Package 'corteza'

May 29, 2026

Title AI Agent Runtime

Version 0.6.9

Description An agent runtime that gives Large Language Models (LLMs) from 'Anthropic' <https://www.anthropic.com/>, 'OpenAI' <https://openai.com/>, 'Moonshot' <https://www.moonshot.ai/>, and 'Ollama' <https://ollama.com/> direct access to a live R session with managed workspace state. Tools execute as R function calls with provenance tracking, and a deterministic retrieval system keeps relevant objects in context across turns. Three entry points: a shell command-line interface (CLI), a console read-eval-print-loop via chat(), and a Model Context Protocol (MCP) server via serve() for external clients.

License Apache License (>= 2)

URL https://github.com/cornball-ai/corteza

BugReports https://github.com/cornball-ai/corteza/issues

Depends R (>= 4.4.0)

Imports callr, codetools, curl, jsonlite, llm.api (>= 0.1.4), printify, processx, saber

Suggests clipr, fortunes, mx.api, rstudioapi, simplermarkdown, tinytest

VignetteBuilder simplermarkdown

SystemRequirements On Windows, Rtools45 (R 4.5.x) or Rtools44 (R 4.4.x) is recommended so the 'bash' shell tool is available; minimal installs fall back to a 'cmd' tool. 'git' is required for the git_status, git_diff, and git_log tools (install Git for Windows, or 'pacman -Sy git' from an Rtools shell).

Encoding UTF-8

NeedsCompilation no

Author Troy Hernandez [aut, cre] (ORCID: <https://orcid.org/0009-0005-4248-604X>), Grant McDermott [ctb] (ORCID: <https://orcid.org/0000-0001-7883-8573>), Jorge Krzyzaniak [ctb], cornball.ai [cph]

Maintainer Troy Hernandez <troy@cornball.ai>

Repository CRAN

Date/Publication 2026-05-28 23:50:08 UTC

Contents

`add_observer` *Add a tool-call observer to a session*

Description

Observers run after every tool call (run, denied, or declined). They receive a single `event` list with fields:

Usage

```
add_observer(session, observer)
```

Arguments

session	A session environment from new_session.
observer	A function of one argument (the event list).

Details

- call — the call list passed to policy().
- decision — the policy decision for the call.
- outcome — one of "ran", "deny", "declined".
- result — the string returned to the LLM.
- success — logical; TRUE only for "ran" with no tool error.
- elapsed_ms — wall time including policy overhead.
- turn_number — the session's tool-call counter.

Errors raised inside an observer are swallowed.

Value

The session, invisibly.

Examples

```
s <- new_session(provider = "anthropic")
add_observer(s, function(event) {
    # An observer is just a function of one argument; record the
    # outcome for inspection.
    message(event$outcome)
})
length(s$on_tool)
```

---

chat *Start Interactive Chat*

Description

Run a conversational agent inside your R session. Tools execute as direct function calls, no MCP server needed.

Usage

```
chat(provider = NULL, model = NULL, tools = NULL, session = NULL,
     max_turns = NULL)
```

Arguments

provider	LLM provider: "anthropic", "openai", "moonshot", or "ollama". Defaults to config value or "anthropic".
model	Model name. Defaults to config value or provider default.
tools	Character vector of tool names or categories to enable. Categories: file, code, r, data, web, git, chat, memory. Use "core" for file+code+git, "all" for everything (default).
session	Session resume control. NULL (default) starts fresh, TRUE resumes the latest session, or a character session key to resume a specific session.
max_turns	Integer or NULL. Maximum LLM turns per user prompt before the loop stops with [Max turns reached]. NULL (default) reads getOption("corteza.max_turns"), then falls back to the session_setup default (50).

Value

The session object (invisibly).

