

Build a Link Preview integration

 developers.notion.com/docs/build-a-link-preview-integration

A Link Preview is a real-time excerpt of authenticated content that unfurls in Notion when an authenticated user pastes a supported link in their workspace.

Developers can build Link Preview integrations to customize how links unfurl in Notion workspaces for domains they own.

An example Link Preview for a GitHub workflow

This guide explains how to use the Notion Link Previews API to create Link Previews for your product. After you've read this guide, you'll know how to:

1. [Configure Link Preview settings in the integration dashboard](#)
2. [Set up the authorization flow](#)
3. [Use the Unfurl Callback URL](#)
4. [Manage updates to Link Previews](#)
5. [Submit a Link Preview integration for security review](#)



To build a Link Preview integration, you must first request access to the Link Preview API. Fill out the [Link Preview request form](#) before proceeding if you haven't already.

Requirements

- You've [requested and received access to the Link Previews API](#).
- You own the domain that you're using to create Link Previews. You'll need to share a verification code from Notion with your domain host when you initialize the integration.
- Your application supports OAuth 2.0.
- You've read the [Link Previews overview](#), so you have a good idea of what you're building and how it works.
- You've read the [Authorization guide](#) and have familiarized yourself with Notion integrations.

With those requirements met, read on!



To build a Link Preview integration, you must first request access to the Link Preview API. Fill out the [Link Preview request form](#) before proceeding if you haven't already.

Create a public Notion integration

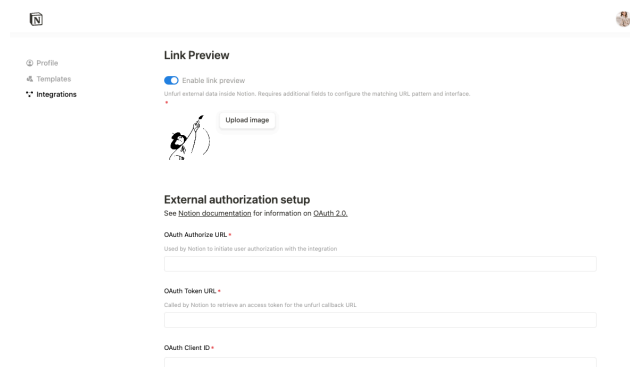
Link Preview integrations are a type of public integration. To create a Link Preview integration, you must first create a public integration. Once the public integration is created, you can enable the link preview setting through the [integration's settings](#).

To learn how to create a public integration, follow the [Authorization guide](#).

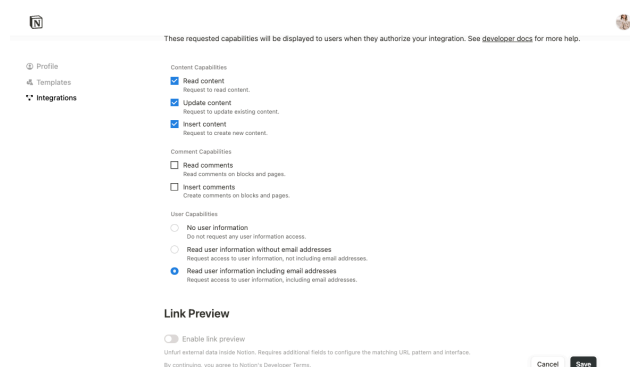
Configure Link Preview settings in the integration dashboard

This step will guide you through enabling link previews for your integration, as well as filling out the link preview integration forms found in your [integration settings](#).

To start, navigate to the public integration you will be using for your link preview integration. It can be found in the [integration dashboard](#). If you have access to the Link Preview API, you will see a Link Preview section in the Configuration tab.



This page contains a toggle input to enable link previews for your integration. Switching it on will display the External Authorization Setup form that will you need to fill out and save.



Once the link preview toggle is turned on for the integration, you will need to fill out two forms in the following order:

1. The External Authorization Setup form.
2. The Unfurling Domain & Patterns form.

In the next section, we'll review how to fill out these forms.

