Package: livetennisapi (via r-universe)

November 9, 2026

Title Client for the 'Live Tennis API'

Version 0.2.0

Description Read live and historical tennis data from the 'Live Tennis API' <https://livetennisapi.com> into R. Covers live scores, fixtures, players, tournaments, rankings, head-to-head records and the 1968 to 2022 results archive, for the ATP, WTA, Challenger and ITF tours. List endpoints return tibbles.

License MIT + file LICENSE

URL https://livetennisapi.com, https://docs.livetennisapi.com, https://github.com/livetennisapi/livetennisapi-r

BugReports https://github.com/livetennisapi/livetennisapi-r/issues

Depends R (>= 4.1)

Imports httr2 (>= 1.1.0), stats, tibble (>= 3.0.0), utils

Suggests jsonlite, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libssl-dev

Repository https://livetennisapi.r-universe.dev

Date/Publication 2026-11-09 00:09:16 UTC

RemoteUrl https://github.com/livetennisapi/livetennisapi-r

RemoteRef HEAD

RemoteSha 25694dea159417fad55903a721149f10d8591abc

Contents

api_health . . . 2
archive_matches . . . 3
get_match . . . 4
head_to_head . . . 5
list_fixtures . . . 5
list_matches . . . 6
list_tournaments . . . 7
lta_get . . . 8
lta_key . . . 9
lta_meta . . . 10
rankings . . . 10
search_players . . . 12

Index **14**

`api_health` *Check the API and your plan*

Description

`api_health()` needs no key and says whether the API is up. `api_usage()` reports your plan, your limits and what you have used today.

Usage

```
api_health()

api_usage(key = NULL)
```

Arguments

`key`	The API key. Defaults to `lta_key()`.

Value

A list.

Examples

```
api_health()
if (nzchar(Sys.getenv("LIVETENNISAPI_KEY"))) api_usage()$tier
```

`archive_matches` *The results archive, 1968 to 2022*

Description

`archive_matches()` searches past results. `archive_players()` returns player bios. `archive_career()` sums one player's career over the archive. These need a paid plan, from Basic up.

Usage

```
archive_matches(
  tour = NULL,
  name = NULL,
  from = NULL,
  to = NULL,
  round = NULL,
  level = NULL,
  limit = NULL,
  offset = NULL,
  ...,
  key = NULL
)

archive_players(
  name = NULL,
  tour = NULL,
  limit = NULL,
  offset = NULL,
  ...,
  key = NULL
)

archive_career(name, key = NULL)
```

Arguments

`tour`	`"atp"` or `"wta"`.
`name`	A player name, or part of one (at least three characters for `archive_career()`).
`from`, `to`	Play dates, as `Date` values or `"YYYY-MM-DD"` strings. In the results archive they are tournament start dates.
`round`	A round code such as `"F"` or `"QF"`.
`level`	A level code such as `"G"` for a Grand Slam.
`limit`	Rows per page.
`offset`	Rows to skip, for paging.
`...`	Other query parameters the endpoint accepts.
`key`	The API key. Defaults to `lta_key()`.

Value

`archive_matches()` and `archive_players()` return tibbles. `archive_career()` returns a list.

Examples

```
archive_matches(tour = "atp", name = "federer", round = "F", level = "G")
```

`get_match` *Read one match*

Description

`get_match()` returns the full match. `match_score()` returns only the current score, which is the fastest read while a match is live.

Usage

```
get_match(match_id, key = NULL)

match_score(match_id, ..., key = NULL)
```

Arguments

`match_id`	The match id, from `list_matches()`.
`key`	The API key. Defaults to `lta_key()`.
`...`	Other query parameters, such as `sequence` for `match_score()`.

Value

A list.

Examples

```
m <- live_matches()
if (nrow(m) > 0) match_score(m$id[1])
```

head_to_head *Head-to-head record*

Description

Every meeting between two players across the results archive and recent matches, with totals and a split by surface.

Usage

```
head_to_head(p1, p2, key = NULL)
```

Arguments

p1, p2	Player names, or parts of them (at least three characters).
key	The API key. Defaults to `lta_key()`.

Value

A list with `totals`, `by_surface` and `meetings`.

