Package ‘ellmercodex’

November 11, 2026

Title 'ChatGPT' Plan Provider for 'ellmer'

Version 0.2.1

URL https://simxnherrera.github.io/ellmercodex/,
https://github.com/simxnherrera/ellmercodex

BugReports https://github.com/simxnherrera/ellmercodex/issues

Description Connects 'ellmer' chats to a 'ChatGPT' subscription through 'OpenAI''s documented ``Sign in with ChatGPT'' flow for open-source, locally hosted apps <https://developers.openai.com/siwc/token-sharing-open-source>. Provides a stable, explicitly bounded core of the public 'ellmer' (>= 0.5.1) Chat object, including browser sign-in with dynamic client registration, package-scoped credential storage, redacted diagnostics, multi-turn text, streaming, structured-output, function-tool, asynchronous, cancellation, cloning, metadata, multimodal chat, account-specific model discovery, and reasoning-effort selection through the public 'Responses' API. Parallel and batch helper functions are explicitly unsupported because 'ChatGPT' plan usage is stream-only. This independent third-party integration is not endorsed by 'OpenAI'.

License MIT + file LICENSE

Depends R (>= 4.1.0)

Encoding UTF-8

Language en-US

Imports coro, ellmer (>= 0.5.1), httr2 (>= 1.3.0), httpuv, jose, jsonlite, openssl, promises, rlang, S7

Suggests callr, knitr, later, pkgload, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Simon Herrera [aut, cre, cph]

Maintainer Simon Herrera <simxnherrera@gmail.com>

Repository CRAN

Date/Publication 2026-11-11 00:20:02 UTC

Contents

chat_codex 2
codex_account 4
codex_available 5
codex_login 6
codex_logout 7
codex_models 8
ellmercodex 9
ellmercodex-conditions 10

Index **12**

`chat_codex` *Create a Codex chat backed by ellmer*

Description

`chat_codex()` returns a normal ellmer `Chat` object configured for the public Responses API with ChatGPT plan usage. It does not start browser authentication. Call `codex_login()` explicitly first when no stored credential is available; an existing credential is loaded and refreshed as needed.

Usage

```
chat_codex(
  system_prompt = NULL,
  model = NULL,
  echo = c("none", "output", "all"),
  effort = NULL,
  params = NULL,
  api_args = list()
)
```

Arguments

`system_prompt`	Optional system prompt passed to ellmer.
`model`	Optional Codex model name. If omitted, `chat_codex()` queries the authenticated account catalog and chooses its lowest-priority usable model. Set `ELLMERCODEX_MODEL` for an explicit environment override; model availability remains account- and workspace-specific. If discovery is empty or unavailable, construction fails with an actionable error.

`echo` One of `"none"`, `"output"`, or `"all"`; controls whether `$chat()` prints the completed text. The compatibility wrapper accepts `TRUE`, `FALSE`, and `"text"` as aliases for `"output"`, `"none"`, and `"output"`, respectively.

`effort` Optional Codex reasoning effort. It is forwarded through ellmer's `reasoning_effort` parameter. When the selected model appears in `codex_models()`, the value must be one of its advertised efforts. For an explicit model absent from that catalog, the service validates the model and value.

`params` Optional ellmer model parameters, usually created with `ellmer::params()`. A supplied `reasoning_effort` must agree with `effort`, when both are provided. Other ellmer-supported model parameters are passed through unchanged, except those ChatGPT plan usage rejects (for example `temperature`, `top_p`, and `max_tokens`), which fail before a request is sent.

`api_args` Optional named list of additional Responses arguments passed through ellmer on every request. This is an advanced escape hatch for Responses arguments accepted with ChatGPT plan usage. Arguments the documentation prohibits (such as `metadata`, `truncation`, or `previous_response_id`) fail before a request is sent; other unsupported arguments may be rejected by the remote service.