- Make sure that you've [applied](#) and received access to the Link Previews API.
- Confirm that you're logged in to the Notion account that you used to request access.
- Reach out to developers@makenotion.com if you continue to have issues.

1. Fill out the External Authorization Setup form

The External Authorization Setup settings give Notion the information that it needs to let a user authenticate with your service when they paste a Link Preview enabled URL in Notion.

| Field | Description | Example value |
|---------------------|---|---|
| OAuth Authorize URL | <p>The URL that Notion redirects the user to when they connect your integration to their account.</p> <p>When Notion redirects to this URL, it passes a <code>code</code> as a query param in the request. Your integration trades the <code>code</code> for an <code>access_token</code> to make authenticated requests to the Link Previews APIs.</p> | <code>https://<your_domain>.com/notion/authorize</code> |
| OAuth Token URL | <p>The URL that responds to a Notion POST request with an <code>access_token</code> from your service.</p> <p>Notion uses the <code>access_token</code> to make authenticated requests to your systems.</p> | <code>https://<your_domain>.com/notion/token</code> |
| OAuth Client ID | The client ID that Notion uses in its requests to your Authorize and OAuth Token URLs. | <code>mRkZGFjM</code> |

| Field | Description | Example value |
|---------------------------------------|--|---|
| OAuth Client Secret | The client secret that Notion uses in its requests to the Authorize and Token URLs. | ZGVmMjMz |
| OAuth Scopes (optional) | An optional scopes string for Notion to send as a parameter in the request to your OAuth Authorize URL. | unfurl, user_name |
| Deleted Token Callback URL (optional) | <p>A URL that Notion sends a DELETE request to when a user removes a Link Preview from a Notion page or disconnects your integration from their workspace, so that you can delete their tokens.</p> <p>You can use the request body that Notion sends to look up the user and deactivate their associated <code>access_tokens</code> from your service.</p> <p>Whether or not you use a deleted token callback, Notion invalidates any Notion-side tokens corresponding to the user and the Link Preview that they delete.</p> | https://<your_domain>.com/notion/deletion |

After you've filled out this information, click "**Submit ->**" to continue to Unfurling Domain & Patterns.

2. Fill out the Unfurling Domain & Patterns form

The Unfurling Domain & Patterns settings give Notion the information that it needs to recognize the URLs that you want to unfurl Link Previews.

| Field | Description | Example value |
|------------------------------|--|---|
| Unfurl Callback URL | <p>The URL that shares data to be displayed in the Link Preview with Notion. Notion sends POST and DELETE requests to this URL when a user adds or deletes a Link Preview.</p> <p>You can leave this blank as you set up OAuth and return to it once you've created your unfurl attributes. Refer to Step 3 for details.</p> <p>Must be an internet-accessible URL that can and should be protected by authentication.</p> | <code>https://<your_domain>.com/unfurl</code> |
| Unfurl URL Domain | <p>The root domain that maps to this integration.</p> <p>After you add a domain, follow the prompts to verify your domain with Notion.</p> | <code><your_domain>.com</code> |
| URL matching and placeholder | <p>The pattern that a URL must match in order to unfurl as a Link Preview.</p> <p>You can provide multiple patterns for one integration.</p> | Refer to the table below. |

The URL matching and placeholder field includes its own fields:

| Field | Description | Example value |
|-------------|--|---|
| Rule name | A name for the pattern. | <code>"item"</code> |
| Sample URLs | An example URL that matches the pattern and triggers a Link Preview. | <code>https://acme.com/items/23487</code> |

| Field | Description | Example value |
|-------------------------------|---|--|
| Pattern | <p>A Regex pattern that Notion can use to identify URLs that trigger Link Previews.</p> <p>If the Regex pattern fails to match any sample URL, then an error prompt appears when you save settings.</p> | <code>^(?<site>https\:\/\/acme\.com)\/items\/(?<itemNo>\d+)\\$</code> |
| Unfurl Regex Attributes | <p>An array of JSON objects. Each JSON object contains placeholders, populated from Regex capture groups, for a Link Preview's unfurl attributes.</p> <p>Placeholders are displayed when a Link Preview is waiting for data to populate from your service.</p> <p>You can leave this blank as you set up OAuth and return to it once you've created your unfurl attributes.</p> | <pre>[{ "id": "title", "name": "Title", "type": "inline", "inline": { "title": { "value": "Acme Item #\${itemNo}", "section": "title" } } }, { "id": "itemId", "name": "Item Id", "type": "inline", "inline": { "plain_text": { "value": "#\${itemNo}", "section": "identifier" } } }, { "id": "dev", "name": "Developer Name", "type": "inline", "inline": { "plain_text": { "value": "Acme Inc", "section": "secondary" } } }]</pre> |

