Registering a GitHub App from a manifest

In this article

A GitHub App manifest is a way to share a preconfigured GitHub App registration with other users. The manifest flow allows someone to quickly register a GitHub App.

About GitHub App Manifests

When someone registers a GitHub App from a manifest, they only need to follow a URL and name the app. The manifest includes the permissions, events, and webhook URL needed to automatically register the app. The manifest flow creates the GitHub App registration and generates the app's webhook secret, private key (PEM file), client secret, and GitHub App ID. The person who creates the GitHub App registration from the manifest will own the GitHub App registration and can choose to edit the registration's settings, delete it, or transfer it to another person on GitHub.

You can use Probot to get started with GitHub App Manifests or see an example implementation. See "Using Probot to implement the GitHub App Manifest flow" to learn more.

Here are some scenarios where you might use GitHub App Manifests to register pre-configured apps:

- Help new team members come up-to-speed quickly when developing GitHub Apps.
- Allow others to extend a GitHub App using the GitHub APIs without requiring them to configure an app.
- Create GitHub App reference designs to share with the GitHub community.
- Ensure you deploy GitHub Apps to development and production environments using the same configuration.
- Track revisions to a GitHub App configuration.

Implementing the GitHub App Manifest flow

The GitHub App Manifest flow uses a handshaking process similar to the OAuth flow. The flow uses a manifest to register a GitHub App and receives a temporary `code` used to retrieve the app's private key, webhook secret, and ID.

> **Note:** You must complete all three steps in the GitHub App Manifest flow within one hour.

Follow these steps to implement the GitHub App Manifest flow:

1. You redirect people to GitHub to register a new GitHub App.
2. GitHub redirects people back to your site.
3. You exchange the temporary code to retrieve the app configuration.

1. You redirect people to GitHub to register a new GitHub App

To redirect people to register a new GitHub App, provide a link for them to click that sends a `POST` request to `https://github.com/settings/apps/new` for a personal account or `https://github.com/organizations/ORGANIZATION/settings/apps/new` for an organization account, replacing `ORGANIZATION` with the name of the organization account where the app will be registered.

You must include the GitHub App Manifest parameters as a JSON-encoded string in a parameter called `manifest`. You can also include a `state` parameter for additional security.

The person registering the app will be redirected to a GitHub page with an input field where they can edit the name of the app you included in the `manifest` parameter. If you do not include a `name` in the `manifest`, they can set their own name for the app in this field.

GitHub App Manifest parameters

Name	Type	Description
`name`	`string`	The name of the GitHub App.
`url`	`string`	**Required.** The homepage of your GitHub App.
`hook_attributes`	`object`	The configuration of the GitHub App's webhook.
`redirect_url`	`string`	The full URL to redirect to after a user initiates the registration of a GitHub App from a manifest.
`callback_urls`	`array of strings`	A full URL to redirect to after someone authorizes an installation. You can provide up to 10 callback URLs.
`setup_url`	`string`	A full URL to redirect users to after they install your GitHub App if additional setup is required.
`description`	`string`	A description of the GitHub App.
`public`	`boolean`	Set to `true` when your GitHub App is available to the public or `false` when it is only accessible to the owner of the app.
`default_events`	`array`	The list of events the GitHub

~~default_events~~	~~array~~	The list of events the GitHub App subscribes to.
`default_permissions`	`object`	The set of permissions needed by the GitHub App. The format of the object uses the permission name for the key (for example, `issues`) and the access type for the value (for example, `write`).
`request_oauth_on_install`	`boolean`	Set to `true` to request the user to authorize the GitHub App, after the GitHub App is installed.
`setup_on_update`	`boolean`	Set to `true` to redirect users to the `setup_url` after they update your GitHub App installation.

The `hook_attributes` object has the following keys.

Name	Type	Description
`url`	`string`	**Required.** The URL of the server that will receive the webhook `POST` requests.
`active`	`boolean`	Deliver event details when this hook is triggered, defaults to true.

Parameters

Name	Type	Description
`state`	`string`	An unguessable random string. It is used to protect against cross-site request forgery attacks.

