Webhooks

Last Modified on 04/22/2026 2:45 pm EDT

We have pre-built Slack notifications and Email notifications to alert you when certain events occur in your knowledge base, but sometimes those aren't what you need.

Use webhook notifications to generate API object data to build your own alerting integrations with other tools.

When an event occurs that you've created a webhook notification for, we'll POST a message to the endpoint of your choice containing information about the event.

Create a webhook notification

> Only authors with **Full Account Admin** rights can create, edit, and delete notifications.

To create a new webhook notification for your knowledge base:

1. Go to **Account > Notifications**.
2. Select **+ Create notification**. The Create new notification modal opens.
3. Enter a **Name** for this notification. This name is displayed as the title of the card in the **Webhook notifications** display.
4. Select **Webhook** from the **Notification channel** dropdown.
5. Enter a **Webhook endpoint** to send the API payload to.
6. Enter the **Knowledge bases** you'd like to trigger these notifications for.
    a. The field defaults to **All Knowledge Bases**. Once you enter a specific knowledge base, the notifications will only send to the selected knowledge bases.
7. If you've restricted your notifications to a single knowledge base, you can search for a category to **Limit article events to a single category**. This option isn't available if you've selected multiple or all knowledge bases for the notification.
8. Select the **Events** you'd like to receive notifications about from the dropdown. Refer to Available webhook events for more detailed information about each event type and payload.
9. Select **Create notification**.

The page updates to display your **Webhook notifications**,including your newly created notification. As soon as any of your selected notification event(s) occurs, webhook notifications will be sent to the endpoint you provided.

Sample webhook creation ping

As soon as you create your new webhook, we'll send a ping to the endpoint so you can verify it's working. Here's a sample payload:

```
{
 "type": "ping",
 "data": {
  "object": {
   "webhook": "create"
  }
 }
}
```

Edit a webhook notification

> Only authors with **Full Account Admin** rights can create, edit, and delete notifications.

To edit an existing notification:

1. Go to **Account > Notifications**.
2. Select the tab for the type of notification you wish to edit (Email, Slack, or webhook).
3. Hover over the card for the notification you wish to edit.
4. Select the wrench icon that appears in the upper right:
   The **Edit Notification** modal opens.
5. Make the changes you'd like.
6. Select **Update notification** to save your changes.

Reactivate a webhook

If you've been using a webhook notification and the endpoint you set up stops accepting our submissions, we automatically deactivate that webhook after 24 hours of consecutive failed retries.

If we do deactivate a webhook notification, your notification displays a warning that states:

> Webhook deactivated due to 24 consecutive hourly retry failures. Ensure the endpoint returns an HTTP 200 and

click update to reactivate.

Sample Webhook deactivation warning

If you get a report that a webhook-based integration isn't working, head to **Account > Notifications** and open the **Webhooks** tab to check if that webhook has this warning.

Reactivate a deactivated webhook

To reactivate a deactivated webhook:

1. Fix whatever caused the sends to fail for 24 consecutive hours in the first place. Sometimes that means updating the endpoint KnowledgeOwl has for the webhook. Sometimes it means fixing the endpoint that's receiving them.
2. Go to **Account > Notifications**.
3. Open the **Webhooks** tab.
4. If you need to update the endpoint of the webhook settings, select the gear cog icon for that webhook. Refer to Edit a webhook notification for more information.
5. Once you're done making your changes for the webhook, select **Update notification** to save those changes.
6. We'll send a test ping to the webhook and you should receive a message to that effect. As long as you receive that message, your webhook is now reactivated and properly working.

Delete a webhook notification

Only authors with **Full Account Admin** rights can create, edit, and delete notifications.

To delete a notification:

1. Go to **Account > Notifications**.
2. Select the tab for the type of notification you wish to Delete (Email notifications, Slack notifications, or webhooks).
3. Hover over the card for the notification you wish to delete.
4. Select the trash can icon that appears in the upper right:

Select the trash can icon in the notification's card.

The **Delete Notification** modal opens.

5. Select **Delete** to confirm the deletion.

Available webhook events

Below are the knowledge base events you can trigger an email notification on, along with some sample email notifications.