Examples

```
h2h <- head_to_head("djokovic", "nadal")
h2h$totals
```

list_fixtures *Upcoming fixtures*

Description

Scheduled matches, earliest first.

Usage

```
list_fixtures(
  tour = NULL,
  draw = NULL,
  limit = NULL,
  offset = NULL,
  ...,
  key = NULL
)
```

Arguments

tour	One of "atp", "wta", "challenger", "itf" or "juniors".
draw	"singles" or "doubles".
limit	Rows per page.
offset	Rows to skip. To read a whole filtered set, add limit to offset until lta_meta(x)$has_more is FALSE.
...	Other query parameters the endpoint accepts.
key	The API key. Defaults to lta_key().

Value

A tibble, one row per fixture.

Examples

```
list_fixtures(tour = "wta", limit = 20)
```

---

list_matches *List matches*

Description

list_matches() reads matches by lifecycle status. live_matches() is the same call with status = "live". Live and upcoming matches are on the free plan. Completed results need a paid plan.

Usage

```
list_matches(
  status = "live",
  tour = NULL,
  draw = NULL,
  player = NULL,
  tier = NULL,
  from = NULL,
  to = NULL,
  limit = NULL,
  offset = NULL,
  updated_since = NULL,
  cursor = NULL,
  ...,
  key = NULL
)

live_matches(tour = NULL, ..., key = NULL)
```

Arguments

status	One of "live", "upcoming", "completed" or "cancelled".
tour	One of "atp", "wta", "challenger", "itf" or "juniors".
draw	"singles" or "doubles".
player	One or more player ids. A match matches when either side is one of them.
tier	One or more tier codes, such as "atp_500". See https://docs.livetennisapi.com/reference.html for the list.
from, to	Play dates, as Date values or "YYYY-MM-DD" strings. In the results archive they are tournament start dates.
limit	Rows per page.
offset	Rows to skip. To read a whole filtered set, add limit to offset until lta_meta(x)$has_more is FALSE.
updated_since	Read the change feed: rows changed at or after this time, oldest first. Page it with cursor.
cursor	lta_meta(x)$next_cursor from the previous change-feed page.
...	Other query parameters the endpoint accepts.
key	The API key. Defaults to lta_key().

Value

A tibble, one row per match. Read its paging details with lta_meta().

Examples

```
live <- live_matches(tour = "atp")
if (nrow(live) > 0) live[, c("id", "tournament", "players.p1.name", "players.p2.name")]
```

---

list_tournaments *Tournaments*

Description

list_tournaments() searches the tournament catalogue. get_tournament() returns one tournament by its stable id.

Usage

```
list_tournaments(
  search = NULL,
  tour = NULL,
  draw = NULL,
  tier = NULL,
  limit = NULL,
```

```
  offset = NULL,
  ...,
  key = NULL
)

get_tournament(tournament_id, key = NULL)
```

Arguments

search	A name or part of one. Leave it out to list ranked players first.
tour	One of "atp", "wta", "challenger", "itf" or "juniors".
draw	"singles" or "doubles".
tier	One or more tier codes, such as "atp_500". See https://docs.livetennisapi.com/reference.html for the list.
limit	Rows per page.
offset	Rows to skip. To read a whole filtered set, add limit to offset until lta_meta(x)$has_more is FALSE.
...	Other query parameters the endpoint accepts.
key	The API key. Defaults to lta_key().
tournament_id	The tournament id.

Value

list_tournaments() returns a tibble, get_tournament() a list.

Examples

```
list_tournaments(search = "open", tier = "grand_slam")
```

---

lta_get *Call any Live Tennis API endpoint*

Description

The other functions in this package are thin wrappers around this one. Use it for an endpoint that has no wrapper yet. The full list is in the reference at https://docs.livetennisapi.com/reference.html.

Usage

```
lta_get(path, query = list(), key = NULL, auth = TRUE, simplify = FALSE)
```

Arguments

path	The endpoint path, for example "/matches".
query	A named list of query parameters. NULL values are dropped.
key	The API key. Defaults to lta_key().
auth	Send the key. Only /health works without one.
simplify	Simplify JSON arrays into vectors and data frames.

