

Package: strandai (via r-universe)

September 20, 2026

Type Package

Title Client for the Strand AI Platform API

Version 0.10.0

Description R client for the Strand AI Platform 'REST' API. Provides functions to upload Whole Slide Images via 'GCS' resumable upload, submit 'Lattice' H&E to multiplex protein inference jobs, and download 'OME-Zarr' predictions as 'SpatialExperiment' objects from Bioconductor. Designed for spatial-omics research workflows in pharma and academic labs.

License Apache License (>= 2)

Encoding UTF-8

URL <https://strandai.com>, <https://docs.strandai.com>,
<https://github.com/Strand-AI/strand-sdk-r>

BugReports <https://github.com/Strand-AI/strand-sdk-r/issues>

Depends R (>= 4.3)

Imports httr2 (>= 1.0.0), jsonlite (>= 1.8), utils

Suggests SpatialExperiment, SummarizedExperiment, S4Vectors, testthat (>= 3.2.0), webfakes, withr, knitr, rmarkdown

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libssl-dev

Repository <https://strand-ai.r-universe.dev>

Date/Publication 2026-08-21 01:14:43 UTC

RemoteUrl <https://github.com/Strand-AI/strand-sdk-r>

RemoteRef main

RemoteSha 7f39a6980e8c66727f58638fb7400cb3746e1d25

Contents

predict.strand_client	2
strand_client	3
strand_download_results	4
strand_export_download	5
strand_export_get	5
strand_export_request	6
strand_job_cancel	7
strand_job_get	7
strand_job_wait	8
strand_markers_list	8
strand_patch_sample	9
strand_predict	10
strand_restore	11
strand_results_to_array	12
strand_results_to_spatial_experiment	12
strand_run	13
strand_samples_get	14
strand_samples_list	15
strand_segmentation_get	16
strand_segmentation_manifest	17
strand_segmentation_start	17
strand_set_expiration	18
strand_set_expiration_bulk	19
strand_upload_file	20
strand_uploads_get	21
strand_uploads_list	21
Index	23

predict.strand_client *Run a prediction via the stats::predict generic*

Description

Dispatches `predict(client, image_path, markers, ...)` to `strand_run()`, so the standard R generic works on a `strand_client`. See `strand_run()` for the full argument list and return shape — `model =` and `wait =` are both forwarded.

Usage

```
## S3 method for class 'strand_client'
predict(object, image_path, markers, ...)
```

Arguments

object	A strand_client.
image_path	Path to a local WSI file.
markers	Character vector of markers to predict.
...	Additional arguments passed to strand_run() (e.g. model, wait, timeout_sec, output_dir, poll_interval_sec, on_progress).

Value

A strand_predict_result list when wait = TRUE, or a strand_job handle when wait = FALSE; see [strand_run\(\)](#).

strand_client	<i>Create a Strand AI Platform client</i>
---------------	---

Description

Constructs a client object that bundles authentication, base URL, and a shared http2 request template. The returned list is passed to every other function in this package.

Usage

```
strand_client(api_key = NULL, base_url = NULL, timeout = 60, user_agent = NULL)
```

Arguments

api_key	Bearer API key (sk-strand-...). Defaults to the STRAND_API_KEY environment variable.
base_url	Platform base URL (without /api/v1). Defaults to STRAND_BASE_URL env var, then https://app.strandai.com.
timeout	Per-request timeout in seconds.
user_agent	Override the default User-Agent string.

Details

Default base URL is https://app.strandai.com. The /api/v1/ path is appended automatically; pass the root site URL, not the API root.

Value

An object of class strand_client: a list with api_key, base_url, api_root, timeout, and user_agent.

Examples

```
## Not run:
client <- strand_client() # reads STRAND_API_KEY
client <- strand_client(api_key = "sk-strand-...",
                        base_url = "http://localhost:3000")

## End(Not run)
```

strand_download_results

Download job results and convert to a SpatialExperiment

Description

Walks the OME-Zarr result store via the API-key-authenticated proxy at `/api/v1/jobs/{id}/results/files/{path}`, assembles the per-marker arrays into one matrix (rows = markers, columns = pixels), and returns a [SpatialExperiment::SpatialExperiment\(\)](#) object.

