

# Package ‘tinyoauth’

June 24, 2026

**Type** Package

**Title** Minimal OAuth 2.0 Client

**Version** 0.1.1

**Date** 2026-06-22

**Description** A dependency-light OAuth 2.0

<<https://www.rfc-editor.org/rfc/rfc6749>> client supporting the client-credentials and authorization-code grants with token refresh.

Built on 'curl' and 'jsonlite', with base R's socket server for the redirect listener, avoiding heavier HTTP stacks.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Depends** R (>= 4.0)

**Imports** curl, digest, jsonlite

**Suggests** tinytest

**URL** <https://github.com/cornball-ai/tinyoauth>

**BugReports** <https://github.com/cornball-ai/tinyoauth/issues>

**NeedsCompilation** no

**Author** Troy Hernandez [aut, cre] (ORCID:

<<https://orcid.org/0009-0005-4248-604X>>),

cornball.ai [cph]

**Maintainer** Troy Hernandez <troy@cornball.ai>

**Repository** CRAN

**Date/Publication** 2026-06-24 07:40:07 UTC

## Contents

|                                   |   |
|-----------------------------------|---|
| anthropic_claude_client . . . . . | 2 |
| oauth_authorize_url . . . . .     | 3 |
| oauth_bearer . . . . .            | 3 |
| oauth_cache_path . . . . .        | 4 |

|                                    |           |
|------------------------------------|-----------|
| oauth_client . . . . .             | 4         |
| oauth_exchange_code . . . . .      | 5         |
| oauth_expired . . . . .            | 5         |
| oauth_import_httr . . . . .        | 6         |
| oauth_jwt_payload . . . . .        | 7         |
| oauth_refresh . . . . .            | 7         |
| oauth_request . . . . .            | 8         |
| oauth_token . . . . .              | 9         |
| oauth_token_anthropic . . . . .    | 9         |
| oauth_token_authcode . . . . .     | 10        |
| oauth_token_client . . . . .       | 11        |
| oauth_token_openai_codex . . . . . | 12        |
| openai_codex_account_id . . . . .  | 13        |
| openai_codex_client . . . . .      | 13        |
| <b>Index</b>                       | <b>14</b> |

---

anthropic\_claude\_client

*OAuth client for the Anthropic Claude (Claude Code) login flow*

---

### Description

A preconfigured [oauth\_client] for Claude-subscription-backed access, carrying Anthropic's authorize and token endpoints plus the Claude Code scope string. The client id is Anthropic's public Claude Code identifier, not a secret.

### Usage

```
anthropic_claude_client()
```

### Value

A `tinyoauth_client` with an extra scope field.

### Examples

```
anthropic_claude_client()
```

---

oauth\_authorize\_url     *Build an authorization URL*

---

**Description**

Build an authorization URL

**Usage**

```
oauth_authorize_url(client, scope = NULL, state = NULL)
```

**Arguments**

|        |  |
|--------|--|
| client | A [oauth_client] with an auth_url.         |
| scope  | Optional space-delimited scope string.     |
| state  | Optional opaque state for CSRF protection. |

**Value**

The authorization URL to open in a browser.

**Examples**

```
oauth_authorize_url(
  oauth_client("id", token_url = "https://x/token",
              auth_url = "https://x/authorize"),
  scope = "user-read-email")
```

---

oauth\_bearer     *Authorization header value for a token*

---

**Description**

Authorization header value for a token

**Usage**

```
oauth_bearer(token)
```

**Arguments**

|       |  |
|-------|--|
| token | A tinyoauth_token, a (legacy) httr Token2.0, or a raw access-token string. |
|-------|--|

**Value**

A string like "Bearer abc123" for use as an HTTP Authorization header.

**Examples**

```
## Not run:
h <- curl::new_handle()
curl::handle_setheaders(h, Authorization = oauth_bearer(tok))

## End(Not run)
```

---

|                  |  |
|------------------|--|
| oauth_cache_path | <i>Default on-disk cache path for a client's token</i> |
|------------------|--|

---

**Description**

Default on-disk cache path for a client's token

**Usage**

```
oauth_cache_path(client)
```

**Arguments**

client            A [oauth\_client].

**Value**

Path to the token cache file under tools::R\_user\_dir.

