base software.

For example, if my URL on the Pages page is `https://app.knowledgeowl.com/kb/articles/id/11abc2d3e45fg678h9012345`**, my project ID is** `11abc2d3e45fg678h9012345`**.**

Pagination and limiting results

Limit results

You can limit the number of objects returned per page in your query results. To do so, add `"limit": <number>` **to your query parameters. For example, to get all articles in your knowledge base, limited to 20 objects per page, add** `"limit": 20` **to your query parameters.**

Accessing pages

If your query returns a large amount of data, or you set a limit on results, KnowledgeOwl paginates the results and returns the first page. If your data is paginated, the response will include the following:

```
"page_stats": {
 "total_records": <number>
 "total_pages": <number>
}
```

To access later pages, you have to make multiple calls, requesting each page in turn. Add `"page": <number>`**, along with any other query parameters. For example, to access the fourth page of data, include** `"page": 4` **in your query parameters.**

Usage limits

The KnowledgeOwl API is not intended for extremely heavy use. If you anticipate making more than 1500 requests per five minutes, please contact us to discuss your requirements.

Input and output formats

KnowledgeOwl's API accepts the following input types:

- **Standard REST HTTP headers -** `?field1=value&field2=value&array1[]=value1&array1[]=value2`
- **Application/JSON -** `'{"field1": "value", "field2": "value", "array1": ["value1", "value2"]}'`

KnowledgeOwl's API can return the following output types as specified by the call endpoint:

- **JSON (default) —** `https://app.knowledgeowl.com/api/head/<object>.json`
- **JSONP —** `https://app.knowledgeowl.com/api/head/<object>.jsonp?callback=<functionName>`
- **HTML —** `https://app.knowledgeowl.com/api/head/<object>.html`
- **PSON —** `https://app.knowledgeowl.com/api/head/<object>.pson`

Date formats

For PUT or POST API calls, dates can be formatted in one of two ways:

Copyright © 2015 – 2025 Silly Moose, LLC. All rights reserved.
Powered by KnowledgeOwl knowledge base software.

- **Unix timestamps**: Also known as epoch timestamps. Refer to Unix Time Stamp if you're unfamiliar with using this format. Example: 1701745258.
- **Y-m-d H:i:s format**: Four-digit year, two-digit month and day, military time format for GMT or its current equivalent. Example: "2023-10-05 18:48:00".

Query operators

The KnowledgeOwl API supports a selection of query operators, allowing you to add logic to the body of your API calls. This means you can do things like search for a particular name or term, or filter your results. This section lists all the available operators, with examples of how to use them.

String and array comparisons

```
//Example article object
{
 "id": "9999",
 "name": "Article 1",
 "parents": [
  "1234",
  "4321"
 ]
}
```

$in - in array

```
//Use $in to find the example article based off of the "name" field, which is a string
//This filter matches any article that has a "name" of "Article 1", OR "Article 2"
curl -u {{API Key}}:X
 -H "Content-type: application/json"
 -X GET "https://app.knowledgeowl.com/api/head/article.json"
 -d '{"name": {"$in": ["Article 1", "Article 2"]}}'
```

```
//Use $in to find the example article based off of the "parents" field, which is an array
//This filter matches any article that contains the value "1234" within the "parents" array
curl -u {{API Key}}:X
 -H "Content-type: application/json"
 -X GET "https://app.knowledgeowl.com/api/head/article.json"
 -d '{"parents": {"$in": ["1234"]}}'
```

$nin - not in array

```
//Use $nin to exclude the example article based off of the "name" field, which is a string
//This filter excludes any articles that have a "name" of "Article 1" OR "Article 2"
curl -u {{API Key}}:X
 -H "Content-type: application/json"
 -X GET "https://app.knowledgeowl.com/api/head/article.json"
 -d '{"name": {"$nin": ["Article 1", "Article 2"]}}'
```