Examples

```
if (interactive()) {
    # Start chatting with defaults from config
    chat()

    # Use a specific provider/model
    chat(provider = "ollama", model = "llama3.2")

    # Minimal tools for focused work
    chat(tools = "core")
}
```

---

default_local_model *Detect the preferred local Ollama model*

Description

Walks getOption("corteza.local_models") (default c("gpt-oss:120b", "gpt-oss:20b")) and returns the first one that is currently installed in the local Ollama server. Returns NULL if Ollama is unreachable or none of the candidates are installed. Cached per R process.

Usage

```
default_local_model()
```

Value

Character scalar model name, or NULL.

Examples

```
# NULL when Ollama isn't running locally; a model name otherwise.
model <- default_local_model()
is.null(model) || is.character(model)
```

`install_cli` *Install corteza CLI*

Description

Install the `corteza` command-line tool to a directory in your PATH. On Unix (Linux, macOS) installs the Rscript shebang binary. On Windows installs a `.cmd` wrapper alongside the script so `corteza` works from cmd.exe / PowerShell.

Usage

```
install_cli(path = NULL, force = FALSE)
```

Arguments

`path`	Directory to install to. Default is `~/bin` on Unix, `tools::R_user_dir("corteza", "data")/bin` on Windows.
`force`	Overwrite existing installation.

Details

Requires:

- `r` (littler) for fast R script execution (Unix only — Windows uses `Rscript`).
- The `llm.api` package for LLM connectivity
- The `corteza` package itself

After installation, run `corteza` from any terminal (you may need to add the install directory to PATH; the function prints the PATH hint if it isn't already there).

Value

The installed script path, invisibly.

Examples

```
## Not run:
install_cli()
install_cli("/usr/local/bin")

## End(Not run)
```

matrix_archive_all *Flush all in-memory matrix sessions to the pensar vault*

Description

Walks the per-room session registry and archives any turns that haven't been ingested yet via the pensar archive ingest. Each session tracks an `ingested_through` watermark so repeated calls only write new turns. Silent no-op when `pensar` is not installed.

Usage

```
matrix_archive_all(sessions, mx_sess = NULL)
```

Arguments

`sessions`	A registry environment built by `matrix_run`/`matrix_poll`. Keys are room IDs, values are session lists carrying `$history`.
`mx_sess`	Optional Matrix session for room-name lookups. When NULL, the room ID is used as the source identifier.

Value

Integer count of rooms ingested, invisibly.

Examples

```
## Not run:
# Requires a running Matrix session registry and the optional
# pensar package for the actual archive step.
reg <- new.env(parent = emptyenv())
matrix_archive_all(reg)

## End(Not run)
```

matrix_configure *Configure the Matrix channel for this host*

Description

Logs in to a Matrix homeserver as the bot account, joins (or records) the target room, and writes credentials to `tools::R_user_dir("corteza", "config")/matrix.json` with file mode 0600. Call once per host. Model, provider, tools_filter, and auto_approve_asks are defaults the poll loop uses unless overridden at call time.

Usage

```
matrix_configure(server, user, password, room, model = NULL,
                 provider = c("anthropic", "openai", "moonshot", "ollama"),
                 tools_filter = NULL, auto_approve_asks = FALSE)
```

Arguments

server	Character. Homeserver base URL.
user	Character. Bot localpart or full Matrix ID.
password	Character. Bot password. Stored locally so the bot can re-authenticate if its access token is invalidated.
room	Character. Room ID or alias the bot should read and post to. If the bot has been invited but not joined, it will be joined.
model	Character or NULL. Default model name.
provider	Character. LLM provider: "anthropic", "openai", "moonshot", or "ollama".
tools_filter	Character vector or NULL. Passed to `get_tools()` to restrict which tools the bot can invoke. NULL allows all registered tools.
auto_approve_asks	Logical. When TRUE, tool calls that policy returns `"ask"` for are auto-approved. Suitable for a personal bot on a trusted tailnet. When FALSE (default) asks are declined until the thumbs-up reaction protocol lands.

Details

Pre-CRAN releases stored the file at `~/.corteza/matrix.json`; that path is still read for backward compatibility, but the next `matrix_configure()` call writes to the new location.

