Migrating GraphQL global node IDs

In this article

Learn about the two global node ID formats and how to migrate from the legacy format to the new format.

Background

The GitHub Enterprise Cloud GraphQL API currently supports two types of global node ID formats. The legacy format will be deprecated and replaced with a new format. This guide shows you how to migrate to the new format, if necessary.

By migrating to the new format, you ensure that the response times of your requests remain consistent and small. You also ensure that your application continues to work once the legacy IDs are fully deprecated.

To learn more about why the legacy global node ID format will be deprecated, see "New global ID format coming to GraphQL."

Determining if you need to take action

You only need to follow the migration steps if you store references to GraphQL global node IDs. These IDs correspond to the `id` field for any object in the schema. If you don't store any global node IDs, then you can continue to interact with the API with no change.

Additionally, if you currently decode the legacy IDs to extract type information (for example, if you use the first two characters of `PR_kwDOAHz1OX4uYAah` to determine if the object is a pull request), your service will break since the format of the IDs has changed. You should migrate your service to treat these IDs as opaque strings. These IDs will be unique, therefore you can rely on them directly as references.

Migrating to the new global IDs

To facilitate migration to the new ID format, you can use the `X-Github-Next-Global-ID` header in your GraphQL API requests. The value of the `X-Github-Next-Global-ID` header can be `1` or `0`. Setting the value to `1` will force the response payload to always use the new ID format for any object that you requested the `id` field for. Setting the value to `0` will revert to default behavior, which is to show the legacy ID or new ID depending on the object creation date.

Here is an example request using a `curl` command:

```
$ curl \
```

```
  -H "Authorization: Bearer $GITHUB_TOKEN" \
  -H "X-Github-Next-Global-ID: 1" \
  https://api.github.com/graphql \
  -d '{ "query": "{ node(id: \"MDQ6VXNlcjM0MDczMDM=\") { id } }" }'
```

Even though the legacy ID `MDQ6VXNlcjM0MDczMDM=` was used in the query, the response will contain the new ID format:

```
{"data":{"node":{"id":"U_kgDOADP9xw"}}}
```

With the `X-Github-Next-Global-ID` header, you can find the new ID format for legacy IDs that you reference in your application. You can then update those references with the ID received in the response. You should update all references to legacy IDs and use the new ID format for any subsequent requests to the API. To perform bulk operations, you can use aliases to submit multiple node queries in one API call. For more information, see "the GraphQL docs."

You can also get the new ID for a collection of items. For example, if you wanted to get the new ID for the last 10 repositories in your organization, you could use a query like this:

```
{
  organization(login: "github") {
    repositories(last: 10) {
      edges {
        cursor
        node {
          name
          id
        }
      }
    }
  }
}
```

Note that setting `X-Github-Next-Global-ID` to `1` will affect the return value of every `id` field in your query. This means that even when you submit a non-`node` query, you will get back the new format ID if you requested the `id` field.

Sharing feedback

If you have any concerns about the rollout of this change impacting your app, please contact us through the GitHub Support portal and include information such as your app name so that we can better assist you.