Generating a JSON Web Token (JWT) for a GitHub App

In this article

Learn how to create a JSON Web Token (JWT) to authenticate to certain REST API endpoints with your GitHub App.

About JSON Web Tokens (JWTs)

In order to authenticate as an app or generate an installation access token, you must generate a JSON Web Token (JWT). If a REST API endpoint requires a JWT, the documentation for that endpoint will indicate that you must use a JWT to access the endpoint.

Your JWT must be signed using the `RS256` algorithm and must contain the following claims.

Claim	Meaning	Details
`iat`	Issued At	The time that the JWT was created. To protect against clock drift, we recommend that you set this 60 seconds in the past and ensure that your server's date and time is set accurately (for example, by using the Network Time Protocol).
`exp`	Expires At	The expiration time of the JWT, after which it can't be used to request an installation token. The time must be no more than 10 minutes into the future.
`iss`	Issuer	The ID of your GitHub App. This value is used to find the right public key to verify the signature of the JWT. You can find your app's ID with the `GET /app` REST API endpoint. For more information, see "Apps" in the REST API documentation.
`alg`	Message authentication code algorithm	This should be `RS256` since your JWT must be signed using the `RS256` algorithm.

To use a JWT, pass it in the `Authorization` header of an API request. For example:

```
curl --request GET \
--url "https://api.github.com/app" \
--header "Accept: application/vnd.github+json" \
--header "Authorization: Bearer YOUR_JWT" \
--header "X-GitHub-Api-Version: 2022-11-28"
```

In most cases, you can use `Authorization: Bearer` or `Authorization: token` to pass a token. However, if you are passing a JSON web token (JWT), you must use `Authorization: Bearer`.

Generating a JSON Web Token (JWT)

Most programming languages have a package that can generate a JWT. In all cases, you must have a private key and the ID of your GitHub App. For more information about generating a private key, see "Managing private keys for GitHub Apps". You can find your app's ID with the `GET /app` REST API endpoint. For more information, see "Apps" in the REST API documentation.

> Note: Instead of creating a JWT, you can use GitHub's Octokit SDKs to authenticate as an app. The SDK will take care of generating a JWT for you and will regenerate the JWT once the token expires. For more information, see "Scripting with the REST API and JavaScript."

Example: Using Ruby to generate a JWT

> **Note:** You must run `gem install jwt` to install the `jwt` package in order to use this script.

In the following example, replace `YOUR_PATH_TO_PEM` with the file path where your private key is stored. Replace `YOUR_APP_ID` with the ID of your app. Make sure to enclose the values for `YOUR_PATH_TO_PEM` and `YOUR_APP_ID` in double quotes.

```ruby
require 'openssl'
require 'jwt'  # https://rubygems.org/gems/jwt

# Private key contents
private_pem = File.read("YOUR_PATH_TO_PEM")
private_key = OpenSSL::PKey::RSA.new(private_pem)

# Generate the JWT
payload = {
  # issued at time, 60 seconds in the past to allow for clock drift
  iat: Time.now.to_i - 60,
  # JWT expiration time (10 minute maximum)
  exp: Time.now.to_i + (10 * 60),
  # GitHub App's identifier
  iss: "YOUR_APP_ID"
}

jwt = JWT.encode(payload, private_key, "RS256")
puts jwt
```

Example: Using Python to generate a JWT

> **Note:** You must run `pip install jwt` to install the `jwt` package in order to use this script.

Python

```python
#!/usr/bin/env python3
import jwt
import time
import sys

# Get PEM file path
if len(sys.argv) > 1:
    pem = sys.argv[1]
else:
    pem = input("Enter path of private PEM file: ")

# Get the App ID
if len(sys.argv) > 2:
    app_id = sys.argv[2]
else:
    app_id = input("Enter your APP ID: ")

# Open PEM
with open(pem, 'rb') as pem_file:
    signing_key = jwt.jwk_from_pem(pem_file.read())

payload = {
    # Issued at time
    'iat': int(time.time()),
    # JWT expiration time (10 minutes maximum)
    'exp': int(time.time()) + 600,
    # GitHub App's identifier
    'iss': app_id
}

# Create JWT
jwt_instance = jwt.JWT()
encoded_jwt = jwt_instance.encode(payload, signing_key, alg='RS256')

print(f"JWT:  {encoded_jwt}")
```

This script will prompt you for the file path where your private key is stored and for the ID of your app. Alternatively, you can pass those values as inline arguments when you execute the script.