Value

The saved configuration, invisibly.

Examples

```
## Not run:
# Requires a real Matrix server and bot credentials. Configuration
# is written under tools::R_user_dir("corteza", "config").
matrix_configure(
    server = "https://matrix.example.org",
    user = "bot",
    password = "secret",
    room = "!roomid:example.org"
)

## End(Not run)
```

`matrix_poll` *One iteration of sync-and-reply*

Description

Fetches new messages across all joined rooms and runs `turn` against each. Auto-joins any pending invites the bot has received. Replies are sent back to the originating room. On first run there is no saved sync token, so this call establishes a baseline and returns without processing history.

Usage

```
matrix_poll(system = NULL, model = NULL, provider = NULL, tools_filter = NULL,
            timeout = 0L, sessions = NULL)
```

Arguments

`system`	Character or NULL. System prompt override.
`model`	Character or NULL. Model override.
`provider`	Character or NULL. Provider override.
`tools_filter`	Character vector or NULL. Tool filter override.
`timeout`	Integer. Long-poll timeout in milliseconds. 0 returns immediately.
`sessions`	Environment from `matrix_new_session_registry()` keyed by room_id, or NULL to build fresh sessions each call.

Details

Pass `sessions = NULL` (the default) for a stateless one-shot — each incoming message builds a fresh session. Pass a registry created by `matrix_new_session_registry()` so a long-running `matrix_run` keeps a separate history per room (conversations in different rooms don't cross-contaminate).

Value

An integer count of messages replied to, invisibly.

Examples

```
## Not run:
# Single poll cycle against the configured Matrix homeserver.
matrix_poll(timeout = 5000L)

## End(Not run)
```

matrix_request_flush *Ask the running matrix bot to archive sessions to pensar*

Description

Drops an `archive.signal` file in the corteza state directory. The next iteration of the long-poll loop in `matrix_run` picks it up, runs `matrix_archive_all`, and removes the file. Safe to call from any process or scheduler — systemd, Task Scheduler, launchd, cron, or a separate R session — without needing to know the bot's PID or share its memory.

Usage

```
matrix_request_flush()
```

Value

The signal file path, invisibly.

Examples

```
# Writes a sentinel file under CORTEZA_STATE_DIR (or the package's
# R_user_dir data path). Redirect to a tempdir for the example so
# we don't touch persistent state.
old <- Sys.getenv("CORTEZA_STATE_DIR")
Sys.setenv(CORTEZA_STATE_DIR = file.path(tempdir(), "state"))
sig <- matrix_request_flush()
file.exists(sig)
unlink(Sys.getenv("CORTEZA_STATE_DIR"), recursive = TRUE)
Sys.setenv(CORTEZA_STATE_DIR = old)
```

matrix_run *Run the Matrix adapter as a long-poll loop*

Description

Creates one session up front and reuses it across polls so conversation history accumulates within the process lifetime. Intended as the entry point for a systemd user unit.

Usage

```
matrix_run(timeout = 30000L, system = NULL, model = NULL, provider = NULL,
           tools_filter = NULL)
```

Arguments

timeout	Integer. Long-poll timeout in milliseconds.
system	Character or NULL. System prompt override.
model	Character or NULL. Model override.
provider	Character or NULL. Provider override.
tools_filter	Character vector or NULL. Tool filter override.

Value

Never returns under normal operation. Crashes on fatal error so systemd can restart.

Examples

```
## Not run:
# Run the Matrix bot loop -- typically launched by a systemd unit
# rather than from an interactive R session.
matrix_run()

## End(Not run)
```

---

matrix_send *Send a message to a Matrix room*

Description

Send a message to a Matrix room

Usage

```
matrix_send(text, room_id = NULL, msgtype = "m.text")
```

Arguments

text	Character. Plain text body.
room_id	Character. Matrix room id. Defaults to cfg$room_id from the saved Matrix config (see matrix_configure).
msgtype	Character. Matrix msgtype, default "m.text".