Need something else?
If you have an event that you'd like to receive email notifications on that's not listed below, contact us to request we add it!

Article created

The **Article created** event is triggered when an author selects **Create** for a new article.

Sample Article created webhook payload

The Article created webhook sends type `article.create`, containing `data` with the `article` and the details for the author (`user`) who created it:

```
{
 "sender": {
  "application": "KnowledgeOwl",
  "webhook_id": {{webhook id}},
  "token": {{token}}
 },
 "type": "article.create",
 "created": {{webhook firing UNIX timestamp}}
 "data": {
  "article": {
   "id": {{article id}},
   "name": {{article title}},
   "status": draft,
   "callout_expire": null
  },
  "user": {
   "id": {{author id}},
   "email": {{author email address}},
   "first_name": {{author first name}},
   "last_name": {{author last name}}
  }
 }
}
```

Article updated

The **Article updated** event is triggered whenever an author saves changes to an article.

It does not provide information about what was changed.

Sample Article updated webhook payload

The Article updated webhook sends type `article.update`, containing `data` with the `article`, the details for the author (`user`) who saved the change, and a link to the live article:

```
{
 "sender": {
  "application": "KnowledgeOwl",
  "webhook_id": {{webhook id}},
  "token": {{token}}
 },
 "type": "article.update",
 "created": {{webhook firing UNIX timestamp}},
 "data": {
  "article": {
   "id": {{article id}},
   "name": {{article title}},
   "status": {{article publishing status, e.g. "published"}},
   "callout_expire": {{callout expiration date, UNIX timestamp, if it exists}}
  },
  "user": {
   "id": {{author id}},
   "email": {{author email address}},
   "first_name": {{author first name}},
   "last_name": {{author last name}}
  },
  "live_article_link": {{link to article in live knowledge base}}
 }
}
```

Article updated callout added

The **Article updated callout added** event is triggered when an author saves an article with the Updated callout. It's only triggered if the article previously didn't have an existing New/Updated callout.

Sample Article updated callout added webhook payload

The Article updated callout added webhook sends type `article.updatedCallout`, containing `data` with the `article`, the details for the author (`user`) who added the callout, and a link to the live article:

```
{
 "sender": {
  "application": "KnowledgeOwl",
  "webhook_id": {{webhook id}},
  "token": {{token}}
 },
 "type": "article.updatedCallout",
 "created": {{webhook firing UNIX timestamp}},
 "data": {
  "article": {
   "id": {{article id}},
   "name": {{article title}},
   "status": "published",
   "callout_expire": {{callout expiration date, UNIX timestamp}}
  },
  "user": {
   "id": {{author id}},
   "email": {{author email address}},
   "first_name": {{author first name}},
   "last_name": {{author last name}}
  },
  "live_article_link": {{link to article in live knowledge base}}
 }
}
```

Article new callout added

The Article new callout added event is triggered when an author saves an article with the New callout. It's only triggered if the article previously didn't have an existing New/Updated callout.

Sample Article new callout added webhook payload

The Article new callout added webhook sends type `article.newCallout`, containing `data` with the `article`, the details for the author (`user`) who added the callout, and a link to the live article:

```
{
 "sender": {
  "application": "KnowledgeOwl",
  "webhook_id": {{webhook id}},
  "token": {{token}}
 },
 "type": "article.newCallout",
 "created": {{webhook firing UNIX timestamp}},
 "data": {
  "article": {
   "id": {{article id}},
   "name": {{article title}},
   "status": "published",
   "callout_expire": {{callout expiration date, UNIX timestamp}}
  },
  "user": {
   "id": {{author id}},
   "email": {{author email address}},
   "first_name": {{author first name}},
   "last_name": {{author last name}}
  },
  "live_article_link": {{link to article in live knowledge base}}
 }
}
```

Article published

The Article published event is triggered when when an author publishes a previously unpublished article, either in the article editor or through a bulk edit in Manage.