Details

The returned chat is the public `ellmer` `Chat` object (0.5.1 or later) for interactive, single-conversation operations, including ordinary and structured chat, synchronous and asynchronous streaming, tool declarations and multi-round execution, callbacks, cancellation, cloning, history, echo, model/provider configuration, rich content, and response metadata. Provider token counting and file management are unavailable with ChatGPT plan usage and fail before making a request. The compatibility layer uses one contract-checked provider/turn-submission seam while leaving public Chat methods and ellmer lifecycle semantics intact. The separate ellmer parallel/batch helpers are outside this stable core contract. Parallel helpers use the package blocker, while batch helpers stop in ellmer's generic unsupported-provider check because the Codex endpoint is stream-only.
`$chat_structured()` intentionally follows ellmer's semantics and disables registered tools for that request. Call `$chat()` first when tool-assisted context is needed, then use `$chat_structured()` to extract structured data.

Value

An object inheriting from ellmer's `Chat` class. The returned object retains ellmer's public Chat methods, history, callbacks, registered tools, asynchronous methods, and R6 cloning behavior.

Conditions

Invalid arguments signal `codex_chat_argument_error`. Missing or incompatible ellmer installations signal `codex_ellmer_missing` or `codex_ellmer_compatibility_error`. Authentication failures found before the Chat is constructed retain their package condition classes. Transport and protocol errors retain their specific Codex/ellmer condition classes; compatibility-shape failures use `codex_ellmer_compatibility_error`.

Side effects

Constructing a chat reads the current process credential or this package's credential store and may refresh an expired stored credential. When `model` is omitted, or when an effort is supplied, it also queries the authenticated account catalog. It does not open a browser or generate a response.

See Also

`codex_login()`, `codex_account()`, `codex_models()`, `codex_logout()`, and `ellmer::chat_openai()`

Examples

```
chat <- chat_codex(
  system_prompt = "Be concise."
)
chat$chat("Hello")

weather_tool <- ellmer::tool(
  function(city) paste("Sunny in", city),
  name = "get_weather",
  description = "Get the current weather for a city.",
  arguments = list(city = ellmer::type_string())
)
chat$register_tool(weather_tool)
chat$chat("What is the weather in Montevideo?")
```

`codex_account` *Show a redacted Codex authentication summary.*

Description

The account identifier and all token material are deliberately replaced or omitted. Calling this function does not open a browser; an expired stored credential may be refreshed.

Usage

```
codex_account(auth = NULL)
```

Arguments

`auth`	Optional in-memory credential. If omitted, the current process-local session and then the package's own credential store are checked.

Value

A one-row data frame with columns:

`authenticated`	Logical; whether a valid package credential was found.
`account`	Always `"<redacted>"`; account identifiers are never returned.
`expires_at`	The access-token expiry as a UTC `POSIXct` value, or `NA` when unauthenticated.

Conditions

A malformed in-memory credential signals `codex_auth_argument_error` and a malformed stored value signals `codex_credential_store_error`.

Examples

```
codex_account()
```

`codex_available` *Check whether the local ellmercodex integration is available.*

Description

This is an offline check. By default it verifies package dependencies and static configuration only; it does not open a browser, make a request, or inspect a credential store. Set `check_credentials = TRUE` when an explicit credential-presence check is desired.

Usage

```
codex_available(check_credentials = FALSE)
```

Arguments

`check_credentials`
: Whether to inspect this package's own credential backend. Defaults to `FALSE` to keep ordinary availability checks free of credential-store I/O.

Value

A single logical value. `TRUE` means that the required dependencies, static configuration, and the supported `ellmer` compatibility seam are available. With `check_credentials = TRUE`, it also requires a usable package-owned credential to be present. `FALSE` does not identify a single cause; use the package startup diagnostics or contact the package maintainer with a sanitized reproducible report.

Note

The default check does not prove that the remote Codex service is reachable, that a model is available to the account, or that a request will succeed.

Examples

```
codex_available()
if (interactive()) codex_available(check_credentials = TRUE)
```

codex_login *Sign in with ChatGPT in the browser.*

Description

`codex_login()` is the explicit authentication entry point. It follows OpenAI's documented "Sign in with ChatGPT" flow for open-source, locally hosted apps: it opens the system browser for registration and consent, listens once on the loopback callback, exchanges the PKCE-protected code for tokens with the issued client ID, verifies the ID token, and checks that ChatGPT plan usage (`chatgpt.tokens.use.direct`) was granted.