Value

The event ID of the sent message.

Examples

```
## Not run:
# Requires matrix_configure() to have run.
matrix_send("hello from corteza")

## End(Not run)
```

`mcp_tool_executor` *Build a tool executor that routes through an MCP connection*

Description

Returns a closure suitable for the `tool_executor` argument of `turn`. Each tool call is forwarded to the connected MCP server via `llm.api::mcp_call`.

Usage

```
mcp_tool_executor(conn)
```

Arguments

`conn`	An open MCP connection (from `llm.api::mcp_connect`).

Value

A function with signature `function(name, args)` that returns an MCP-format result list.

Examples

```
## Not run:
# Needs an open MCP connection to a running corteza::serve().
conn <- llm.api::mcp_connect("tcp://localhost:7850")
executor <- mcp_tool_executor(conn)
s <- new_session(provider = "anthropic")
turn("Hello", s, tool_executor = executor)

## End(Not run)
```

new_session *Create a new turn session*

Description

Returns an environment with sensible defaults. Adapters set channel- specific fields (e.g. `approval_cb`, `tools_filter`) before calling `turn`.

Usage

```
new_session(channel = c("cli", "console", "matrix"), history = NULL,
            model_map = NULL, provider = "anthropic", tools_filter = NULL,
            system = NULL, approval_cb = NULL, max_turns = 10L,
            verbose = FALSE, plan_mode = FALSE)
```

Arguments

`channel`	Character, one of "cli", "console", "matrix".
`history`	List of prior messages, or NULL.
`model_map`	Named list with `cloud` and `local` model names. Defaults to configured defaults.
`provider`	LLM provider passed to `llm.api::agent`.
`tools_filter`	Character vector passed to `get_tools()`.
`system`	System prompt override (NULL for built-in default).
`approval_cb`	Function called when policy returns `"ask"`. Signature: `function(call, decision) -> TRUE|FALSE`. Default denies (safe fallback).
`max_turns`	Maximum LLM turns per call.
`verbose`	Print tool call progress.
`plan_mode`	Logical. When TRUE, the session is in plan mode: the LLM is told to research and propose without acting, the policy engine denies write/exec tool calls (except `exit_plan_mode`), and `exit_plan_mode` is added to the tool list. A successful `exit_plan_mode` call flips this back to FALSE.

Value

An environment holding the session state.

Examples

```
# Build a stateless session for the CLI channel without making any
# network calls. The returned environment carries history, the
# active provider/model, and the approval callback.
s <- new_session(channel = "cli", provider = "anthropic")
is.environment(s)
identical(s$provider, "anthropic")
```

observer_progress *Built-in progress observer that prints to stdout*

Description

Prints one line per tool call suitable for an interactive REPL: `"[tool] hint (N lines)\n"`. The hint is a short summary of the call (file path, code snippet, search pattern) computed by `tool_hint()`.

Usage

```
observer_progress()
```

Value

A function to pass to `add_observer`.

Examples

```
obs <- observer_progress()
s <- new_session(provider = "anthropic")
add_observer(s, obs)
```

policy *Evaluate policy for a tool call*

Description

Returns a decision `list(model, approval, reason)`. `model` is `"cloud"` or `"local"`; `approval` is `"allow"`, `"ask"`, or `"deny"`.

Usage

```
policy(call, config = NULL)
```

Arguments

`call`	A list describing the tool call. See the file header in `R/policy.R` for the expected fields.
`config`	Optional config list from `load_config()`. When `NULL` (default), only the built-in tensor and user policy apply; this matches the historical `policy(call)` contract.

Details

When `config` is supplied, the project's `approval_mode` / `dangerous_tools` / per-tool `permissions` are overlaid on top of the default tensor: a tool the user has configured as `"ask"` or `"deny"` will have its decision promoted accordingly. Safety verdicts (credential paths, plan mode) still win because those represent invariants the user can't waive from config.