Copyright © 2015 – 2025 Silly Moose, LLC. All rights reserved.
Powered by KnowledgeOwl knowledge base software.

```
//Use $nin to exclude the example article based off of the "parents" field, which is an array
//This filter excludes any articles that have the value "1234" within the "parents" array
curl -u {{API Key}}:X
  -H "Content-type: application/json"
  -X GET "https://app.knowledgeowl.com/api/head/article.json"
  -d '{"parents": {"$nin": ["1234"]}}'
```

$ne - not equal to

```
//Use $ne to find all articles that do not have a "status" of "deleted"
curl -u {{API Key}}:X
  -H "Content-type: application/json"
  -X GET "https://app.knowledgeowl.com/api/head/article.json"
  -d '{"status": {"$ne": "deleted"}}'
```

$regex - regular expression string comparison

```
//Use regex to find the example article based off of the "name" field, which is a string
//This filter matches any article that contains the string "article" in the "name" field
curl -u {{API Key}}:X
  -H "Content-type: application/json"
  -X GET "https://app.knowledgeowl.com/api/head/article.json"
  -d '{"name": {"$regex": "article", "$options": "i"}}'

//The "$options" value of "i" modifies the regular expression to be case insensitive
//Available regex modifiers are "i", "m", "x", and "s"
```

Date and numeric value comparisons



Date formatting

Dates are returned via the API in human readable format based on the Timezone and Date Format settings in your knowledge base. If you're filtering API objects based off of dates, the API expects Unix timestamps to be passed in. Refer to API date formats for more information.

```
//Example article object
{
  "id": "1234",
  "index": 2,
  "date_created": "11/07/2015 11:06 am GMT", //Equal to 1446894360 unix timestamp
}
```

$gt - greater than

```
//Use $gt to include the example article based off of the "index" field, which is numeric
curl -u {{API Key}}:X
  -H "Content-type: application/json"
  -X GET "https://app.knowledgeowl.com/api/head/article.json"
  -d '{"index": {"$gt": 1}}'
```

Copyright © 2015 – 2025 Silly Moose, LLC. All rights reserved.
Powered by KnowledgeOwl knowledge base software.

```
//Use $gt to include the example article based off of the "date_created" field, which is a timestamp
//1446807960 is equal to 11/06/2015 11:06 am GMT
curl -u {{API Key}}:X
  -H "Content-type: application/json"
  -X GET "https://app.knowledgeowl.com/api/head/article.json"
  -d '{"date_created": {"$gt": "1446807960"}}'
```

$gte - greater than or equal to

```
//Use $gte to include the example article based off of the "index" field, which is numeric
curl -u {{API Key}}:X
  -H "Content-type: application/json"
  -X GET "https://app.knowledgeowl.com/api/head/article.json"
  -d '{"index": {"$gte": 2}}'
```

```
//Use $gte to include the example article based off of the "date_created" field
//1446894360 is equal to 11/07/2015 11:06 am GMT
curl -u {{API Key}}:X
  -H "Content-type: application/json"
  -X GET "https://app.knowledgeowl.com/api/head/article.json"
  -d '{"date_created": {"$gte": "1446894360"}}'
```

$lt - less than

```
//Use $lt to include the example article based off of the "index" field, which is numeric
curl -u {{API Key}}:X
  -H "Content-type: application/json"
  -X GET "https://app.knowledgeowl.com/api/head/article.json"
  -d '{"index": {"$lt": 3}}'
```

```
//Use $lt to include the example article based off of the "date_created" field
//1446980760 is equal to 11/08/2015 11:06 am GMT
curl -u {{API Key}}:X
  -H "Content-type: application/json"
  -X GET "https://app.knowledgeowl.com/api/head/article.json"
  -d '{"date_created": {"$lt": "1446980760"}}'
```

$lte - less than or equal to

```
//Use $lte to include the example article based off of the "index" field, which is numeric
curl -u {{API Key}}:X
  -H "Content-type: application/json"
  -X GET "https://app.knowledgeowl.com/api/head/article.json"
  -d '{"index": {"$lte": 2}}'
```

```
//Use $lte to include the example article based off of the "date_created" field
//1446894360 is equal to 11/07/2015 11:06 am GMT
curl -u {{API Key}}:X
  -H "Content-type: application/json"
  -X GET "https://app.knowledgeowl.com/api/head/article.json"
  -d '{"date_created": {"$lte": "1446894360"}}'
```

Copyright © 2015 – 2025 Silly Moose, LLC. All rights reserved.
Powered by KnowledgeOwl knowledge base software.