Usage

```
codex_login(persist = TRUE, timeout = 300)
```

Arguments

`persist`	Whether to save the credential for later sessions. The default is `TRUE`; the credential is written to this package's own directory (see Details) with owner-only permissions. Use `FALSE` for a process-only session.
`timeout`	Maximum callback wait in seconds. This must be one positive number. The loopback listener is closed when authentication succeeds, fails, or times out. When `persist = FALSE`, the credential remains available to `chat_codex()` only in the current R process.

Details

The first sign-in creates a registration bound to the selected ChatGPT account and workspace. Later sign-ins reauthorize that registration. Use `codex_logout()` first to sign in with a different account or workspace. The package never reads credentials created by Codex CLI or another application.

Credentials are stored in `tools::R_user_dir("ellmercodex", "config")`, or in the directory named by the `ELLMERCODEX_HOME` environment variable. The same directory holds a stable, opaque host identifier (`ext_agent_host_id`) that is created before the first sign-in, even when `persist = FALSE`, because the flow requires it to be stable for each host.

Value

An internal `codex_auth` object. Use `codex_account()` for a safe account summary; token fields are intentionally not printed.

Conditions

Invalid arguments signal `codex_auth_argument_error`; callback and token failures use `codex_oauth_callback_error`, `codex_oauth_timeout`, or `codex_token_exchange_error` as appropriate. If ChatGPT plan usage was not granted, `codex_plan_scope_error` is signaled.

Note

This function is interactive: it opens the default browser and listens on `http://127.0.0.1:1455/callback` (set `ELLMERCODEX_CALLBACK_PORT` to use another port).

Examples

```
auth <- codex_login()
codex_account(auth)
```

---

codex_logout *Sign out and remove ellmercodex's stored credential.*

Description

This function revokes the refresh token with OpenAI (by default), clears the process-local session, and deletes this package's own credential file. It never removes Codex CLI credentials or another application's entries. The host identifier is kept because the documented flow requires it to stay stable for this host.

Usage

```
codex_logout(revoke = TRUE)
```

Arguments

`revoke`	Whether to revoke the refresh token at OpenAI's documented revocation endpoint before deleting it locally. Revocation is best effort: local credentials are removed even when the network request fails, in which case a `codex_revocation_warning` warning is signaled.

Value

`TRUE`, invisibly. The process-local credential is cleared even if no persistent credential exists.

Conditions

Logout signals no backend details. Storage failures are reported as `codex_credential_store_error` by the explicit login and refresh paths that write credentials.

Note

To use a different ChatGPT account or workspace, call `codex_logout()` and then `codex_login()`; the next sign-in creates a new registration instead of reauthorizing the previous one.

Examples

```
codex_logout()
```

codex_models *List models available to the authenticated Codex account.*

Description

This makes one authenticated request to OpenAI's documented model catalog (`GET https://api.openai.com/v1/models`) with the ChatGPT plan access token, and keeps the models marked for listing (`visibility: "list"`). The returned rows include each model's advertised reasoning effort levels, so callers can select an effort without copying a stale catalog into this package. Availability is account- and workspace-specific. The catalog may lag behind models accepted by the chat service; explicitly supplied IDs absent from the catalog are passed to the service for validation.

Usage

```
codex_models(auth = NULL, client_version = NULL)
```

Arguments

`auth`	Optional package credential. If omitted, the current session or package-owned credential is loaded and refreshed as needed.
`client_version`	Deprecated and ignored. The documented catalog does not take a client version. Kept so existing calls keep working.

Value

A `codex_models` data frame with one row per catalog model. Its columns are:

`id`	The model identifier accepted by chat_codex.
`owned_by`	The service or owner label advertised by the catalog.
`display_name`	A human-readable model name.
`description`	The catalog description, when supplied.
`default_reasoning_effort`	The advertised default effort.
`supported_reasoning_efforts`	A list-column of effort names.
`supported_in_api`	Whether the catalog marks the model as API-supported.
`priority`	The catalog priority, when supplied.
`default_service_tier`	The advertised default service tier.
`service_tiers`	A list-column of advertised service tiers.