Usage

```
strand_download_results(job, path = NULL, level = 0L, markers = NULL)
```

Arguments

job	A strand_job from strand_predict() .
path	Optional output directory. When set, downloads the store to disk and returns the directory path; no SpatialExperiment is built.
level	Pyramid level to read; 0 is full resolution.
markers	Optional character vector restricting which multiscales to load. Defaults to every multiscale except "H&E".

Details

For large slides this materializes a dense $n_markers \times (H * W)$ matrix in memory; pass path to mirror the zarr store to disk instead and read selectively with Rarr.

Value

A SpatialExperiment with assay = "expression" and pixel coordinates in spatialCoords(). When path is supplied, the directory path is returned invisibly.

Examples

```
## Not run:
job <- strand_predict(client, upload$id, c("CD3", "CD8"))
strand_job_wait(job)
spe <- strand_download_results(job)

## End(Not run)
```

strand_export_download	<i>Download a generated result export</i>
------------------------	---

Description

Download a generated result export

Usage

```
strand_export_download(  
  job,  
  format,  
  path,  
  include_segmentation = FALSE,  
  timeout = Inf,  
  poll_interval = 2  
)
```

Arguments

- job A strand_job.
- format One of "ome-zarr", "ome-zarr-zip", or "ome-tiff".
- path Destination file.
- include_segmentation Attach the latest segmentation artifact manifest.
- timeout Maximum seconds to wait; Inf waits indefinitely.
- poll_interval Seconds between status requests.

Value

Normalized destination path, invisibly.

strand_export_get	<i>Get format-driven result export status</i>
-------------------	---

Description

Get format-driven result export status

Usage

```
strand_export_get(job, format, include_he = NULL, include_segmentation = FALSE)
```

Arguments

job	A strand_job.
format	One of "ome-zarr", "ome-zarr-zip", or "ome-tiff".
include_he	Include source H&E. OME-TIFF always includes H&E.
include_segmentation	Attach the latest segmentation artifact manifest.

Value

A refreshed result export manifest.

strand_export_request	<i>Request a format-driven result export</i>
-----------------------	--

Description

Native OME-Zarr is returned without conversion. OME-Zarr ZIP and OME-TIFF are asynchronous, cached exports. The response manifest explicitly records H&E and segmentation/cell-expression contents.

Usage

```
strand_export_request(  
  job,  
  format,  
  include_he = NULL,  
  include_segmentation = FALSE  
)
```

Arguments

job	A strand_job.
format	One of "ome-zarr", "ome-zarr-zip", or "ome-tiff".
include_he	Include source H&E. OME-TIFF always includes H&E.
include_segmentation	Attach the latest segmentation artifact manifest.

Value

A result export manifest.

strand_job_cancel	<i>Cancel an in-flight job</i>
-------------------	--------------------------------

Description

Calls POST /api/v1/jobs/{id}/cancel, then fetches and returns the updated job status. Cancellation refunds the reserved credits. The server returns 400 if the job is already terminal.

Usage

```
strand_job_cancel(job)
```

Arguments

job	A strand_job from <code>strand_predict()</code> .
-----	---

Value

The updated job status list with status = "cancelled".

Examples

```
## Not run:  
job <- strand_predict(client, upload$id, c("CD3", "CD8"))  
status <- strand_job_cancel(job)  
  
## End(Not run)
```

strand_job_get	<i>Fetch the latest status snapshot for a job</i>
----------------	---

Description

Fetch the latest status snapshot for a job

Usage

```
strand_job_get(client, job_id)
```

Arguments

client	A strand_client.
job_id	Job identifier (or a strand_job list).