Logic filtering

```
//Example article object
{
  "id": "9999",
  "name": "Article 1",
  "index": 10,
  "status": "deleted",
}
```

$and - must match ALL specified filters

```
//Use $and to include the example article based off of multiple filters
//Article must have an "index" that is greater than 1 AND less than 20
curl -u {{API Key}}:X
  -H "Content-type: application/json"
  -X GET "https://app.knowledgeowl.com/api/head/article.json"
  -d '{"$and": [{"index": {"$gt": 9}}, {"index": {"$lt": 20}}]}'
```

$or - must match ONE of the specified filters

```
//Use $or to include the example article based off of multiple filters
//Article can either have an "index" that is greater than 20 OR less than 11
curl -u {{API Key}}:X
  -H "Content-type: application/json"
  -X GET "https://app.knowledgeowl.com/api/head/article.json"
  -d '{"$or": [{"index": {"$gt": 20}}, {"index": {"$lt": 11}}]}'
```

$nor - must NOT match ANY of the specified filters

```
//Use $nor to include the example article based off of multiple filters
//Article must NOT have an "index" that is greater than 11 NOR less than 9
curl -u {{API Key}}:X
  -H "Content-type: application/json"
  -X GET "https://app.knowledgeowl.com/api/head/article.json"
  -d '{"$nor": [{"index": {"$gt": 11}}, {"index": {"$lt": 9}}]}'
```

API calls in snippets

Feature availability
This feature is available on select plans.

Sometimes you might want to extend KnowledgeOwl's built in functionality by using client-side API calls in your knowledge base.

To keep the security of your knowledge base intact, we encourage you to use an API merge code within a snippet to handle this functionality. When the page renders, the merge code will be replaced with a unique, single-use URL that does not contain your API key or any account specific information.

Copyright © 2015 – 2025 Silly Moose, LLC. All rights reserved.
Powered by KnowledgeOwl knowledge base software.

Before you begin

In order for API calls in snippets to work, you must have at least one active API key in your account with read-only permissions for this knowledge base and for the objects you're querying.

If you don't have an available API key that meets this requirement, the merge code URL returns an error.

Refer to API keys for detailed instructions on creating and editing API key permissions. Refer to API key permission-endpoint mapping for more details on which resources map to which endpoints.

Construct the merge code

Let's take a look at how to construct this merge code and then what we can do with it. Here's a template for the merge code:

```
[ko_api(API Object|{JSON API filter})]
```

Let's explore each required piece in turn:

1. **The outer wrapper:** `[ko_api( )]`
   This wrapper and everything within it will be replaced server side with a unique URL at the time of page rendering.

2. **The API Object**
   We first pass in information about the API object or endpoint we're getting information about followed by the pipe symbol `|`. For example, if you want to get information about categories, you'd use: `category |`. Refer to our API endpoint reference for a complete list of available endpoints/objects.

3. **JSON API Filter**
   We then pass in a JSON-formatted string containing a valid API filter. Let's say we want to query for the 5 newest categories in our knowledge base that aren't deleted. We'll need to use filters for our `project_id` (knowledge base ID) and the category's `status` ("active"), and then use query operators to `sort` by the most recently created and `limit` to 5. Our JSON string might look like this:

   ```
   //project_id = Knowledge base ID
   {"project_id": "123456", "status": "active", "limit": 5, "sort": {"date_created": 1}}
   ```

When we put all the parts from our example together, we get the following fully constructed merge code:

```
[ko_api(category|{"project_id": "123456", "status": "active", "limit": 5, "sort": {"date_created": 1}})]
```

Knowledge base variables

The API merge code is replaced server side, so you can't use Javascript variables within it. However, we have a lot of variables you can use to reference information about the current page and the current reader viewing the page:

Variable Name	Variable Value	Example JSON

Copyright © 2015 – 2025 Silly Moose, LLC. All rights reserved.
Powered by KnowledgeOwl knowledge base software.