The `supported` and `source` attributes describe the catalog result.

Conditions

Missing credentials signal `codex_auth_missing`; HTTP and protocol failures use the same sanitized transport conditions as generation requests.

Note

The catalog is account- and workspace-specific and is queried at call time. Cache or select a model explicitly if a reproducible workflow must not change when the account's catalog changes.

Examples

```
models <- codex_models()
models[c("id", "default_reasoning_effort", "supported_reasoning_efforts")]
```

`ellmercodex` *ellmercodex: ChatGPT plan provider for ellmer*

Description

`ellmercodex` provides a stable, explicitly bounded core integration between ellmer and a ChatGPT subscription. It uses OpenAI's documented "Sign in with ChatGPT" flow for open-source, locally hosted apps (`https://developers.openai.com/siwc/token-sharing-open-source`) and the public Responses API. The stable compatibility target is the public ellmer `Chat` (0.5.1 or later) object for interactive, single-conversation operations. This package is independent and is not affiliated with or endorsed by OpenAI.

Details

Authentication is never started as a package-loading side effect. Call `codex_login()` explicitly before `chat_codex()` when no stored credential is available. Examples and offline tests do not authenticate, open a browser, read credentials, or make network requests.

The usual workflow is to check `codex_available()`, authenticate with `codex_login()`, inspect the account with `codex_account()`, and then create a chat with `chat_codex()`. `codex_models()` can be used after sign-in to inspect the account-specific model catalog and its advertised reasoning efforts. `codex_logout()` removes only the credential owned by this package.

The public API is deliberately small: `codex_login()`, `codex_logout()`, `codex_account()`, `codex_models()`, `codex_available()`, and `chat_codex()`. The chat compatibility layer is checked against the required ellmer contracts at runtime and supports interactive Chat methods: text and content streaming, model parameters, per-model reasoning effort, structured output, multi-turn history, rich image/PDF content, asynchronous chat, tool loops, callbacks, cancellation, cloning, echo, and response metadata. Provider token counting and file management are unavailable. The separately exported ellmer parallel/batch helpers are outside the stable core contract and are explicitly blocked for the Codex stream-only endpoint rather than silently degraded. ChatGPT plan usage also rejects some Responses arguments, such as `temperature` and `max_output_tokens`; these fail before a request is sent.

External service and compatibility

ChatGPT plan usage through "Sign in with ChatGPT" is a preview feature for open-source and locally hosted apps; paid or remotely hosted apps need OpenAI's approval. It requires an eligible ChatGPT plan and workspace, and its limits and supported features may change. The package does not claim affiliation with or endorsement by OpenAI.

Author(s)

Maintainer: Simon Herrera <simxnherrera@gmail.com> [copyright holder]

Authors:

- Simon Herrera <simxnherrera@gmail.com> [copyright holder]

See Also

Useful links:

- https://simxnherrera.github.io/ellmercodex/
- https://github.com/simxnherrera/ellmercodex
- Report bugs at https://github.com/simxnherrera/ellmercodex/issues

---

ellmercodex-conditions

ellmercodex condition classes

Description

ellmercodex signals structured conditions so callers can handle failures without parsing messages. Messages are sanitized and never intentionally include tokens, authorization codes, OAuth state, account identifiers, or credential-store contents.

Details

Authentication and credential conditions inherit from `codex_auth_error`. The principal subclasses are `codex_auth_missing`, `codex_auth_argument_error`, `codex_oauth_argument_error`, `codex_oauth_callback_erro` `codex_oauth_timeout`, `codex_token_exchange_error`, `codex_refresh_error`, `codex_account_error`, `codex_credential_store_error`, and `codex_plan_scope_error` (ChatGPT plan usage, the `chatgpt.tokens.use.direc` scope, was not granted). Refresh and token-exchange errors carry the OAuth error code in an `oauth_error` field when the server returned one. `codex_logout()` signals the warning `codex_revocation_warning` when remote revocation fails.