Value

A list with the job's current state.

strand_job_wait	<i>Block until a job reaches a terminal status</i>
-----------------	--

Description

Polls GET /api/v1/jobs/{id} on a fixed interval and returns the terminal status. (The platform also exposes an SSE stream at /jobs/{id}/stream; the R SDK uses polling for compatibility — there is no widely-used SSE client in base R, and http2 streaming is awkward to integrate with R CMD check.)

Usage

```
strand_job_wait(job, timeout = Inf, poll_interval = 2, progress = FALSE)
```

Arguments

job	A strand_job from strand_predict() .
timeout	Max seconds to wait. Inf to wait forever.
poll_interval	Seconds between status polls.
progress	If TRUE, prints a one-line status update per poll.

Value

The terminal job status list. status = "completed" means every requested marker delivered; "partial_failed" means some markers delivered (results are downloadable) and the rest terminally failed after server-side retries — it returns normally rather than erroring, so check \$status. Only a "failed" terminal status (nothing delivered) raises a strand_job_failed_error.

strand_markers_list	<i>List available markers</i>
---------------------	-------------------------------

Description

List the markers your account can request, scoped to your entitlement: a self-signup account sees the public panel; a full-panel account sees the whole vocabulary. Credit-free. The names returned are exactly what [strand_predict\(\)](#) accepts in both submit and dry-run modes.

Usage

```
strand_markers_list(client)
```

Arguments

client	A strand_client from strand_client() .
--------	--

Value

A named list with `full_panel` (logical; whether the account holds the full-panel entitlement), `markers` (a character vector of marker names), and `public_markers` (the subset that is part of the public panel).

Examples

```
## Not run:
client <- strand_client()
catalog <- strand_markers_list(client)
catalog$markers

## End(Not run)
```

strand_patch_sample	<i>Patch an owned sample</i>
---------------------	------------------------------

Description

Updates any combination of display name, complete user-editable tag set, and physical pixel size in one request. NULL arguments are omitted. Use `clear_name = TRUE` to send JSON null and restore the filename as the display name. Passing `tags = character(0)` clears all user-editable tags.

Usage

```
strand_patch_sample(
  client,
  sample_id,
  name = NULL,
  tags = NULL,
  mpp = NULL,
  clear_name = FALSE
)
```

Arguments

<code>client</code>	A <code>strand_client</code> from strand_client() .
<code>sample_id</code>	UUID of an owned sample.
<code>name</code>	New display name, or NULL to leave it unchanged.
<code>tags</code>	Character vector containing the complete desired tag set, <code>character(0)</code> to clear it, or NULL to leave it unchanged.
<code>mpp</code>	Isotropic microns per pixel, greater than 0 and at most 100, or NULL to leave it unchanged.
<code>clear_name</code>	Send JSON null for name, restoring the filename as the display name. Cannot be combined with a non-NULL name.

Value

The refreshed owned sample record, including jobs and job_count; see [strand_samples_get\(\)](#).

Examples

```
## Not run:
strand_patch_sample(client, sample_id, name = "Baseline biopsy")
strand_patch_sample(client, sample_id, tags = c("baseline", "responding"))
strand_patch_sample(client, sample_id, tags = character(0), mpp = 0.26)
strand_patch_sample(client, sample_id, clear_name = TRUE)

## End(Not run)
```

strand_predict	<i>Price or submit a prediction</i>
----------------	-------------------------------------

Description

Calls POST /api/v1/predict. The default mode atomically reserves credits and creates a job. With dry_run = TRUE, the same marker, model, and sample validation runs without reserving credits or creating a job.

Usage

```
strand_predict(client, upload_id, markers, model = NULL, dry_run = FALSE)
```

Arguments

client	A strand_client.
upload_id	Upload identifier (from strand_upload_file()).
markers	Character vector of marker names (e.g. c("CD3", "CD8")).
model	Optional explicit Lattice version. "v0.7" is the current dispatchable version and default. "v0.4" and "v0.5" remain recognized identifiers for historical records, but requesting either for a new job returns 400 model_sunset. When NULL (default), the platform picks "v0.7".
dry_run	When TRUE, price the request without reserving credits or creating a job. Must be one non-missing logical value.