Both `corteza::chat()` and the CLI tool dispatch loop pass their session's config through here so the `/permissions` contract advertised by both surfaces is enforced consistently.

Value

A decision list with fields `model`, `approval`, `reason`.

Examples

```
# A bare-environment read_file call resolves under the default
# built-in policy without needing any session config.
decision <- policy(list(name = "read_file",
                        arguments = list(path = "DESCRIPTION")))
decision$approval
```

`serve` *Start MCP Server*

Description

Start the corteza MCP server. This exposes R tools to MCP clients like Claude Desktop, VS Code, or the corteza CLI.

Usage

```
serve(port = NULL, cwd = NULL, tools = NULL, expose_subagents = NULL)
```

Arguments

`port`	Port number for socket transport. If NULL, uses stdio transport.
`cwd`	Working directory for the server. Defaults to current directory.
`tools`	Character vector of tools or categories to enable. Categories: file, code, r, data, web, git, chat. Use "core" for file+code+git, "all" for everything (default).
`expose_subagents`	Whether MCP clients may call the subagent tools (‘spawn_subagent‘, ‘query_subagent‘, ‘collect_subagent‘, ‘list_subagents‘, ‘kill_subagent‘). ‘NULL‘ (default) defers to the ‘subagents$expose_over_mcp‘ config flag (itself FALSE by default); ‘TRUE‘/‘FALSE‘ overrides it. Off by default because a spawned subagent runs its own agent loop and spends autonomously on the host's LLM credentials – an unattended MCP client could otherwise trigger unbounded cost the client never sees. When on, cumulative subagent spend is capped by ‘subagents$mcp_spend_cap_usd‘ (default $5.00).

Details

The server supports two transport modes:

- **stdio** (default): For Claude Desktop and other MCP clients. Communication happens via stdin/stdout.

- **socket**: For the corteza CLI and R clients. Listens on a TCP port.

Tools Provided

- ‘read_file‘, ‘write_file‘, ‘replace_in_file‘, ‘list_files‘, ‘grep_files‘ - File operations - ‘run_r‘ - Execute R code in the server session - ‘bash‘ - Run shell commands - ‘r_help‘ - Query package docs via saber (exports, function help) - ‘installed_packages‘ - List installed packages - ‘web_search‘ - Search the web via Tavily (requires TAVILY_API_KEY) - ‘fetch_url‘ - Fetch web content - ‘git_status‘, ‘git_diff‘, ‘git_log‘ - Git operations - ‘chat‘, ‘chat_models‘ - LLM chat (requires llm.api)

Value

NULL (runs until interrupted or client disconnects)

Examples

```
## Not run:
# For Claude Desktop (stdio)
serve()

# For corteza CLI (socket) with all tools
serve(port = 7850)

# Minimal tools for small context models
serve(port = 7850, tools = "core")

# Specific categories
serve(port = 7850, tools = c("file", "git"))

## End(Not run)
```

---

session_setup *Configure and construct a session for any channel*

Description

Performs pre-turn setup common to all channels:

Usage

```
session_setup(channel = c("cli", "console", "matrix"), cwd = getwd(),
              provider = NULL, model = NULL, tools = NULL, system = NULL,
              approval_cb = NULL, history = NULL, load_project_context = TRUE,
              validate_api_key = TRUE, verbose = FALSE, max_turns = 50L)
```