After you've filled out the External Authorization Setup and Unfurling Domain & Patterns settings, click **Submit ->** to create the integration.

Set up the authorization flow

There are two high-level parts to the auth flow for a Link Preview:

- **Your service authenticates with Notion.** Notion sends a **code** to your OAuth Authorize URL. Your integration exchanges this **code** for a Notion **access_token** that enables your service to make authenticated requests to the Link Previews APIs.

- **Notion authenticates with your service.** Your service responds to Notion's request with a `code`. Notion exchanges this token for your service's `access_token` via your OAuth Token URL. This allows Notion to embed the data from your service in Link Previews.

The tokens **need to be exchanged the first time** a user attempts to add your Link Preview enabled URL to a page. After the initial exchange, the Notion `access_token` is long-living and doesn't need to be updated. If you prefer, Notion can also [support refresh tokens](#) and fetch new tokens from your service.

The auth flow begins when a user shares your Link Preview enabled URL in Notion. Notion recognizes the link and redirects to the OAuth Authorize URL that you provided in the integration settings.

Notion includes the following query params to kick off the OAuth flow with your service:

| Parameter | Description | Value |
|---------------------------|--|---|
| <code>code</code> | A UUID that Notion generates to authenticate with your service. | <code>614846ff-b061-4eac-a511-fc20c3f0838a</code> (example value) |
| <code>redirect_uri</code> | <p>A constant string. Your service redirects a user to this URL after they grant permission for it to access Notion.</p> <p>To prevent attackers from providing arbitrary URIs, your service should validate that the redirect URI matches this value.</p> | <code>https://notion.so/externalIntegrationAuthCallback</code> |

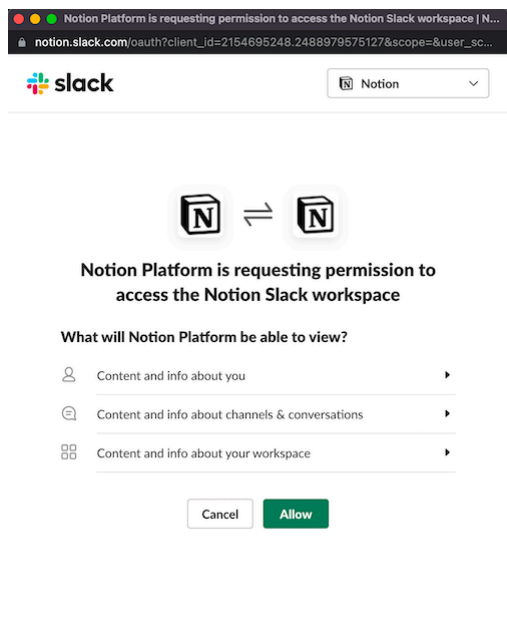
| Parameter | Description | Value |
|-------------------------------|--|---|
| <code>client_id</code> | The OAuth Client ID that you provided when you created the integration. | <code>mRkZGFjM</code> (example value) |
| <code>scope</code> (optional) | The OAuth Scopes value that you provided when you created the integration. | <code>unfurl</code> , <code>user_name</code> (example values) |
| <code>response_type</code> | A constant string. | <code>code</code> |
| <code>state</code> | A randomized string for security validation. | <code>tga@YNV9cfw4yrv0thw</code> (example value) |

Your implementation begins after Notion sends the request.

1. Provide OAuth form

Listen for requests to your OAuth Authorize URL. When you detect a request from Notion, present a UI that asks the user to allow the authentication process to continue.

