Package 'tinyoauth'

June 17, 2026

Type Package

Title Minimal OAuth 2.0 Client

Version 0.1.0

Date 2026-06-10

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. Includes a helper for the 'OpenAI' <https://openai.com/> 'Codex' device login, a similar but non-standard variant of the OAuth 2.0 device authorization grant.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.0)

Imports curl, 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>), Sounkou Mahamane Toure [ctb], cornball.ai [cph]

Maintainer Troy Hernandez <troy@cornball.ai>

Repository CRAN

Date/Publication 2026-06-17 17:40:02 UTC

Contents

`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

```
tok <- structure(list(access_token = "abc123"), class = "tinyoauth_token")
oauth_bearer(tok)
# pass it to a request, e.g.:
#   curl::handle_setheaders(curl::new_handle(),
#                           Authorization = oauth_bearer(tok))
```

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

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.

Examples

```
client <- oauth_client("my_app", token_url = "https://example.com/token")
oauth_cache_path(client)
```

`oauth_client` *Define an OAuth 2.0 client*

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.

Examples

```
## Not run:
# `code` is the value the provider redirects back after the user approves.
tok <- oauth_exchange_code(client, code)

## End(Not run)
```

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.

Examples

```
expired <- structure(list(expires_at = Sys.time() - 1),
                     class = "tinyoauth_token")
oauth_expired(expired)
fresh <- structure(list(expires_at = Sys.time() + 3600),
                   class = "tinyoauth_token")
oauth_expired(fresh)
```

`oauth_import_httr` *Import an httr '.httr-oauth' cache into tinyoauth*

Description

Reads a token cached by **httr**'s `oauth2.0_token()` and returns a tinyoauth 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 `tinyoauth_token`).

Examples

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

## End(Not run)
```

oauth_jwt_payload *Decode a JWT payload*

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 *Refresh an access token*

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` *Make an authenticated request*

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

`token`	A `tinyoauth_token`, a (legacy) httr token, or a raw access-token string.
`url`	Endpoint URL.
`method`	HTTP method (default "GET").
`query`	Optional named list of query parameters.
`body`	Optional R object sent as a JSON body.
`headers`	Optional named character vector of extra headers.
`flatten`	Passed to `jsonlite::fromJSON` (default FALSE).
`retries`	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 *Get a valid token, using the cache and refreshing as needed*

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_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.

Examples

```
# A token whose access-token JWT carries the account-id claim:
claim <- jsonlite::toJSON(
  list("https://api.openai.com/auth" = list(chatgpt_account_id = "acct_123")),
  auto_unbox = TRUE)
b64url <- function(s) {
  chartr("+/", "-_", gsub("[\n=]", "", jsonlite::base64_enc(charToRaw(s))))
}
jwt <- paste("header", b64url(claim), "signature", sep = ".")
openai_codex_account_id(jwt)
```

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