Arguments

channel	Character, one of "cli", "console", "matrix".
cwd	Working directory. Defaults to the current directory.
provider	Character or NULL. LLM provider override. NULL falls back to config$provider, then "anthropic".
model	Character or NULL. Model override. NULL falls back to config$model, then the provider default.
tools	Character vector, NULL, or the string "all". Tool filter passed through to get_tools(). NULL is treated as "all".
system	Character or NULL. System prompt. NULL auto-builds via load_context(cwd) when load_project_context = TRUE, otherwise left NULL (channel supplies its own).
approval_cb	Function or NULL. Approval callback for "ask" verdicts; see new_session.
history	List or NULL. Prior conversation messages to seed the session with (each entry a list with role and content).
load_project_context	Logical. When TRUE, auto-call load_context(cwd) to assemble the system prompt. Channels with their own short system prompt (like matrix) pass FALSE.
validate_api_key	Logical. When TRUE, error if the provider's API key env var is unset or empty.
verbose	Logical. Passed through to new_session.
max_turns	Integer. Passed through to new_session. Defaults to 50, a safety net for interactive channels where a multi-step request (read + edit + verify several files) can easily exceed the new_session() default of 10.

Details

1. Loads project + global corteza config from cwd.
2. Resolves provider, model, and verifies the required API environment variable is set.
3. Registers built-in skills and loads user/project skills and skill docs from tools::R_user_dir("corteza", "data")/skills and <cwd>/.corteza/skills.
4. Loads skill packages declared in the config.
5. Optionally builds the system prompt via load_context(cwd).
6. Returns a new_session() built from the above.

Value

A session environment from new_session, with an extra cwd field set.

Examples

```
## Not run:
# Requires the relevant provider API key in the environment.
s <- session_setup("cli", provider = "anthropic",
                   load_project_context = FALSE)
s$model

## End(Not run)
```

---

skill_install *Install a skill from a path or URL*

Description

Install a skill from a path or URL

Usage

```
skill_install(source, target_dir = NULL, force = FALSE)
```

Arguments

source	Path to skill directory or URL
target_dir	Installation directory. Default is tools::R_user_dir("corteza", "data")/skills.
force	Overwrite if exists

Value

Installed skill name

Examples

```
# Install into a throwaway directory rather than the user's
# R_user_dir, so this example doesn't mutate state.
src <- file.path(tempdir(), "demo_skill")
dir.create(src, showWarnings = FALSE)
writeLines(c(
               "---",
               "name: demo",
               "description: A demo skill",
               "---",
               "Demo body."
           ),
          file.path(src, "SKILL.md"))

dest <- file.path(tempdir(), "skills_lib")
skill_install(src, target_dir = dest)

unlink(src, recursive = TRUE)
unlink(dest, recursive = TRUE)
```

skill_list_installed *List installed skills*

Description

List installed skills

Usage

```
skill_list_installed(skill_dir = NULL)
```

Arguments

skill_dir	Skills directory

Value

Data frame with skill info

Examples

```
# List skills from an empty tempdir; returns a zero-row data frame
# with the documented columns.
empty <- file.path(tempdir(), "empty_skills")
dir.create(empty, showWarnings = FALSE)
skill_list_installed(skill_dir = empty)
unlink(empty, recursive = TRUE)
```

skill_remove *Remove an installed skill*

Description

Remove an installed skill

Usage

```
skill_remove(name, skill_dir = NULL)
```

Arguments

name	Skill name
skill_dir	Skills directory

Value

Invisible TRUE on success

Examples

```
# Install a demo skill into a tempdir, then remove it. The
# installed name is the source directory's basename.
src <- file.path(tempdir(), "demo_skill")
dir.create(src, showWarnings = FALSE)
writeLines(c("---", "name: demo_skill",
             "description: A demo skill", "---"),
           file.path(src, "SKILL.md"))
dest <- file.path(tempdir(), "skills_lib")
name <- skill_install(src, target_dir = dest)
skill_remove(name, skill_dir = dest)

unlink(src, recursive = TRUE)
unlink(dest, recursive = TRUE)
```

---

skill_test *Run skill tests*

Description

Executes test_*.R files in a skill directory.