Sample Article published webhook payload

The Article published webhook sends type `article.publish`, containing `data` with the `article`, the details for the author (`user`) who published the article, and a link to the live article:

```
{
 "sender": {
  "application": "KnowledgeOwl",
  "webhook_id": {{webhook id}},
  "token": {{token}}
 },
 "type": "article.publish",
 "created": {{webhook firing UNIX timestamp}},
 "data": {
  "article": {
   "id": {{article id}},
   "name": {{article title}},
   "status": "published",
   "callout_expire": null
  },
  "user": {
   "id": {{author id}},
   "email": {{author email address}},
   "first_name": {{author first name}},
   "last_name": {{author last name}}
  },
  "live_article_link": {{link to article in live knowledge base}}
 }
}
```

Article archived

The Article archived event is triggered when an author saves an article with the Archived publishing status or archives articles in Manage.

Sample Article archived webhook payload

The Article archived webhook sends type `article.archive`, containing `data` with the `article` and the details for the author (`user`) who archived the article:

```
{
 "sender": {
  "application": "KnowledgeOwl",
  "webhook_id": {{webhook id}},
  "token": {{token}}
 },
 "type": "article.archive",
 "created": {{webhook firing UNIX timestamp}},
 "data": {
  "article": {
   "id": {{article id}},
   "name": {{article title}},
   "status": "archived",
   "callout_expire": null
  },
  "user": {
   "id": {{author id}},
   "email": {{author email address}},
   "first_name": {{author first name}},
   "last_name": {{author last name}}
  }
 }
}
```

Article deleted

The Article deleted event is triggered when an author deletes an article, whether from the Articles page, the article editor, or as a bulk edit in Manage.

Sample Article deleted webhook payload

The Article deleted webhook sends type `article.delete`, containing `data` with the `article` and the details for the author (`user`) who deleted the article:

```
{
 "sender": {
  "application": "KnowledgeOwl",
  "webhook_id": {{webhook id}},
  "token": {{token}}
 },
 "type": "article.delete",
 "created": {{webhook firing UNIX timestamp}},
 "data": {
  "article": {
   "id": {{article id}},
   "name": {{article title}},
   "status": "deleted",
   "callout_expire": null
  },
  "user": {
   "id": {{author id}},
   "email": {{author email address}},
   "first_name": {{author first name}},
   "last_name": {{author last name}}
  }
 }
}
```

Article publishing status changed

The Article publishing status changed event is triggered when when an article's saved with a different publishing status.

This event will also trigger any time the Article published, Article archived, and Article deleted events trigger.

Sample Article publishing status changed webhook payload

The Article publishing status changed webhook sends type `article.statusChange`, containing `data` with the `article`, the details for the author (`user`) who changed the article's status, and a link to the live article:

```
{
 "sender": {
  "application": "KnowledgeOwl",
  "webhook_id": {{webhook id}},
  "token": {{token}}
 },
 "type": "article.statusChange",
 "created": {{webhook firing UNIX timestamp}},
 "data": {
  "article": {
   "id": {{article id}},
   "name": {{article title}},
   "status": {{publishing status}},
   "callout_expire": null,
   "prior_status": {{previous publishing status, e.g. "draft"}},
   "new_status": {{previous publishing status, e.g. "published"}}
  },
  "user": {
   "id": {{author id}},
   "email": {{author email address}},
   "first_name": {{author first name}},
   "last_name": {{author last name}}
  },
  "live_article_link": {{link to article in live knowledge base}}
 }
}
```

Comment created

The Comment created event is triggered when a reader submits a new comment or an author creates a new comment from Reporting > Comments.

Sample Comment created webhook payload

The Comment created webhook sends type `comment.create`, containing `data` for the `comment`, the API link for the comment (`api`), and the details of the article or category the comment was left on:

```
{
 "sender": {
  "application": "KnowledgeOwl",
  "webhook_id": {{webhook id}},
  "token": {{token}}
 },
 "type": "comment.create",
 "created": {{webhook firing UNIX timestamp}},
 "data": {
  "comment": {
   "id": {{comment id}},
   "content": {{full text of comment}},
   "poster": {{Name and email of commenter, e.g. "Linus Owl (linus@knowledgeowl.com)"}},
   "status": "pending",
   "article_id": {{article or category id}}
  },
  "api": "https://app.knowledgeowl.com/api/head/comment/{{comment id}}.json",
  "article_name": {{article or category title}},
  "article_edit_link": {{link to article or category in editor}},
  "article_type": {{"Article" for article; "Category" for category}}
 }
}
```

Comment deleted

The Comment deleted event is triggered when an author deletes a comment.