Value

With dry_run = FALSE, a strand_job list with id, reserved_credits, and client. With dry_run = TRUE, a list with dry_run = TRUE, patch_count, marker_count, estimated_credits, org_balance, and org_pending.

Examples

```
## Not run:
estimate <- strand_predict(
  client, upload$id, c("CD3", "CD8"), dry_run = TRUE
)
message("approximately ", estimate$estimated_credits, " credits")

job <- strand_predict(client, upload$id, c("CD3", "CD8"))
strand_job_wait(job)

job <- strand_predict(client, upload$id, c("CD3", "CD8"),
  model = "v0.7")

## End(Not run)
```

strand_restore	<i>Restore a sample from Trash</i>
----------------	------------------------------------

Description

Available within the 7-day Trash window. Brings the sample back to the active list and extends its expiration so it isn't immediately re-trashed. Caller must have the same permissions required for [strand_set_expiration\(\)](#).

Usage

```
strand_restore(client, sample_id)
```

Arguments

client	A strand_client from strand_client() .
sample_id	UUID of the sample.

Value

The restored sample payload (id, archivedAt, expiresAt).

Examples

```
## Not run:
client <- strand_client()
strand_restore(client, sample_id)

## End(Not run)
```

strand_results_to_array

Download a single multiscale as a raw [C, H, W] numeric array

Description

Download a single multiscale as a raw [C, H, W] numeric array

Usage

```
strand_results_to_array(job, name = NULL, level = 0L, root_meta = NULL)
```

Arguments

job	A strand_job from strand_predict() .
name	Name of the multiscale to read (e.g. "CD3"). When omitted, defaults to the first marker multiscale (excluding H&E).
level	Pyramid level to read; 0 is full resolution.
root_meta	Optional pre-fetched root group metadata (internal use).

Value

A list with array (3-D [C, H, W]) and meta (parsed zarr.json).

strand_results_to_spatial_experiment

Build a SpatialExperiment from a [C, H, W] array

Description

Build a SpatialExperiment from a [C, H, W] array

Usage

```
strand_results_to_spatial_experiment(array, markers, base_path = NA_character_)
```

Arguments

array	3-D array [C, H, W].
markers	Character vector of marker / channel names; length equals C.
base_path	Optional GCS path; stored in metadata(spe)\$strand\$base_path.

Value

A SpatialExperiment with assay = "expression" and pixel coords in spatialCoords.

strand_run

*Run the full prediction pipeline in one call***Description**

Orchestrates upload → submit → (optional) wait → (optional) download. With wait = TRUE (default) returns when the job reaches a terminal state. With wait = FALSE, returns the strand_job handle as soon as upload + submit complete so the caller can drive strand_job_wait() / strand_download_results() later. All sub-operations use the same exported primitives ([strand_upload_file\(\)](#), [strand_predict\(\)](#), [strand_job_wait\(\)](#), [strand_download_results\(\)](#)) so callers can drop down a level whenever they need finer control.

Usage

```
strand_run(
  client,
  image_path,
  markers,
  model = NULL,
  wait = TRUE,
  timeout_sec = 1800,
  output_dir = NULL,
  poll_interval_sec = 5,
  on_progress = NULL,
  auto_segment = NULL
)
```

Arguments

client	A strand_client.
image_path	Path to a local WSI file (SVS / TIFF / NDPI / ...).
markers	Character vector of markers to predict (e.g. c("HER2", "CD8", "PD1")).
model	Optional explicit model id; see strand_predict() . Defaults to NULL (platform picks).
wait	When TRUE (default), block through upload → submit → wait → download and return a strand_predict_result. When FALSE, return the strand_job handle once upload + submit complete. timeout_sec, poll_interval_sec, output_dir, and the "wait" / "download" progress stages are ignored when wait = FALSE.
timeout_sec	Max seconds to wait for the job to finish; defaults to 30 minutes.
output_dir	Optional output directory. When provided, the entire OME-Zarr result store is mirrored under this directory.
poll_interval_sec	Seconds between status polls while waiting.