Variable Name	Variable Value	Example JSON
%cur_kb_id%	The ID of the knowledge base that is currently being viewed.	"project_id": "%cur_kb_id%"
%cur_cat_id%	If viewing a category: returns the ID of the category currently being viewed; If viewing an article: returns the ID of the category in which the article is contained. If the article is in a subcategory, this is the category *immediately above* this article in the hierarchy, not it's ultimate top-level parent.	"category": "%cur_cat_id%"
%cur_top_cat_id%	The ID of the top most parent category that the current article or category is in.	"category": "%cur_top_cat_id%"
%cur_parent_cat_ids%	Array of all parent category IDs that the current article or category is in.	"category": {"$in": "%cur_parent_cat_ids%"}
%cur_art_id%	The ID of the article that is currently being viewed	"id": "%cur_art_id%"
%cur_art_tags%	Array of tag IDs that are in use on the currently viewed article	"tags": {"$in": "%cur_art_tags%"}
%cur_art_permalink%	The permalink of the currently viewed article	"url_hash": "%cur_art_permalink%"
%cur_reader_id%	The ID of the currently logged in reader; will filter results by content that reader has access to; will not work for authors who are also readers	"reader_id": "%cur_reader_id%"
%cur_reader_groups%	Array of reader groups IDs that the currently logged in reader is assigned to. Includes an $in comparison so you don't need to add one!	"reader_roles": "%cur_reader_groups%"

Copyright © 2015 – 2025 Silly Moose, LLC. All rights reserved.
Powered by KnowledgeOwl knowledge base software.

Variable Name	Variable Value	Example JSON
%cur_reader_username%	The username of the currently logged in reader	"username": "%cur_reader_username%"
%cur_search_term%	The "phrase" parameter in the URL	"phrase": "%cur_search_term%"

Using the merge code

Now that we have our merge code, let's look at how we can use it within our snippet content to get the information requested. Here's a sample script that console logs the information returned from our API call:

```
<script>
 $(function() {
  $.get('[ko_api(category|{"project_id": "123456", "status": "active", "limit": 5, "sort": {"date_created": 1}})]',
  function(apiData) {
       //do something with the returned data
       console.log(apiData);
  }).fail(function(error) {
    //uh oh something went wrong. Alert the end-user or otherwise handle the error
  });
 });
</script>
```

When this code is rendered to the page, the merge code is replaced with a safe, valid URL and results in something like the following.

```
<script>
 $(function() {
  $.get('/help/ko-api/mid/9999aaaaadsfsdfsdf',
  function(apiData) {
       //do something with the returned data
       console.log(apiData);
  }).fail(function(error) {
    //uh oh something went wrong. Alert the end-user or otherwise handle the error
  });
 });
</script>
```

This example uses some of the knowledge base variables listed above to get all of the other articles that are in the currently viewed article's category:

Copyright © 2015 – 2025 Silly Moose, LLC. All rights reserved.
Powered by KnowledgeOwl knowledge base software.

```
<script>
  $(function() {
    //get all published or needs review articles in the current category except for the one currently being viewed
    $.get('[ko_api(article|{"project_id": "%cur_kb_id%", "status": {"$in": ["published", "review"]}, "category": "%cur_cat_id
%", "url_hash": {"$ne": "%cur_art_permalink%"}, "sort": {"index": 1}})]',
    function(apiData) {
        //do something with the returned data
        console.log(apiData);
    }).fail(function(error) {
      //uh oh something went wrong. Alert the end-user or otherwise handle the error
    });
  });
</script>
```

Working with article status

If you're pulling a list of articles via API snippet, the odds are pretty good that you're going to be using the status field. While most of our other API endpoints have a status field that is "active" or "deleted", the publishing status on articles has two statuses that could be considered active: Published ("published" in the API) and Needs Review ("review" in the API).