Sample Comment deleted webhook payload

The Comment deleted webhook sends type `comment.delete`, containing `data` with the `comment`, the API endpoint for the comment (`api`), the details of the article or category the comment was left on, and the details for the author (`user`) who deleted the comment:

```
{
 "sender": {
  "application": "KnowledgeOwl",
  "webhook_id": {{webhook id}},
  "token": {{token}}
 },
 "type": "comment.delete",
 "created": {{webhook firing UNIX timestamp}},
 "data": {
  "comment": {
   "id": {{comment id}},
   "content": {{full text of comment}},
   "poster": {{Name and email of commenter, e.g. "Linus Owl (linus@knowledgeowl.com)"}},
   "status": "deleted",
   "article_id": {{article or category id}}
  },
  "api": "https://app.knowledgeowl.com/api/head/comment/{{comment id}}.json",
  "article_name": {{article or category title}},
  "article_edit_link": {{link to article or category in editor}},
  "article_type": {{"Article" for article; "Category" for category}}
  "user": {
   "id": {{author id}},
   "first_name": {{author first name}},
   "last_name": {{author last name}},
   "email": {{author email address}},
   "icon": {{author icon URL}}
  }
 }
}
```

Comment updated

The Comment updated event is triggered whenever an author edits a comment, such as by approving or deleting it.

Sample Comment updated webhook payload

The Comment updated webhook sends type `comment.update`, containing `data` with the `comment`, the API endpoint for the comment (`api`), the details of the article or category the comment was left on, and the details for the author (`user`) who updated the comment:

```
{
 "sender": {
  "application": "KnowledgeOwl",
  "webhook_id": {{webhook id}},
  "token": {{token}}
 },
 "type": "comment.update",
 "created": {{webhook firing UNIX timestamp}},
 "data": {
  "comment": {
   "id": {{comment id}},
   "content": {{full text of comment}},
   "poster": {{Name and email of commenter, e.g. "Linus Owl (linus@knowledgeowl.com)"}},
   "status": {{comment status: "pending", "deleted", or "approved"}},
   "article_id": {{article or category id}}
  },
  "api": "https://app.knowledgeowl.com/api/head/comment/{{comment id}}.json",
  "article_name": {{article or category title}},
  "article_edit_link": {{link to article or category in editor}},
  "article_type": {{"Article" for article; "Category" for category}}
  "user": {
   "id": {{author id}},
   "first_name": {{author first name}},
   "last_name": {{author last name}},
   "email": {{author email address}},
   "icon": {{author icon URL}}
  }
 }
}
```

Comment status changed

The Comment status changed event is triggered when an author changes a comment's status, for example from Pending to Approved or Deleted.

Sample Comment status changed webhook payload

The Comment status changed webhook sends type `comment.statusChange`, containing `data` with the `comment`, the API endpoint for the comment (`api`), the details of the article or category the comment was left on, and the details for the author (`user`) who changed the comment's status:

```
{
 "sender": {
  "application": "KnowledgeOwl",
  "webhook_id": {{webhook id}},
  "token": {{token}}
 },
 "type": "comment.statusChange",
 "created": {{webhook firing UNIX timestamp}},
 "data": {
  "comment": {
   "id": {{comment id}},
   "content": {{full text of comment}},
   "poster": {{Name and email of commenter, e.g. "Linus Owl (linus@knowledgeowl.com)"}},
   "status": {{comment status: "pending", "deleted", or "approved"}},
   "article_id": {{article or category id}}
   "previous_status": {{previous comment status, e.g. "pending"}},
   "new_status": {{new comment status, e.g. "approved"}}
  },
  "api": "https://app.knowledgeowl.com/api/head/comment/{{comment id}}.json",
  "article_name": {{article or category title}}
  "article_edit_link": {{link to article or category in editor}},
  "article_type": {{"Article" for article; "Category" for category}}
  "user": {
   "id": {{author id}},
   "first_name": {{author first name}},
   "last_name": {{author last name}},
   "email": {{author email address}},
   "icon": {{author icon URL}}
  }
 }
}
```

Contact form submitted

The Contact form submitted event is triggered when a reader submits the Contact Form.