on_progress	Optional callback function(stage, fraction) where stage is one of "upload", "submit", "wait", "download" (only "upload" and "submit" fire when wait = FALSE). fraction is always a numeric in [0, 1] — 0 at the start of each stage and 1 at its end, with intermediate values where the underlying step exposes progress.
auto_segment	Opt out of automatic cell segmentation for the uploaded slide. NULL (default) uses the org default; FALSE skips segmentation; TRUE forces it on. Passed through to strand_upload_file() .

Value

When wait = TRUE, a list with class strand_predict_result containing: job_id, status, credits_used, marker_outputs (named list mapping marker name → on-disk directory when output_dir is provided, else an empty list), output_dir, and job (the underlying strand_job handle). When wait = FALSE, the strand_job handle directly (same shape as [strand_predict\(\)](#)).

Examples

```
## Not run:
client <- strand_client()
result <- strand_run(
  client, "biopsy.ome.tiff",
  markers = c("HER2", "CD8", "PD1"),
  output_dir = "./outputs/"
)
cat("used", result$credits_used, "credits\n")

# Choose a version explicitly:
result <- strand_run(client, "biopsy.ome.tiff", c("CD8", "Ki67"),
  model = "v0.7")

# Fire-and-forget – upload + submit only, then drive the job later:
job <- strand_run(client, "biopsy.ome.tiff", c("CD8"), wait = FALSE)
status <- strand_job_wait(job)
spe <- strand_download_results(job)

# The S3 generic also works: predict(client, ...)
result <- predict(client, "biopsy.ome.tiff", markers = c("HER2"))

## End(Not run)
```

strand_samples_get	<i>Get a sample</i>
--------------------	---------------------

Description

Fetches either an owned sample by sample id or a public sample by share id. The mandatory ownership field identifies the returned branch. Owned samples include up to the 50 newest inference jobs and the uncapped job_count; public samples never expose job history.

Usage

```
strand_samples_get(client, sample_id)
```

Arguments

client	A strand_client from <code>strand_client()</code> .
sample_id	Owned sample UUID or public share UUID.

Value

For an owned sample, a named list with ownership = "mine", sample identity and lifecycle fields, jobs, and job_count. Each job contains id, status, progress, reserved_credits, markers, timestamps, error_message, and results_available. For a public sample, a named list with ownership = "public", canonical share id, title, tags, public metadata, geometry, markers, thumbnail_url, and pyramid_url.

Examples

```
## Not run:
client <- strand_client()
sample <- strand_samples_get(client, sample_id)
if (sample$ownership == "mine") {
  message(sample$job_count, " inference jobs")
} else {
  print(sample$markers)
}

## End(Not run)
```

strand_samples_list	<i>List samples</i>
---------------------	---------------------

Description

Lists owned samples, Strand's public cohort, or both through one cursor-paginated collection. Public and owned records use distinct shapes; inspect each item's mandatory ownership field before using branch fields.

Usage

```
strand_samples_list(
  client,
  scope = "mine",
  limit = 48L,
  cursor = NULL,
  tag = NULL
)
```

Arguments

client	A strand_client from <code>strand_client()</code> .
scope	Exactly one of "mine", "public", or "all". The default lists owned samples. Values are case-sensitive and are never partially matched.
limit	Items per page, from 1 through 100.
cursor	Opaque cursor returned as next_cursor by the previous call, or NULL for the first page.
tag	Optional exact tag filter.

Value

A named list with items and next_cursor. Every item has ownership and canonical id. Owned items also have name, filename, status, numeric file_size, tags, and created_at. Public items also have title, thumbnail_url, tags, and public metadata.