If you'd like to filter your article API call to get status, instead of using "status": "active" here, you'd want to use an $in operator and look for the status to be in one of those two: `"status": {"$in": ["published", "review"]}` **, as used in the example above using knowledge base variables.**

API calls with paged results

Sometimes your API call may have multiple pages of results. In this case, we will return the next API call URL as part of the returned data. The URL will be located in the "page_stats" array like so:

```
page_stats: {
  total_records: 203
  total_pages: 3
  next_page: 2
  next_page_url: /help/ko-api/mid/9999aaaaadsfsdfsdf
}
```

Here's a template to get you started with paged API snippet calls:

Copyright © 2015 – 2025 Silly Moose, LLC. All rights reserved.
Powered by KnowledgeOwl knowledge base software.

```
<script>
  $(function(){
    //first page of results API call
    var firstUrl = '[ko_api(article|{"project_id": "%cur_kb_id%","_fields": ["name"], "limit": 75})]';

    //function to get multiple pages of results from API
    var getArticles = function(curUrl) {
      $.get(curUrl, function(data) {
        console.log(data);
        $.each(data['data'], function(index, value){
          //do something with api objects
        });
        //now fetch the next page of results if there is one
        //using the URL returned from the previous API call
        if(data['page_stats']['next_page_url'])
          getArticles(data['page_stats']['next_page_url']);
      }).fail(function(error) {
//you failed!
        console.log(error);
      });
    }

    //get the first page of results;
    getArticles(firstUrl);
  });
</script>
```

Requirements for use

Keep these factors in mind as you use API snippet merge codes:

- API merge codes can only be used for **GET (read) calls**. Attempts to POST, PUT, or DELETE will return an error.
- You must have **at least one active API key in your account with read-only permissions for this knowledge base and for the objects you're querying.** If you don't have an available API key that meets this requirement, the merge code URL returns an error.
- The JSON string containing the API filter must contain **a valid knowledge base ID** in the format of `{"project_id": "1234"}` .
- API calls in snippets won't display in article **Preview** mode. You'll need to publish the article to view the results of the API snippet.



Never use an actual API key
Don't include your API key in the merge code JSON. If you include an API key in the JSON, the merge code URL will return an error.

API guide for AI coding assistants

This page collects the KnowledgeOwl API behavior that isn't obvious from the rest of our API docs. It's written for AI coding assistants such as Claude, ChatGPT, or Copilot, and for developers building integrations.



Using an AI assistant?

Copyright © 2015 – 2025 Silly Moose, LLC. All rights reserved.
Powered by KnowledgeOwl knowledge base software.

Give it this page's URL, or paste the page into its instructions file (for example, `CLAUDE.md`), before it starts working with the API. It will avoid the trial and error below.

Start with these docs

Start with these reference docs:

- Use the KnowledgeOwl API: Authentication, API keys and their permissions, finding your knowledge base ID, query operators, pagination, date formats, and usage limits
- API endpoint reference: Every endpoint and its fields

Every request

Use these guidelines for every request:

- The base URL is `https://app.knowledgeowl.com/api/head/`, followed by the object and format, for example `article.json`. Use HTTPS and basic authentication, with the API key as the username and any dummy password, such as `x`.
- Send `project_id` on every call to a knowledge base resource (articles, categories, files, and so on). Scoped API keys (keys starting with `ko_`) refuse those calls without it, including a GET of a single article by ID. Account-level calls, such as authors and readers, don't need it.
- Check `valid` in the response body, not the HTTP status code. A call the key isn't allowed to make returns HTTP 200 with `{"valid": false, "data": "Not Authorized"}`.
- Results are wrapped in a `data` property. Lists are paginated, with totals in `page_stats`.
- Text that can be translated comes back as an object keyed by language, for example a category name as `{"en": "Getting started"}`. An article's title and body are in `current_version.en.title` and `current_version.en.text`.