Sample Contact Form submitted webhook payload

The Contact Form submitted webhook sends type `contactForm.submit`, containing `data` with the `ticket_fields` and `submission` details:

```
{
 "sender": {
  "application": "KnowledgeOwl",
  "webhook_id": {{webhook id}},
  "token": {{token}}
 },
 "type": "contactForm.submit",
 "created": {{webhook firing UNIX timestamp}},
 "data": {
  "ticket_fields": {
   "subject": {{contact form submission subject}},
   "content": {{contact form submission body}},
   "from_email": {{submitter's email address, e.g. "linus@knowledgeowl.com"}},
   "from_name": {{submitter's name, e.g. "Linus Owl"}},
   "ip_address": "00.00.00.000",
   "browser": "Chrome",
   "OS": "Windows",
   "ua": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/5
37.36"
  },
  "submission_time": {{contact form submission UNIX timestamp}},
  "submission_status": "success",
  "subject": {{contact form submission subject}},
  "body": {{contact form submission body}},
  "sender_name": {{submitter's name, e.g. "Linus Owl"}},
  "sender_email": {{submitter's email address, e.g. "linus@knowledgeowl.com"}}
 }
}
```

Reader metadata

If you follow the instructions in What data is collected in the Contact Form? to disable storing metadata, the `ticket_fields` won't be included, but the remaining data fields will be.

Webhook API calls

GET LIST

List current webhook subscriptions for your account:

```
curl -u {{API key}}:x https://app.knowledgeowl.com/api/head/webhook.json
```

GET

Pull info for a specific webhook subscription by ID:

```
curl -u {{API key}}:x https://app.knowledgeowl.com/api/head/webhook/{{webhook ID}}.json
```

Example Response

```
{
 "valid": true,
 "data": {
  "id": "12345abcde",
  "type": "webhook",
  "status": "active",
  "output": "default",
  "project_ids": ["all"], //can also be array of Knowledge base IDs
  "event": ["article.create","article.update"],
  "endpoint":"https:\/\/my.webhookconsumer.com\/webhook-receive",
  "token": "98766abadada",
  "date_created": "06\/16\/2017 3:16 pm EDT",
  "date_modified": "06\/16\/2017 3:16 pm EDT",
  "date_deleted": null
 }
}
```


POST

Create a new webhook subscription:

Before you begin
Make sure you have an endpoint that is ready to receive information and will return an HTTP status code of 200.

During the subscription creation, a test call will be made to your endpoint. If the endpoint you specify does not return an HTTP status code of 200, your webhook subscription won't be created.

JSON object sent during test call to provided endpoint when the output is set to "default":

```
{"type":"ping","data":{"object":{"webhook":"create"}}}
```

Single event:

```
curl -u {{API key}}:x -H "Content-Type: application/json" -X POST -d
'{"event": "article.delete", "endpoint": "https://hooks.slack.com/services/abcdefe123/abcd1234", "output": "slack"}'
https://app.knowledgeowl.com/api/head/webhook.json
```

Multiple events:

```
curl -u {{API key}}:x -H "Content-Type: application/json" -X POST -d
'{"event": ["article.delete", "article.create"], "endpoint": "https://my.webhookconsumer.com/webhook-receive"}'
https://app.knowledgeowl.com/api/head/webhook.json
```

PUT

Update a webhook subscription:

```
curl -u {{API key}}:x -H "Content-Type: application/json" -X PUT -d https://app.knowledgeowl.com/api/webhook.json"
'{"event": "article.delete", "endpoint": "https://my.webhookconsumer.com/webhook-receive-new"}'
https://app.knowledgeowl.com/api/head/webhook/{{webhook ID}}.json
```