Usage

```
skill_test(path, verbose = TRUE)
```

Arguments

path	Path to skill directory
verbose	Print test output

Value

List with passed, failed, errors

Examples

```
# A skill directory with no test_*.R files returns a zero-result
# summary rather than erroring.
p <- file.path(tempdir(), "skill_no_tests")
dir.create(p, showWarnings = FALSE)
skill_test(p, verbose = FALSE)
unlink(p, recursive = TRUE)
```

subagent_collect *Collect the result of a previously-fired async subagent query.*

Description

Pairs with ‘subagent_query(..., wait = FALSE)‘. Returns the reply text once the child finishes its turn, or NULL while the query is still running. Result is read exactly once: after a successful collect the pending slot is cleared, so the next async query can fire.

Usage

```
subagent_collect(id, wait = TRUE, timeout = 60L)
```

Arguments

id	Subagent identifier (UUID, prefix, or sequence number).
wait	If TRUE (default), block up to ‘timeout‘ seconds waiting for the child to finish. If FALSE, poll once and return immediately.
timeout	Maximum seconds to block when ‘wait = TRUE‘. On timeout the child is left running; caller may collect again later or kill explicitly.

Value

Reply text (character) when ready; NULL when still running.

Examples

```
## Not run:
id <- subagent_spawn("background research")
subagent_query(id, "what's in DESCRIPTION?", wait = FALSE)
# ... do other work ...
subagent_collect(id, wait = TRUE, timeout = 30)
subagent_kill(id)

## End(Not run)
```

subagent_kill *Kill a subagent.*

Description

Kill a subagent.

Usage

```
subagent_kill(id)
```

Arguments

id	Subagent identifier (UUID, prefix, or sequence number).

Value

Invisible TRUE if killed, FALSE if not found.

Examples

```
# Unknown id is a silent no-op (returns FALSE), so this is safe to
# run during R CMD check without a live subagent.
subagent_kill("no-such-id")
```

---

subagent_list *List active subagents.*

Description

Returns a list of info objects per agent: id/seq/task/started_at/ time_remaining/pending plus model/age/cumulative usage and a best-effort live token count for idle agents (‘NA‘ for busy).

Usage

```
subagent_list()
```

Value

List of subagent info objects.

Examples

```
# Empty when no subagent has been spawned yet -- safe to call any time.
subagent_list()
```

---

subagent_query *Query a subagent.*

Description

Sends a prompt to a running subagent. Inside the child it runs through [turn()] with the child's persistent turn session: the LLM replies, any tool calls it makes resolve against the child's in-process skill registry, and history accumulates across queries.

Usage

```
subagent_query(id, prompt, wait = TRUE, timeout = 60L, return_name = NULL)
```

Arguments

id	Subagent identifier. Accepts the canonical UUID, a unique UUID prefix, or the per-session sequence number printed by ‘subagent_list()‘ / ‘/agents‘.
prompt	Prompt to send.
wait	If TRUE (default), block until the child replies and return the reply text. If FALSE, fire the prompt and return the canonical id invisibly; caller must collect via [subagent_collect()].
timeout	Timeout in seconds (currently advisory; callr-level hard timeouts are future work).
return_name	Optional single name or ‘.h_NNN‘ handle for a value the child should hand back. When set, the child must have left the result bound under that name (e.g. via ‘run_r‘); the resolved value is stashed in the parent handle store and the reply gains a ‘[stored as .h_NNN]‘ block referencing it. Requires a subagent with ‘run_r‘ (the ‘work‘ preset). For ‘wait = FALSE‘ the name is captured now and applied when collected.

Details

With ‘wait = FALSE‘ the call returns immediately after firing the prompt; the parent collects the reply later with [subagent_collect()]. A subagent can only carry one in-flight async query at a time: firing a second one while the first is pending raises an error.

Value

Reply text (character) when ‘wait = TRUE‘, with a handle block appended when ‘return_name‘ resolved. Canonical id (character, invisibly) when ‘wait = FALSE‘.