For example, Slack shares the following interstitial when a user initiates a Link Preview from a Slack URL:



An example interstitial from a Slack Link Preview auth flow

2. Authenticate with Notion's `access_token`

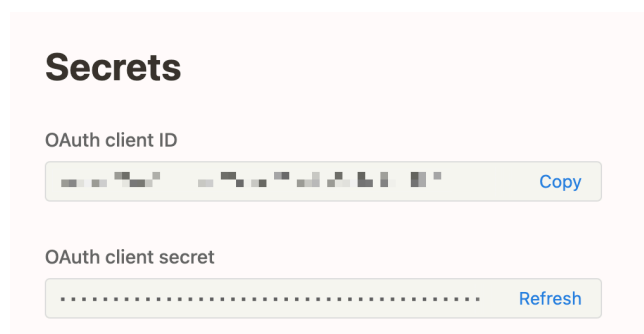
In your integration implementation, retrieve the `code` that Notion sent when it called your OAuth Authorize URL. Then, send the `code` as part of a POST request to Notion's token URL: <https://api.notion.com/v1/oauth/token>.



The Notion `code` is valid for 10 minutes. If the `code` expires, then an error is returned and you need to reinitiate the auth flow for Notion to authenticate with your service ([Step 2a](#)).

To explore other possible errors, refer to the [OAuth 2.0 documentation](#). Default to re-initiating the auth flow to handle errors.

The request is authorized using HTTP Basic Authentication. The credential is a colon-delimited combination of the integration's `CLIENT_ID` and `CLIENT_SECRET`: `CLIENT_ID:CLIENT_SECRET`. You can find these values on the integration settings page. Visit notion.so/profile/integrations, find your integration, and click `View integration`.



You can find your integration's client ID and client secret on the integration settings page



Note that in [HTTP Basic Authentication](#), credentials are `base64` encoded before being added to the `Authorization` header. Notion also requires the word `Basic` before the `base64` encoded string. A complete code param looks something like the following:

```
Basic
NjQ5Mzc0OTIzNzQ5MjM4NDc5MjM4NDc5MjM0NzkyMzc0OjQ3Mzg5Mjc0OTIzODQ3Mjk0OD
cyMzkzNDgyNzk0ODcyMzQ5
```

For more information, read about HTTP Basic Authentication in our [Authorization guide](#).

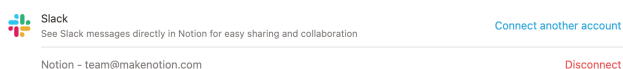
The body of the request contains the following JSON-encoded fields:

| Parameter | Description | Value |
|-------------------------------|--|------------------------------------|
| <code>code</code> | A unique random code that Notion generates to authenticate with your service, generated when a user initiates the auth flow. | <code>"ABC"</code> (example value) |
| <code>grant_type</code> | A constant string. | <code>authorization_code</code> |
| <code>external_account</code> | Object with <code>key</code> and <code>name</code> properties. <code>key</code> should be a unique identifier for the account. Notion uses the <code>key</code> to determine whether or not the user is re-connecting the same account. <code>name</code> should be some way for the user to know which account they used to authenticate with your service. | Refer to the example below. |

The following example demonstrates an `external_account` object for `team@makenotion.com`, a Notion employee account, to authenticate with Slack to use a Slack Link Preview.

```
{
  "key": "A83823453409384",
  "name": "Notion - team@makenotion.com"
}
```

The account `name` appears in the `"My connections"` settings page, where a user can review their authentications for your integration.



An example authenticated Slack connection listed in a user's "My connections" settings page.