---

|              |                                   |
|--------------|-----------------------------------|
| oauth_client | <i>Define an OAuth 2.0 client</i> |
|--------------|-----------------------------------|

---

**Description**

Define an OAuth 2.0 client

**Usage**

```
oauth_client(id, secret = NULL, token_url, auth_url = NULL,
             redirect_uri = "http://127.0.0.1:1410/")
```

**Arguments**

|              |  |
|--------------|--|
| id           | Client (application) id.   |
| secret       | Client secret, or NULL for public clients.   |
| token_url    | The provider's token endpoint.   |
| auth_url     | The provider's authorization endpoint (needed for the authorization-code grant; omit for client-credentials only).           |
| redirect_uri | Redirect URI registered with the provider. Use a loopback IP literal over http (127.0.0.1); many providers reject localhost. |

**Value**

A tinyoauth\_client object.

**Examples**

```
spotify <- oauth_client(
  id = "your_id", secret = "your_secret",
  token_url = "https://accounts.spotify.com/api/token",
  auth_url = "https://accounts.spotify.com/authorize")
```

---

oauth\_exchange\_code      *Exchange an authorization code for a token*

---

**Description**

Exchange an authorization code for a token

**Usage**

```
oauth_exchange_code(client, code)
```

**Arguments**

- client                    A [oauth\_client].
- code                     The authorization code from the redirect.

**Value**

A tinyoauth\_token.

---

oauth\_expired            *Is a token expired?*

---

**Description**

Is a token expired?

**Usage**

```
oauth_expired(token, leeway = 60)
```

**Arguments**

- token                    A tinyoauth\_token.
- leeway                  Seconds of slack before the hard expiry (default 60).

**Value**

TRUE if expired (or within leeway of it); FALSE when there is no expiry recorded.

---

|                   |   |
|-------------------|---|
| oauth_import_httr | <i>Import an httr '.httr-oauth' cache into tinypass</i> |
|-------------------|---|

---

**Description**

Reads a token cached by **httr**'s `oauth2.0_token()` and returns a tinypass client and token built from it – the app credentials, endpoints, and (crucially) the refresh token. This lets a package migrating off **httr** reuse an existing authorization instead of forcing users to log in again.

**Usage**

```
oauth_import_httr(path = ".httr-oauth", which = 1L)
```

**Arguments**

|       |  |
|-------|--|
| path  | Path to the httr cache (default ".httr-oauth").                                |
| which | Which cached token to import when the file holds several (1-based; default 1). |

**Details**

The imported access token is marked expired, since httr's cached access token is usually stale: the durable credential is the refresh token. Pass the result to `[oauth_refresh]` or `[oauth_token]` to mint a fresh access token.

**Value**

A list with `client` (a `[oauth_client]`) and `token` (a `tinypass_token`).

**Examples**

```
## Not run:
imported <- oauth_import_httr("~/project/.httr-oauth")
token <- oauth_refresh(imported$client, imported$token)

## End(Not run)
```

---

|                   |                             |
|-------------------|-----------------------------|
| oauth_jwt_payload | <i>Decode a JWT payload</i> |
|-------------------|-----------------------------|

---

**Description**

Base64url-decodes the payload (middle) segment of a JSON Web Token and parses it as JSON. Does not verify the signature; use only on tokens you already trust (e.g. one the provider just issued you).

**Usage**

```
oauth_jwt_payload(x)
```

**Arguments**

x                    A JWT string, or a tinyoauth\_token (its access\_token is used).

**Value**

The decoded payload as a named list, or NULL if x has no usable JWT.

**Examples**

```
# A toy token: header.payload.signature, payload = {"sub":"abc"}
payload <- jsonlite::base64_enc(charToRaw('{"sub":"abc"}'))
jwt <- paste("x", gsub("=", "", payload), "y", sep = ".")
oauth_jwt_payload(jwt)$sub
```

---

|               |                                |
|---------------|--------------------------------|
| oauth_refresh | <i>Refresh an access token</i> |
|---------------|--------------------------------|

---

**Description**

Refresh an access token

**Usage**

```
oauth_refresh(client, token)
```

**Arguments**

client                A [oauth\_client].  
token                 A tinyoauth\_token carrying a refresh token.

**Value**

A refreshed `tinyoauth_token`. Providers that omit a new refresh token on refresh keep the existing one.

**Examples**

```
## Not run:
tok <- oauth_refresh(spotify, tok)

## End(Not run)
```

---

|               |                                      |
|---------------|--------------------------------------|
| oauth_request | <i>Make an authenticated request</i> |
|---------------|--------------------------------------|

---

**Description**

Sends an HTTP request with the token as a Bearer header, retrying transient failures, and parses a JSON response. A convenience over building a curl handle by hand; for anything exotic, use `[oauth_bearer]` with curl directly.

