Restrict public access to your content

March 15, 2022 • Martina Farkasova and Jan Cerman • 5 min read • TypeScript

Protect your content and assets with secure access. You might want to use this with sensitive content, content hidden behind sign-in walls, or for projects that are not public facing.

By default, your project's assets and published content items are publicly available.

Enable secure access

Using secure access means you'll need to provide a Delivery API key with each request to the Delivery API. This applies for both Delivery REST API and Delivery GraphQL API. The Delivery API allows you to use two concurrent API keys, Primary and Secondary.

> **Primary vs. Secondary key**
>
> The following instructions work equally for both the Primary key and the Secondary key. Both API keys are generated per project and have no expiration date. For continuous use, we recommend using the Primary key. Use the Secondary key only when revoking the Primary key to prevent downtime.

By activating secure access, Kontent will generate two API keys.

1. In Kontent, go to **Project settings** > **API keys**.
2. In the **Secure access** box:
   1. Toggle the **Inactive** switch to activate secure access.
   2. Click for one of the API keys.

You can now use the keys to authenticate your API requests.

Authenticate requests

Every request you make must be sent with an API key in the header.
The header uses the following format: .

We recommend you use the Primary key. The Secondary key is handy in case you need to revoke and rotate the keys.

Retrieve content items securely

Tips for staying secure

Here are a few tips to help you when using Delivery API with secure access enabled:

- Retrieve content on the server side and NOT client side to prevent leaking your API keys.
- Store your API Keys outside your source code. For example, store them as environment variables. Make sure they're encrypted too.
- Regenerate only one key at a time to prevent downtime.
- Regenerate your API keys periodically. The older a key is, the higher the probability it could have been compromised.

When getting content items, add an API key on top of your requests. For example, the code below shows how to securely retrieve content of an article named *My article*.

TypeScript

After performing the request, you receive a single content item in the JSON format. You can filter your requests to, for example, retrieve only specific elements or items.

Retrieve assets securely

With advanced asset management, you can restrict access to all your project's assets by requiring an API key. This API key is different from the API keys in ⚙ **Project settings** > **API keys**.

To set up secure access for assets, contact our support and let them know the following:

- Your project ID
- Whether secure assets should be enabled for the Delivery Preview API, Delivery API, or both

Once you enable secure assets, you'll need to provide an API to fetch every asset. Fetch assets on the server side of your app to prevent exposing the API key.

Revoke the API keys

When you suspect unauthorized key usage, you need to revoke one of the API keys and generate a new one. For example, when a developer with access to the API keys had left your company. In such cases, one or both API keys can be regenerated.

Generating a new key will replace the old key. This process can take up to a couple of minutes. Any requests made with a revoked API key receive the 401 Unauthorized error.

1. In Kontent, go to ⚙ **Project settings** > **API keys**.
2. In the **Delivery API** box, regenerate the Secondary key to ensure it's new and secure.
3. Update your applications using secure access to use the newly generated Secondary key.
4. Validate that your applications work correctly with the new key.
5. Regenerate the Primary key to make sure unauthorized users cannot use this key to access your content.
6. (Optional) Switch back to using the regenerated Primary key in all your applications.

The last step is optional because switching back to the regenerated Primary key might seem unnecessarily complicated. The reason behind this is simple – you can easily keep track of the API key you are currently using for your application. If you only use the Secondary key to prevent downtime when revoking the Primary key, it can keep things simple overall.