A complete POST request looks like the below:

```
curl --location --request POST 'https://api.notion.com/v1/oauth/token' \
--header 'Authorization: Basic "$BASE64_ENCODED_ID_AND_SECRET"' \
--header 'Content-Type: application/json' \
--header 'Notion-Version: 2022-06-28' \
--data '{
  "grant_type": "authorization_code",
  "code": "e202e8c9-0990-40af-855f-ff8f872b1ec6",
  "external_account": {
    "key": "A83823453409384",
    "name": "Notion - team@makenotion.com"
  }
}'
```

Notion responds to the request with a 200 OK and the following response body:

| Parameter | Description | Value |
|-------------------------------------|--|--|
| <code>access_token</code> | A unique random string that Notion generates, in exchange for the <code>code</code> . You can use the <code>access_token</code> to make authorized requests to the Notion API. | "ABC" (example value) |
| <code>bot_id</code> | A UUID representing this authorization. | "3d592781-2dcc-4d4b-bcf3-776a2a7ad7b8" (example value) |
| <code>duplicated_template_id</code> | Always <code>null</code> for Link Preview integrations. Create a standard public integration to use template URLs with the API. | <code>null</code> |
| <code>owner</code> | An object indicating who owns the authorized workspace. | Refer to the bot object documentation. |
| <code>token_type</code> | A constant string. | "bearer" |
| <code>workspace_id</code> | A UUID representing the ID of the Notion workspace where the authorization flow took place. | "3d592781-2dcc-4d4b-bcf3-776a2a7ad7b8" (example value) |
| <code>workspace_icon</code> | A URL to an image, or a string of characters, that identifies the workspace. This could be useful if you'd like to display this authorization in your service's UI. | "🍌" |
| <code>workspace_name</code> | A string representing a human-readable name that can be used to display this authorization in your service's UI. | "My Team Workspace" (example value) |

Store the Notion response, and associate it with the user who initiated the OAuth flow. Notion stores the token that you provided.



For tips on storing `access_tokens`, check out [the auth guide](#).

3. Redirect to the `redirect_uri` with `code`

When a user selects "Allow" to grant Notion the requested permissions, redirect to the `redirect_uri`, the constant string notion.so/externalIntegrationAuthCallback, with your service's unique `code` and the `state` that Notion sent to your service when it initiated the auth flow.

When Notion receives the redirect, it sends a POST request to the OAuth Token URL that you provided with the following body:

| Parameter | Description | Value |
|----------------------------|---|--|
| <code>code</code> | A unique random string that your service generates, retrieved from your service's request to the <code>redirect_uri</code> . Notion is sending back the code that you sent. | <code>WQtaEYNV9jfL4yr89KJA0thw</code> |
| <code>client_id</code> | The OAuth Client ID that you provided when you created the integration. | <code>mRkZGFjM</code> (example value) |
| <code>client_secret</code> | The OAuth Client Secret that you provided when you created the integration. | <code>ZGVmMjMz</code> (example value) |
| <code>redirect_uri</code> | A constant string. | <code>notion.so/externalIntegrationAuthCallback</code> |
| <code>grant_type</code> | A constant string. | <code>authorization_code</code> |

The body is sent in the `application/x-www-form-urlencoded` format and expects a JSON response.

4. Share the `access_token` with Notion

From your OAuth Token URL, respond to Notion's POST request with an `access_token` body parameter in your 200 response.

Notion saves the `access_token` to send in future requests. Notion sends a request to your Unfurl Callback URL every time that the user associated with the token pastes a new Link Preview enabled URL or revisits a page with an existing Link Preview to refresh data.

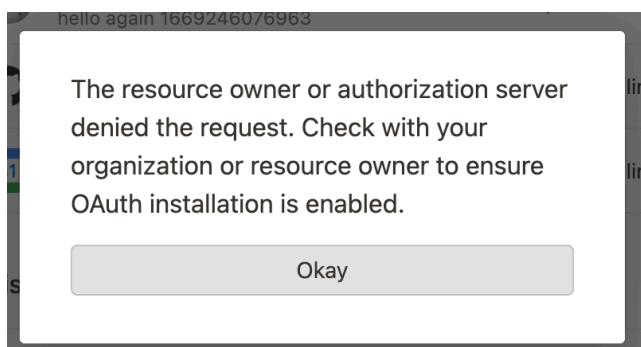
How Notion handles OAuth errors

Notion handles error responses as described by the [OAuth Error Spec](#). The message that Notion displays to the user varies depending on the information that you provide.