Examples

```
## Not run:
client <- strand_client()
page <- strand_samples_list(client, scope = "public", tag = "tcga-coad")
ids <- vapply(page$items, function(sample) sample$id, character(1))

## End(Not run)
```

strand_segmentation_get

Get cell-segmentation state

Description

Get cell-segmentation state

Usage

```
strand_segmentation_get(client, sample_id)
```

Arguments

client	A strand_client.
sample_id	An owned sample UUID. Public sample handles are read-only.

Value

Lifecycle state with latest job/layer metadata.

strand_segmentation_manifest
<i>Get segmentation artifact manifest</i>

Description

Get segmentation artifact manifest

Usage

strand_segmentation_manifest(client, sample_id, layer_id)

Arguments

- | | |
|-----------|--|
| client | A strand_client. |
| sample_id | An owned sample UUID. Public sample handles are read-only. |
| layer_id | Segmentation layer UUID from the lifecycle response. |

Value

Explicit mask, geometry/morphology, marker-expression, and provenance schema.

strand_segmentation_start
<i>Start or retry cell segmentation</i>

Description

Free and idempotent: completed or active work is reused, failed work can be retried after the documented cooldown, and no credits are reserved.

Usage

strand_segmentation_start(client, sample_id)

Arguments

- | | |
|-----------|--|
| client | A strand_client. |
| sample_id | An owned sample UUID. Public sample handles are read-only. |

Value

Lifecycle state and polling URL.

strand_set_expiration *Set expiration on a single sample*

Description

Pin a sample to a specific expiration date, set it to never expire, or clear any custom expiration and follow the org's default policy.

Usage

```
strand_set_expiration(
  client,
  sample_id,
  expires_at = NULL,
  never_expire = FALSE,
  use_org_default = FALSE,
  reason = NULL
)
```

Arguments

client	A strand_client from strand_client() .
sample_id	UUID of the sample.
expires_at	A POSIXct (or ISO 8601 character) expiration date.
never_expire	If TRUE, set the sample to never expire.
use_org_default	If TRUE, clear any custom expiration and follow the org's current expiration policy.
reason	Optional governance reason (10-500 chars).

Value

The updated sample payload (id, expiresAt, expiresAtSource, ...).

Examples

```
## Not run:
client <- strand_client()
strand_set_expiration(client, sample_id, never_expire = TRUE,
  reason = "Active research project - keep until publication")

## End(Not run)
```

strand_set_expiration_bulk

Set expiration on a batch of samples (max 500)

Description

All-or-nothing: if any sample fails the permission gate (caller is not the sample creator, an org owner/admin, or a Strand AI admin), the whole call returns a 403 with no rows touched.

Usage

```
strand_set_expiration_bulk(
  client,
  sample_ids,
  expires_at = NULL,
  never_expire = FALSE,
  use_org_default = FALSE,
  reason = NULL
)
```

Arguments

client	A strand_client from strand_client() .
sample_ids	Character vector of sample UUIDs (length 1-500).
expires_at	A POSIXct (or ISO 8601 character) expiration date.
never_expire	If TRUE, set the sample to never expire.
use_org_default	If TRUE, clear any custom expiration and follow the org's current expiration policy.
reason	Optional governance reason (10-500 chars).

Value

A list with updated (integer count) and batchId (UUID).

Examples

```
## Not run:
client <- strand_client()
strand_set_expiration_bulk(client, c(uuid1, uuid2, uuid3), never_expire = TRUE)

## End(Not run)
```

strand_upload_file *Upload a Whole Slide Image to the Strand AI Platform*

Description

Performs the event-driven resumable upload flow:

1. POST /api/v1/uploads — create a resumable session
2. PUT <uploadUrl> — stream the file to GCS in 8 MiB chunks
3. Poll GET /api/v1/uploads/{id} until GCS OBJECT_FINALIZE starts ingest