**Usage**

```
oauth_request(token, url, method = "GET", query = NULL, body = NULL,
             headers = NULL, flatten = FALSE, retries = 3L)
```

**Arguments**

|                      |   |
|----------------------|---|
| <code>token</code>   | A <code>tinyoauth_token</code> , a (legacy) httr token, or a raw access-token string. |
| <code>url</code>     | Endpoint URL.   |
| <code>method</code>  | HTTP method (default "GET").  |
| <code>query</code>   | Optional named list of query parameters.  |
| <code>body</code>    | Optional R object sent as a JSON body.  |
| <code>headers</code> | Optional named character vector of extra headers.                                     |
| <code>flatten</code> | Passed to <code>jsonlite::fromJSON</code> (default FALSE).                            |
| <code>retries</code> | Attempts on transport errors / HTTP 5xx (default 3).                                  |

**Value**

Parsed JSON, or invisibly NULL for an empty response body. Non-2xx responses raise an error carrying the status and body.

**Examples**

```
## Not run:
oauth_request(tok, "https://api.spotify.com/v1/me")

## End(Not run)
```

---

|             |  |
|-------------|--|
| oauth_token | <i>Get a valid token, using the cache and refreshing as needed</i> |
|-------------|--|

---

**Description**

Returns a cached token if still valid; refreshes it if expired and a refresh token is available; otherwise runs the authorization-code flow. The result is written back to cache.

**Usage**

```
oauth_token(client, scope = NULL, cache = oauth_cache_path(client), ...)
```

**Arguments**

|        |  |
|--------|--|
| client | A [oauth_client].  |
| scope  | Optional space-delimited scope string (for first authorization).             |
| cache  | Cache file path, or NULL to disable caching. Defaults to [oauth_cache_path]. |
| ...    | Passed to [oauth_token_authcode] (e.g. port, open_browser).                  |

**Value**

A valid tinyoauth\_token.

**Examples**

```
## Not run:
tok <- oauth_token(spotify, scope = "user-read-email")

## End(Not run)
```

---

|                       |   |
|-----------------------|---|
| oauth_token_anthropic | <i>Get a valid Anthropic Claude token, using the cache and refreshing as needed</i> |
|-----------------------|---|

---

**Description**

The Claude analogue of [oauth\_token]: returns a cached token if still valid, refreshes it if expired and a refresh token is available, otherwise runs the manual-paste login flow. The token is written back to cache.

**Usage**

```
oauth_token_anthropic(cache = oauth_cache_path(anthropic_claude_client()),
                      open_url = interactive(), login = TRUE)
```

**Arguments**

|          |  |
|----------|--|
| cache    | Cache file path, or NULL to disable caching. Defaults to [oauth_cache_path] for the Claude client.   |
| open_url | Open the authorization URL automatically (default: interactive sessions only).   |
| login    | Run the login flow when no usable cached/refreshable token exists (default TRUE). Pass FALSE to get the cached (and refreshed-if-needed) token or NULL, without ever prompting – useful inside a request path where an interactive login would be wrong. |

**Details**

These are subscription credentials minted for Claude Code; using them is subject to Anthropic's terms for that product.

**Value**

A tinyoauth\_token with access\_token, refresh\_token, and expires\_at; or NULL when login is FALSE and no usable token is cached.

**Examples**

```
## Not run:
tok <- oauth_token_anthropic()
curl::handle_setheaders(curl::new_handle(),
                        Authorization = oauth_bearer(tok),
                        "anthropic-beta" = "oauth-2025-04-20")

## End(Not run)
```

---

oauth\_token\_authcode *Run the authorization-code flow end to end*

---

**Description**

Prints (and optionally opens) the authorization URL, then obtains the redirect either by catching it on a loopback listener (default) or, with manual = TRUE, by having you paste the redirected URL back. After verifying state, it exchanges the code.