For example, you can respond with an `error` and a standard error code like `access_denied`:

`https://notion.so/externalintegrationauthcallback?error=access_denied`

In this instance, Notion shares the following message:

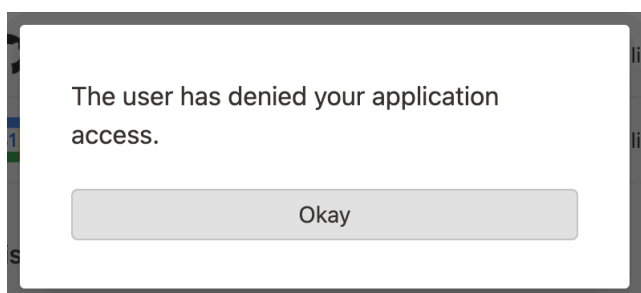


The message that Notion displays in the UI when it receives an `access_denied` error code from your service

You can also add an `error_description` to the response:

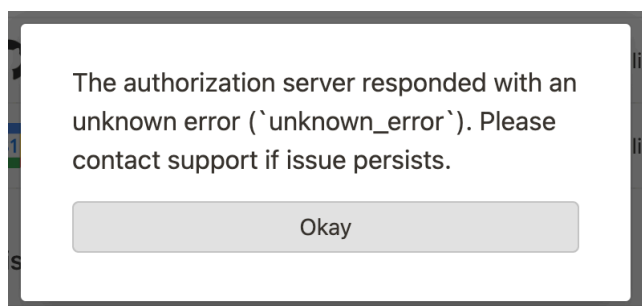
`https://notion.so/externalintegrationauthcallback?error=access_denied&error_description=The+user+has+denied+your+application+access`

If Notion detects a description, then it replaces the standard dialogue prompt with the specified description, as in the below:



If you provide an `error_description` parameter, then the message in the Notion UI displays it

If Notion doesn't recognize the error code, then it notes that the error is unknown:



The message that Notion displays in the UI when it doesn't recognize an error code



For more details on standard error codes, refer to the [OAuth spec](#).

How to use refresh tokens (optional)



Skip to [Step 2e](#) if you're creating long-living access tokens. This section only applies to temporary access tokens.

Return a `refresh_token`

Instead of creating a long-living `access_token`, you can send a `refresh_token` alongside a temporary access token. Notion can then use the `refresh_token` to fetch new tokens from your service.

If you return a `refresh_token`, then you need to also return an `expires_in` integer, as in the following example:

```
{
  "access_token": "ABC",
  "refresh_token": "XYZ",
  "expires_in": 60000
}
```

`expires_in` represents the number of seconds until the `access_token` expires.

Notion requests to refresh a token

If Notion detects that the `access_token` is expired, meaning that the current time exceeds the time of the last refresh plus the `expires_in` value, then Notion refreshes the tokens when it calls your endpoints.

To refresh the token, Notion sends a POST request to your OAuth Token URL with the following parameters:

| Parameter | Description | Value |
|----------------------------|---|--|
| <code>refresh_token</code> | The unique random string returned in the response to the previous request. | <code>"ABC"</code> (example value) |
| <code>client_id</code> | The OAuth Client ID that you provided when you created the integration. | <code>mRkZGFjM</code> (example value) |
| <code>client_secret</code> | The OAuth Client Secret that you provided when you created the integration. | <code>ZGVmMjMz</code> (example value) |
| <code>redirect_uri</code> | A constant string. | <code>https://notion.so/externalIntegrationAuthCallback</code> |
| <code>grant_type</code> | A constant string. | <code>refresh_token</code> |

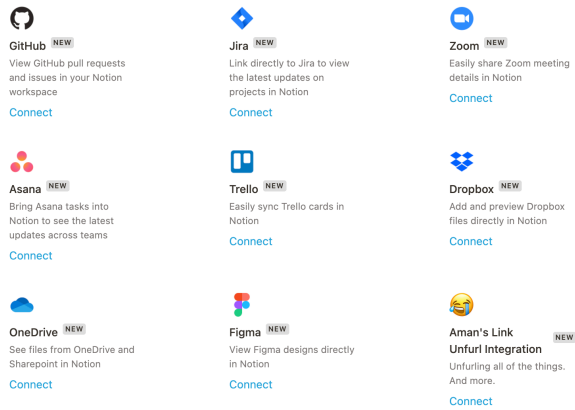
Respond to Notion's request to refresh a token

From your OAuth Token URL, return an `access_token` in your 200 response. You can optionally return new `refresh_token` and `expires_in` values.