Transport and stream conditions are `codex_request_error`, `codex_authentication_error`, `codex_rate_limit_error`, `codex_model_unavailable_error`, `codex_malformed_request_error`, `codex_server_error`, `codex_network_error`, `codex_protocol_error`, `codex_protocol_changed_error`, `codex_generation_error`, and `codex_incomplete_error`.

Documented ChatGPT plan usage error codes map to more specific subclasses that keep the parents above, and carry the server's `code` and `param` fields: `codex_usage_limit_error` (`subscription_sharing_usage_limit_e`

also a codex_rate_limit_error), codex_usage_unavailable_error (subscription_sharing_usage_unavailable and subscription_sharing_user_unavailable; also a codex_server_error), codex_plan_ineligible_error (subscription_sharing_user_not_eligible; also a codex_authentication_error), and codex_unsupported_capabi (subscription_sharing_unsupported_capability; also a codex_malformed_request_error).

ellmer integration conditions are codex_chat_argument_error, codex_chat_error, codex_model_selection_error, codex_ellmer_missing, and codex_ellmer_compatibility_error.

Handle these classes with condition handlers rather than matching their messages. More specific subclasses can be handled before the common codex_auth_error or codex_request_error parents.

Examples

```
error <- tryCatch(
  codex_available(NA),
  codex_auth_argument_error = identity
)
inherits(error, "codex_auth_error")
```

Index

∗ **package**
ellmercodex, 9

chat_codex, 2, *8*
chat_codex(), *6*, *9*
codex_account, 4
codex_account(), *4*, *6*, *9*
codex_account_error
(ellmercodex-conditions), 10
codex_auth_argument_error
(ellmercodex-conditions), 10
codex_auth_error
(ellmercodex-conditions), 10
codex_auth_missing
(ellmercodex-conditions), 10
codex_authentication_error
(ellmercodex-conditions), 10
codex_available, 5
codex_available(), *9*
codex_chat_argument_error
(ellmercodex-conditions), 10
codex_chat_error
(ellmercodex-conditions), 10
codex_credential_store_error
(ellmercodex-conditions), 10
codex_ellmer_compatibility_error
(ellmercodex-conditions), 10
codex_ellmer_missing
(ellmercodex-conditions), 10
codex_generation_error
(ellmercodex-conditions), 10
codex_incomplete_error
(ellmercodex-conditions), 10
codex_login, 6
codex_login(), *2*, *4*, *7*, *9*
codex_logout, 7
codex_logout(), *4*, *6*, *9*
codex_malformed_request_error
(ellmercodex-conditions), 10
codex_model_selection_error
(ellmercodex-conditions), 10
codex_model_unavailable_error
(ellmercodex-conditions), 10
codex_models, 8
codex_models(), *3*, *4*, *9*
codex_network_error
(ellmercodex-conditions), 10
codex_oauth_argument_error
(ellmercodex-conditions), 10
codex_oauth_callback_error
(ellmercodex-conditions), 10
codex_oauth_timeout
(ellmercodex-conditions), 10
codex_plan_ineligible_error
(ellmercodex-conditions), 10
codex_plan_scope_error
(ellmercodex-conditions), 10
codex_protocol_changed_error
(ellmercodex-conditions), 10
codex_protocol_error
(ellmercodex-conditions), 10
codex_rate_limit_error
(ellmercodex-conditions), 10
codex_refresh_error
(ellmercodex-conditions), 10
codex_request_error
(ellmercodex-conditions), 10
codex_revocation_warning
(ellmercodex-conditions), 10
codex_server_error
(ellmercodex-conditions), 10
codex_token_exchange_error
(ellmercodex-conditions), 10
codex_unsupported_capability_error
(ellmercodex-conditions), 10
codex_usage_limit_error
(ellmercodex-conditions), 10
codex_usage_unavailable_error

(ellmercodex-conditions), 10

ellmer, 9
ellmer::chat_openai(), 4
ellmercodex, 9
ellmercodex-conditions, 10
ellmercodex-package (ellmercodex), 9