Usage

```
strand_upload_file(
    client,
    path,
    content_type = NULL,
    chunk_size = 8L * 1024L * 1024L,
    progress = FALSE,
    auto_segment = NULL,
    mpp = NULL
)
```

Arguments

client	A strand_client from strand_client() .
path	Path to a local file (SVS / TIFF / NDPI / ...).
content_type	Override the auto-detected MIME type.
chunk_size	Bytes per chunk. Must be a positive multiple of 256 KiB (the GCS resumable upload requirement). Defaults to 8 MiB.
progress	If TRUE, prints upload progress to the console.
auto_segment	Opt out of automatic cell segmentation for this upload. NULL (default) uses the org's default; FALSE skips segmentation (the slide is still ingested and rendered); TRUE forces it on even when the org default is off.
mpp	Optional user-reported microns per pixel, persisted when the upload is created. Must be greater than 0 and at most 100. This creation-time value takes precedence over embedded slide calibration.

Value

A list with class strand_upload containing id, gcs_path, upload_url, width_px, height_px, status.

Examples

```
## Not run:
client <- strand_client()
upload <- strand_upload_file(client, "slide.svs", progress = TRUE, mpp = 0.26)

## End(Not run)
```

strand_uploads_get	<i>Fetch a single upload by id</i>
--------------------	------------------------------------

Description

Fetch a single upload by id

Usage

```
strand_uploads_get(client, upload_id)
```

Arguments

client	A strand_client.
upload_id	Upload identifier (UUID).

Value

A strand_upload list as documented in [strand_uploads_list\(\)](#). Raises strand_not_found_error if no upload with that id exists for the calling org.

Examples

```
## Not run:
u <- strand_uploads_get(client, "11111111-1111-1111-1111-111111111111")

## End(Not run)
```

strand_uploads_list	<i>List uploads for the calling organization</i>
---------------------	--

Description

Fetches one page of uploads, newest-first. Filters out archived uploads; use [strand_restore\(\)](#) to bring archived rows back into the active set.

Usage

```
strand_uploads_list(client, limit = 100L, cursor = NULL)
```

Arguments

client	A strand_client.
limit	Page size (1-200). Defaults to 100.
cursor	Opaque cursor from a prior response's next_cursor. NULL (default) fetches the first page.

Value

A list with class strand_upload_list containing:

- uploads: list of strand_upload entries (each carries id, filename, file_size, status, gcs_path, width_px, height_px, created_at).
- next_cursor: cursor for the next page, or NULL on the last page.

Examples

```
## Not run:
page <- strand_uploads_list(client, limit = 20)
for (u in page$uploads) message(u$id, " ", u$filename)

## End(Not run)
```

Index

`predict.strand_client`, 2

`SpatialExperiment::SpatialExperiment()`,
4

`strand_client`, 3

`strand_client()`, 8, 9, 11, 15, 16, 18–20

`strand_download_results`, 4

`strand_download_results()`, 13

`strand_export_download`, 5

`strand_export_get`, 5

`strand_export_request`, 6

`strand_job_cancel`, 7

`strand_job_get`, 7

`strand_job_wait`, 8

`strand_job_wait()`, 13

`strand_markers_list`, 8

`strand_patch_sample`, 9

`strand_predict`, 10

`strand_predict()`, 4, 7, 8, 12–14

`strand_restore`, 11

`strand_restore()`, 21

`strand_results_to_array`, 12

`strand_results_to_spatial_experiment`,
12

`strand_run`, 13

`strand_run()`, 2, 3

`strand_samples_get`, 14

`strand_samples_get()`, 10

`strand_samples_list`, 15

`strand_segmentation_get`, 16

`strand_segmentation_manifest`, 17

`strand_segmentation_start`, 17

`strand_set_expiration`, 18

`strand_set_expiration()`, 11

`strand_set_expiration_bulk`, 19

`strand_upload_file`, 20

`strand_upload_file()`, 10, 13, 14

`strand_uploads_get`, 21

`strand_uploads_list`, 21

`strand_uploads_list()`, 21