5. Test the auth flow in Notion

To test the auth flow, make sure that you've added the integration to a workspace. Then, navigate to `"My connections"` in the workspace settings. Click `"Show all"`. Find the integration that you created in the list, and select `"Connect"` to kick off the auth flow.

Discover new apps





Guess which icon represents our test integration



If you don't see your integration in the list, then refresh the page. Notion only loads new integrations on page load.

If the auth flow is successful, then you'll see a new entry under the **"My connections"** menu.

| My connections | | |
|--|---|-----|
| Connection | Access | |
|  Slackbot Local Local AI Slackbot - ryan@makenotion.com | Can preview links Can view, insert, and update content | ... |
|  Wiki hello again 1669157637703 | Can preview links | ... |

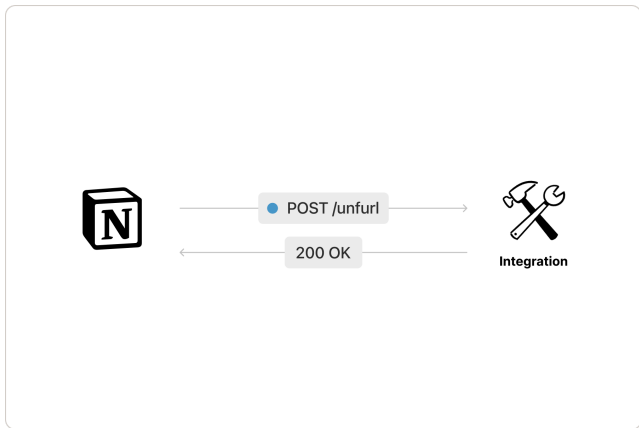
The new entry that appeared when we connected a much happier test Link Preview integration

To verify that the **key** for this connection is unique, repeat the auth flow multiple times using the same credentials to validate that you only get a single entry.

📞 Use the Unfurl Callback URL

1. Configure unfurl attributes

After a user pastes a Link Preview enabled link and completes the auth flow, Notion sends a POST request to the Unfurl Callback URL that you provided in the integration settings.



Notion sends the `access_token` from your service in a POST to the Unfurl Callback URL that you provide

The request includes a Bearer authorization with the user's `access_token` from your service, and the payload is a single field called `uri` that includes the link that the user shared:

```
curl -d '{"uri":"http://example.com/file/123"}' \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer <ACCESS_TOKEN>" \
  -X POST https://example.com/unfurl
```

Set up the Unfurl Callback URL to respond to Notion's request with a 200 OK including the `uri`, and an array of all of the unfurl attributes, the values to display in the Link Preview. **The array must include a `title` attribute that gives the Link Preview a title and a `dev` attribute that indicates the developer or company that created the Link Preview.**

The following is an example response:

```

{
  "uri": "http://example.com/file/123",
  "operations": [
    {
      "path": [
        "attributes"
      ],
      "set": [
        {
          "id": "title",
          "name": "Title",
          "type": "inline",
          "inline": {
            "title": {
              "value": "The Link Preview's Title",
              "section": "title"
            }
          }
        }
      ],
      {
        "id": "dev",
        "name": "Developer Name",
        "type": "inline",
        "inline": {
          "plain_text": {
            "value": "Acme Inc",
            "section": "secondary"
          }
        }
      },
      {
        "id": "color",
        "name": "Color",
        "type": "inline",
        "inline": {
          "color": {
            "value": {
              "r": 235,
              "g": 64,
              "b": 52
            },
            "section": "background"
          }
        }
      }
    ]
  ]
}

```

To preview how different response objects unfurl in a Link Preview, explore the integration's Link Preview Lab.