Rate limits and caching

Here's some information on how the API limits rates and caches:

- The API rate limit is 1,500 requests per five minutes per account. Over the limit, the API currently returns HTTP 200 with the body `Too many requests.` rather than HTTP 429. Check for that body, then wait and retry.
- GET responses can be cached for up to an hour, keyed on the full query string. To read back something you just wrote, add a throwaway parameter with a unique value, such as `&_cb=1718042331`. The API ignores parameters it doesn't recognize.
- After a write, the reader-facing navigation in your knowledge base can take several minutes to show the change, even when the article page itself is already correct.

Article status values

The API's `status` values are `draft`, `ready`, `rejected`, `published`, `review`, `archived`, and `deleted`. Here's a further breakdown of statuses that may trip you up or cause other issues:

- `review` is Needs Review, and readers can see it, the same as `published`.
- Only `draft`, `ready`, and `rejected` keep an article hidden, so create new articles as `draft`.
- Setting `status` to `deleted` moves an article to Manage > Deleted, where it can be recovered.

Refer to Publishing status for what each status means.

Copyright © 2015 – 2025 Silly Moose, LLC. All rights reserved.
Powered by KnowledgeOwl knowledge base software.

Updating articles safely

Follow these guidelines to update articles safely:

- `PUT /article/{id}.json` is a partial update: send `project_id` and only the fields you're changing, and everything else stays as it was.
- **Array fields are replaced, not merged.** To add a tag, GET the article, add to its `tags` array, and send the whole array back.
- **Sending `current_version` in `PUT /article/{id}.json` overwrites the live article body immediately.** It creates no version and no revision, so there's no undo in the app. Save a copy of the current body first.
- If you do edit the live body by editing `current_version`, send the same title and text to `PUT /articleversion/{current_version_id}.json` as well. Otherwise the active version keeps the old text, and reactivating it later puts the old text back.
- Article bodies are HTML content only. Don't include `<html>`, `<head>`, or `<body>` tags. Line endings are saved as `\n`.

Versions: the safe way to change a published article

1. Create a version with `POST /articleversion.json`, sending `project_id`, `article_id`, `name`, `version_number`, `version_note`, and `current_version: {"en": {"title": ..., "text": ...}}`. The new version is not live yet.
2. Review it in the article editor.
3. Make it live with `PUT /article/{id}.json`, sending `project_id` and `current_version_id`. Refer to Activate a version.



A version's `name` is the article title
It isn't a label for the version. Set `name` to the article's exact current title, and describe the change in `version_note`. Otherwise the article is retitled when the version is activated.

- Minor versions go up by 0.01 (1.00 to 1.01). Read the article's `current_version_number` and calculate the next one yourself.
- The title is stored in three places: the article's `name`, the version's `name`, and `current_version.en.title`, which is what the article page displays. To fix a wrong title, update all three, re-sending the article's current body unchanged with the corrected `current_version.en.title`.

Files

Upload files with `POST /file.json` as `multipart/form-data`, not JSON. Send `project_id`, `name`, and the file itself in a field named `file`. The response includes the file's URL to use in article HTML.

API key permission-endpoint mapping

API key action permissions map to API actions like this:

- Read = GET
- Create = POST
- Update = PUT

Copyright © 2015 – 2025 Silly Moose, LLC. All rights reserved.
Powered by KnowledgeOwl knowledge base software.