Examples

```
## Not run:
# Requires LLM credentials in the child's environment.
id <- subagent_spawn("read R/skill.R and summarize", preset = "minimal")
subagent_query(id, "what does this file do?", wait = TRUE)
subagent_kill(id)

## End(Not run)
```

---

subagent_spawn *Spawn a subagent.*

Description

Starts a fresh ‘callr::r_session‘ with corteza loaded and its tool registry set up. Stores the handle in the package-level registry keyed by subagent id.

Usage

```
subagent_spawn(task, model = NULL, tools = NULL, preset = NULL,
               parent_session = NULL, config = NULL)
```

Arguments

task	Task description (stored for bookkeeping; not yet fed into an agent loop).
model	Optional model override (reserved for later use).
tools	Optional explicit tool filter (character vector). Overrides ‘preset‘ when provided. Fixed for the lifetime of the child – cannot be expanded after spawn.
preset	Preset name (fixed for the lifetime of the child). ‘"investigate"‘ (default): ‘read_file‘, ‘grep_files‘, ‘r_help‘, ‘web_search‘, ‘fetch_url‘. ‘"work"‘: investigate + ‘bash‘, ‘write_file‘, ‘replace_in_file‘, ‘list_files‘, ‘git_status‘, ‘git_diff‘, ‘git_log‘, ‘run_r‘. ‘"minimal"‘: ‘read_file‘, ‘grep_files‘.
parent_session	Parent session object; read for nested-spawning control and session-key derivation.
config	Config list.

Details

Permissions: subagents have no interactive approval channel back to the parent or user. The child’s ‘approval_cb‘ denies by default and there is no mid-run escalation path. Whatever capability the child needs must be granted at spawn time through ‘preset‘ or ‘tools‘. If a task may need shell, write, or network capability, pick a preset that includes it (or pass an explicit ‘tools‘ list); otherwise the child should report that it is blocked rather than retry.

Value

Subagent ID (character).

Examples

```
if (interactive()) {
    # Spawns a callr::r_session child loaded with corteza; the
    # registry is in-memory and dies with the parent R session, so
    # we wrap in interactive() to keep R CMD check from leaving
    # children behind.
    id <- subagent_spawn("look up the package version",
                         preset = "minimal")
    subagent_kill(id)
}
```

turn *Run one agent turn*

Description

Sends `prompt` to the configured LLM with tool use enabled. Every tool call the LLM makes is routed through `policy` before being dispatched.

Usage

```
turn(prompt, session, tool_executor = NULL, tools = NULL)
```

Arguments

`prompt`	Character. User prompt.
`session`	A session environment created by `new_session`.
`tool_executor`	Function or NULL. Dispatcher with signature `function(name, args) -> list`. NULL uses the in-process `call_skill` path.
`tools`	List or NULL. Tool schemas to pass the LLM. NULL uses the in-process skill registry (filtered by `session$tools_filter`). Pass explicit schemas when running against a remote skill source.

Details

Tool dispatch is pluggable via `tool_executor`, but the CLI and `chat()` both leave it NULL: tools run in-process through the default `call_skill` dispatcher against the local skill registry. `serve()` is a separate MCP server for external clients only; it is not part of the CLI's tool path. Pass an explicit `function(name, args) -> list` executor only when dispatching tools somewhere other than the in-process registry.

Value

A list with `reply` (character) and `session` (the updated session environment; also mutated in place).

Examples

```
## Not run:
# Requires ANTHROPIC_API_KEY (or the configured provider's key) and
# a network connection to the LLM.
s <- new_session(provider = "anthropic")
out <- turn("Say hello", s)
out$reply

## End(Not run)
```

`uninstall_cli` *Uninstall corteza CLI*

Description

Remove the `corteza` command-line tool.

Usage

```
uninstall_cli(path = NULL)
```

Arguments

`path`	Directory where corteza is installed. Default matches `install_cli()`: `~/bin` on Unix, `tools::R_user_dir("corteza", "data")/bin` on Windows.

Value

TRUE if removed, FALSE if not found, invisibly.

Examples

```
## Not run:
uninstall_cli()

## End(Not run)
```

Index