To learn more about unfurl attributes, refer to the Link Preview [unfurl attributes reference](#).

2. Handle unfurl request errors

Set up the Unfurl Callback URL to handle errors, as in the following example.

```
{
  "uri": "http://example.com/file/123",
  "operations": [
    {
      "path": ["error"],
      "set": { "status": 404, "message": "Content not found" }
    }
  ]
}
```

Manage updates to Link Previews

Update Link Previews to reflect data shared in unfurl attributes

If the unfurl attributes from your service change over time, then you can alert Notion to update the Link Preview to mirror those changes.

When your service detects changes to data that is referenced by a Link Preview, send a PATCH request to Notion's [/v1/external/object](#) endpoint to update the unfurl attributes.

To update the unfurl attributes displayed in a Link Preview, send a PATCH request to Notion's [/v1/external/object](#) endpoint using all of the attributes from the original Unfurl Callback URL response

Include all of the same objects from the Unfurl Callback URL response in the request, including the attributes that haven't changed, as in the following example:

```

curl -d `{
  "uri": "http://example.com/file/123",
  "operations": [
    {
      "path": ["attributes"],
      "set": [
        {
          "id": "title",
          "name": "Title",
          "type": "inline",
          "inline": {
            "title": {
              "value": "The Link Preview's NEW Title",
              "section": "title"
            }
          }
        },
        {
          "id": "color",
          "name": "Color",
          "type": "inline",
          "inline": {
            "color": {
              "value": {
                "r": 235,
                "g": 64,
                "b": 52
              }
            }
          }
        }
      ]
    }
  ]
} \
-H "Content-Type: application/json" \
-H "Authorization: Bearer <ACCESS_TOKEN>" \
-H "Notion-Version: 2021-08-16" \
-X PATCH https://api.notion.com/v1/external/object

```

It's also possible to set a new error request. For example, if the data originally shared in a Link Preview can't be found, then you could send an update request as follows:

```

curl -d `{
  "uri": "http://example.com/file/123",
  "operations": [
    {
      "path": ["error"],
      "set": { "status": 404, "message": "Content not found" }
    }
  ]
}` \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer <ACCESS_TOKEN>" \
  -H "Notion-Version: 2021-08-16" \
  -X PATCH https://api.notion.com/v1/external/object

```

You can also set both new attributes and an error request at the same time, as in the below example:

```

curl -d `{
  "uri": "http://example.com/file/123",
  "operations": [
    {
      "path": ["attributes"],
      "set": [
        {
          "id": "title",
          "name": "Title",
          "type": "inline",
          "inline": {
            "title": {
              "value": "The Link Preview's Title",
              "section": "title"
            }
          }
        },
        {
          "id": "color",
          "name": "Color",
          "type": "inline",
          "inline": {
            "color": {
              "value": {
                "r": 235,
                "g": 64,
                "b": 52
              }
            }
          }
        }
      ],
      "section": "background"
    }
  ],
  "path": ["error"],
  "set": { "status": 404, "message": "Content not found" }
} ` \
-H "Content-Type: application/json" \
-H "Authorization: Bearer <ACCESS_TOKEN>" \
-H "Notion-Version: 2021-08-16" \
-X PATCH https://api.notion.com/v1/external/object

```



When updating a Link Preview's unfurl attributes, there's no need to clear the **error**. If no **error** is sent, then the **error** is automatically cleared.

Notion updates your service when a user deletes Link Preview enabled URLs

When a user deletes all Link Previews associated with a URL from their workspace, Notion sends a DELETE request to your Unfurl Callback URL.

Notion sends a DELETE request to your `/unfur1` endpoint.

Listen for the request to perform any associated actions, like deleting the record from your service.

Submit your integration for security review

Before a Link Preview integration can be publicly distributed, it needs to pass a security review. [Fill out this form](#) to submit your integration for review.

Next steps

To learn more about customizing a Link Preview's unfurl attributes, refer to the [reference docs](#)

Updated over 1 year ago

Did this page help you?