Examples

This example uses a form on a web page with a button that triggers the `POST` request for a personal account:

```
<form action="https://github.com/settings/apps/new?state=abc123" method="post">
 Register a GitHub App Manifest: <input type="text" name="manifest"
id="manifest"><br>
 <input type="submit" value="Submit">
</form>

<script>
 input = document.getElementById("manifest")
 input.value = JSON.stringify({
   "name": "Octoapp",
   "url": "https://www.example.com",
   "hook_attributes": {
     "url": "https://example.com/github/events",
   },
   "redirect_url": "https://example.com/redirect",
   "callback_urls": [
```

```
      "https://example.com/callback"
    ],
    "public": true,
    "default_permissions": {
      "issues": "write",
      "checks": "write"
    },
    "default_events": [
      "issues",
      "issue_comment",
      "check_suite",
      "check_run"
    ]
  })
</script>
```

This example uses a form on a web page with a button that triggers the `POST` request for an organization account. Replace `ORGANIZATION` with the name of the organization account where you want to register the app.

```
<form action="https://github.com/organizations/ORGANIZATION/settings/apps/new?
state=abc123" method="post">
 register a GitHub App Manifest: <input type="text" name="manifest"
id="manifest"><br>
 <input type="submit" value="Submit">
</form>

<script>
 input = document.getElementById("manifest")
 input.value = JSON.stringify({
   "name": "Octoapp",
   "url": "https://www.example.com",
   "hook_attributes": {
     "url": "https://example.com/github/events",
   },
   "redirect_url": "https://example.com/redirect",
   "callback_urls": [
     "https://example.com/callback"
   ],
   "public": true,
   "default_permissions": {
     "issues": "write",
     "checks": "write"
   },
   "default_events": [
     "issues",
     "issue_comment",
     "check_suite",
     "check_run"
   ]
 })
</script>
```

2. GitHub redirects people back to your site

When the person clicks **Create GitHub App**, GitHub redirects back to the `redirect_url` with a temporary `code` in a code parameter. For example:

```
https://example.com/redirect?code=a180b1a3d263c81bc6441d7b990bae27d4c10679
```

If you provided a `state` parameter, you will also see that parameter in the `redirect_url`. For example:

```
https://example.com/redirect?
```

```
code=a180b1a3d263c81bc6441d7b990bae27d4c10679&state=abc123
```

3. You exchange the temporary code to retrieve the app configuration

To complete the handshake, send the temporary `code` in a `POST` request to the Create a GitHub App from a manifest endpoint. The response will include the `id` (GitHub App ID), `pem` (private key), and `webhook_secret`. GitHub creates a webhook secret for the app automatically. You can store these values in environment variables on the app's server. For example, if your app uses dotenv to store environment variables, you would store the variables in your app's `.env` file.

You must complete this step of the GitHub App Manifest flow within one hour.

> **Note:** This endpoint is rate limited. See Rate limits to learn how to get your current rate limit status.

```
POST /app-manifests/{code}/conversions
```

For more information about the endpoint's response, see Create a GitHub App from a manifest.

When the final step in the manifest flow is completed, the person registering the app from the flow will be an owner of a registered GitHub App that they can install on any of their personal repositories. They can choose to extend the app using the GitHub APIs, transfer ownership to someone else, or delete it at any time.

Using Probot to implement the GitHub App Manifest flow

Probot is a framework built with Node.js that performs many of the tasks needed by all GitHub Apps, like validating webhooks and performing authentication. Probot implements the GitHub App manifest flow, making it easy to create and share GitHub App reference designs with the GitHub community.

To create a Probot App that you can share, follow these steps:

1. Generate a new GitHub App.
2. Open the project you created, and customize the settings in the `app.yml` file. Probot uses the settings in `app.yml` as the GitHub App Manifest parameters.
3. Add your application's custom code.
4. Run the GitHub App locally or host it anywhere you'd like. When you navigate to the hosted app's URL, you'll find a web page with a **Register GitHub App** button that people can click to register a preconfigured app.

Using dotenv, Probot creates a `.env` file and sets the `APP_ID`, `PRIVATE_KEY`, and `WEBHOOK_SECRET` environment variables with the values retrieved from the app configuration.

Hosting your app with Glitch

You can see an example Probot app that uses Glitch to host and share the app. The example uses the Checks API and selects the necessary Checks API events and

permissions in the `app.yml` file. Glitch is a tool that allows you to "Remix your own" apps. Remixing an app creates a copy of the app that Glitch hosts and deploys. See "About Glitch" to learn about remixing Glitch apps.