- Delete = DELETE

API key resource permissions map to our API endpoints like this:

API key access area	API key resource name	API endpoint
Knowledge base access	Article	`article`
Knowledge base access	Article revision	`articlerevision`
Knowledge base access	Article version	`articleversion`
Knowledge base access	Category	`category`
Knowledge base access	Comment	`comment`
Knowledge base access	File	`file`
Knowledge base access	Glossary term	`glossaryterm`
Knowledge base access	Manage filter	`managefilter`
Knowledge base access	Remote login	`remotelogin`
Knowledge base access	Remote logout	`remotelogout`
Knowledge base access	Snippet	`snippet`
Knowledge base access	Suggest (search)	`suggest`
Knowledge base access	Synonym	`synonym`
Knowledge base access	Tag	`tag`
Account-level access	Author (agent)	`agent`
Account-level access	Author role (userrole)	`userrole`
Account-level access	Author team (userteam)	`userteam`
Account-level access	Notification (webhook)	`webhook`
Account-level access	Reader	`reader`
Account-level access	Reader filter	`readerfilter`
Account-level access	Reader group (readerrole)	`readerroles`

And in the API endpoint documentation, we map the combo of these permissions required for each operation, so for example: `read:author` means the operation requires **Read** permissions on **Authors**.

Upgrade from legacy to scoped API key

In August 2026, we rolled out new, more tightly scoped API keys.

Our older API keys broadly granted access to all account-level and all knowledge bases' resources. All resources could be accessed at the same level based on the actions you selected for the key.

Our new scoped API keys give much more control, allowing you to set:

- The knowledge base(s) the API key should be able to access.
- The resources within that knowledge base the API key should have Read, Create, Update, and Delete permissions on.
- Whether the API key should have access to account-level resources like authors and readers and whether those should be Read, Create, Update, and Delete permissions.

This gives you a lot more control to make sure API keys have only the permission they need and can only access the knowledge bases they need.

Copyright © 2015 – 2025 Silly Moose, LLC. All rights reserved.
Powered by KnowledgeOwl knowledge base software.

The new scoped keys are also more secure, since we only display the key to you once when you first create it.

How to replace a legacy API key

Due to this much more fine-grained security, we encourage you to replace your use of legacy API keys with new scoped API keys. To do so:

1. Go to **Account > API**. The **API keys** page opens.
2. Create a new API key scoped to the knowledge base, resources, and actions you need. Refer to API keys for more detailed instructions.
    a. If you're using a legacy API key with GET-only permissions to support API calls in snippets, create a new scoped API key to this knowledge base only and grant it **Read** permissions for the objects you need. You won't need to add this key anywhere, but it does need to exist.
3. Replace your legacy API key with your new scoped API key(s) in any existing integrations or calls.
4. If you're sure you've updated all integrations, delete the legacy API key. If you're not sure of all the places the key was used, keep the legacy API key active and monitor its **Last accessed** date. If you don't see new accesses in the coming days, weeks, or months, it should be safe to delete.

How to tell the difference between legacy and scoped API keys

Not sure if you're using a legacy API key or a new scoped API key? There are a few telltale signs in **Account > API** to help you figure out which you're using:

- The **Key** displayed: Legacy keys display the full API key. Scoped API keys display only the last 6 characters and obfuscate the rest.
- The **Allowed access** wording: If **Allowed access** only lists API actions like GET, PUT, POST, or DELETE, it's a legacy API key. If **Allowed access** lists the number of KBs and/or the number of resources, it's a scoped API key.
- The **Description**: If the **Description** only includes text or includes nothing at all, it's a legacy API key. If the **Description** includes a summary of KBs and resources the key can access, it's a scoped API key.
- The **Key** itself: Scoped API keys begin with `ko_`. Legacy keys don't.

And last but not least, if you select the gear cog icon to edit a key, if it's a **Legacy** key it only includes checkboxes for allowed methods and a note that it's a legacy key:

Sample **Edit API key** modal for legacy key

Whereas scoped API keys display the full checkbox grid of knowledge bases and resources/actions:

Sample **Edit API key** modal for scoped key

Copyright © 2015 – 2025 Silly Moose, LLC. All rights reserved.
Powered by KnowledgeOwl knowledge base software.

Copyright © 2015 – 2025 Silly Moose, LLC. All rights reserved.
Powered by KnowledgeOwl knowledge base software.