**Usage**

```
oauth_token_authcode(client, scope = NULL, port = 1410L,
                     open_browser = interactive(), timeout = 120, manual = NA)
```

**Arguments**

|              |   |
|--------------|---|
| client       | A [oauth_client] with an auth_url.  |
| scope        | Optional space-delimited scope string.  |
| port         | Loopback port for the listener; must match the port in client\$redirect_uri (default 1410).   |
| open_browser | Open the URL automatically (default: interactive only).   |
| timeout      | Seconds to wait for the redirect.   |
| manual       | Skip the loopback listener and read the redirected address (or bare code) from the console instead. The default (NA) auto-detects: it switches to manual on a remote/headless session (SSH, RStudio Server, or unix with no display), where the browser runs elsewhere and the redirect can never reach a local listener (so the listener would just hang). Pass TRUE/FALSE to force it. In manual mode the browser shows a "can't reach 127.0.0.1" page after you approve – that is expected; copy its address bar and paste it. |

**Value**

A tinyoauth\_token (with a refresh token, when the provider issues one).

**Examples**

```
## Not run:
tok <- oauth_token_authcode(spotify, scope = "user-read-email")
tok <- oauth_token_authcode(google, manual = TRUE) # force manual paste

## End(Not run)
```

---

oauth\_token\_client      *Fetch a token via the client-credentials grant*

---

**Description**

App-only access (no user context).

**Usage**

```
oauth_token_client(client)
```

**Arguments**

|        |                   |
|--------|-------------------|
| client | A [oauth_client]. |
|--------|-------------------|

**Value**

A tinyoauth\_token.

**Examples**

```
## Not run:
tok <- oauth_token_client(spotify)

## End(Not run)
```

---

```
oauth_token_openai_codex
```

*Get a valid OpenAI Codex token, using the cache and refreshing as needed*

---

**Description**

The Codex analogue of [oauth\_token]: returns a cached token if still valid, refreshes it if expired and a refresh token is available, otherwise runs the device-login flow. The token carries an extra account\_id field (the ChatGPT account id) and is written back to cache.

**Usage**

```
oauth_token_openai_codex(cache = oauth_cache_path(openai_codex_client()),
  open_url = interactive(), timeout = 600, login = TRUE)
```

**Arguments**

|          |   |
|----------|---|
| cache    | Cache file path, or NULL to disable caching. Defaults to [oauth_cache_path] for the Codex client.   |
| open_url | Open the verification URL automatically (default: interactive sessions only).   |
| timeout  | Seconds to wait for device authorization (default 600).   |
| login    | Run the device-login flow when no usable cached/refreshable token exists (default TRUE). Pass FALSE to get the cached (and refreshed-if-needed) token or NULL, without ever prompting – useful inside a request path where an interactive login would be wrong. |

**Value**

A tinyoauth\_token with access\_token, refresh\_token, expires\_at, and account\_id; or NULL when login is FALSE and no usable token is cached.

**Examples**

```
## Not run:
tok <- oauth_token_openai_codex()
curl::handle_setheaders(curl::new_handle(),
  Authorization = oauth_bearer(tok),
  "chatgpt-account-id" = tok$account_id)

## End(Not run)
```

---

`openai_codex_account_id`*Extract the ChatGPT account id from a Codex token*

---

**Description**

Reads the `chatgpt_account_id` claim that OpenAI nests under `https://api.openai.com/auth` in the access-token JWT.

**Usage**

```
openai_codex_account_id(token)
```

**Arguments**

`token`            A `tinyoauth_token` (or raw access-token string).

**Value**

The account id string, or NULL if absent.

---

`openai_codex_client`    *OAuth client for the OpenAI Codex (ChatGPT) device-login flow*

---

**Description**

A preconfigured `[oauth_client]` for ChatGPT-subscription-backed Codex access, carrying OpenAI's device-authorization endpoints alongside the standard token endpoint. The client id is OpenAI's public native-app identifier, not a secret.

**Usage**

```
openai_codex_client()
```

**Value**

A `tinyoauth_client` with extra `device_usercode_url`, `device_token_url`, and `verification_uri` fields.

**Examples**

```
openai_codex_client()
```

# Index

anthropic\_claude\_client, 2

oauth\_authorize\_url, 3

oauth\_bearer, 3

oauth\_cache\_path, 4

oauth\_client, 4

oauth\_exchange\_code, 5

oauth\_expired, 5

oauth\_import\_httr, 6

oauth\_jwt\_payload, 7

oauth\_refresh, 7

oauth\_request, 8

oauth\_token, 9

oauth\_token\_anthropic, 9

oauth\_token\_authcode, 10

oauth\_token\_client, 11

oauth\_token\_openai\_codex, 12

openai\_codex\_account\_id, 13

openai\_codex\_client, 13