Details

Failures raise a condition of class lta_error, with a subclass you can catch: lta_auth_error (401), lta_plan_error (403: the endpoint needs a higher plan), lta_rate_limited (429, with lta_abuse_throttled for the 24-hour block), lta_not_found (404), lta_bad_request (400), lta_gone (410: the record was merged into another, and the body says which), lta_connection_error (the API could not be reached) or lta_http_error (anything else, such as a 5xx after the retries). The condition carries status, code (the API's machine-readable error), body and retry_after.

A call is tried up to three times, each with a 30-second timeout. Dropped connections, timeouts, server errors (5xx) and the per-minute 429 are retried, waiting what Retry-After asks for (at most a minute) or a short backoff. The daily-quota 429 and abuse_throttled are raised at once, because neither clears in seconds.

Value

The parsed JSON body as a list.

Examples

```
lta_get("/matches", list(status = "live", tour = "atp"))
```

---

lta_key *Read the API key*

Description

The key comes from the LIVETENNISAPI_KEY environment variable. Get a free key, with no card, at https://livetennisapi.com/subscribe/free.

Usage

```
lta_key()
```

Value

The key as a string, or an error when it is not set.

Examples

```
# TRUE when a key is set. The key itself is never printed.
nzchar(lta_key())
```

`lta_meta` *Paging details of a list result*

Description

Every list function returns a tibble and keeps the response's `meta` object as an attribute. `lta_meta()` reads it back. It holds `limit`, `offset`, `count`, `total` (which can be `NULL`), `has_more` and, on the change feed, `next_cursor`. Subsetting a tibble drops the attribute, so read it from the result as it came back.

Usage

```
lta_meta(x)
```

Arguments

`x`	A tibble returned by a list function, such as `live_matches()`.

Value

A list, or `NULL` when `x` carries no paging details.

Examples

```
fx <- list_fixtures(limit = 10)
lta_meta(fx)$has_more
```

`rankings` *Rankings and Elo ratings*

Description

With no `player`, returns the rank-ordered table for one official `system`. With `player`, returns the ranking records those players held at `as_of`. Elo ratings (`system = "elo"` with a `tour`) come in four ladders: `"overall"`, `"hard"`, `"clay"` and `"grass"`. The plan each mode needs is on the pricing page, https://livetennisapi.com/pricing.

Usage

```
rankings(
  system = NULL,
  tour = NULL,
  surface = NULL,
  player = NULL,
  as_of = NULL,
  limit = NULL,
  offset = NULL,
  ...,
  key = NULL
)
```

Arguments

system	One or more of "atp", "wta", "atp_doubles", "wta_doubles", "itf_mt", "itf_wt", "itf_jt", "utr" or "elo".
tour	"atp" or "wta".
surface	The Elo ladder.
player	One or more player ids.
as_of	The date the records must be in force on, as a Date or a "YYYY-MM-DD" string. The API's default is today.
limit	Rows per page.
offset	Rows to skip. To read a whole filtered set, add limit to offset until lta_meta(x)$has_more is FALSE.
...	Other query parameters the endpoint accepts.
key	The API key. Defaults to lta_key().

Value

A tibble, one row per ranking record.

Examples

```
rankings(system = "atp", limit = 100)
rankings(system = "elo", tour = "wta", surface = "clay")
```

search_players *Players*

Description

`search_players()` finds players by name. `get_player()` returns one player's profile and ranking. `player_stats()` returns a player's record, with filters for dates, surface and level, and needs a paid plan.

Usage

```
search_players(search = NULL, limit = NULL, offset = NULL, ..., key = NULL)

get_player(player_id, key = NULL)

player_stats(
  player_id,
  from = NULL,
  to = NULL,
  surface = NULL,
  level = NULL,
  last = NULL,
  ...,
  key = NULL
)
```

Arguments

`search`	A name or part of one. Leave it out to list ranked players first.
`limit`	Rows per page.
`offset`	Rows to skip. To read a whole filtered set, add `limit` to `offset` until `lta_meta(x)$has_more` is `FALSE`.
`...`	Other query parameters the endpoint accepts.
`key`	The API key. Defaults to `lta_key()`.
`player_id`	The player id.
`from, to`	Play dates, as `Date` values or `"YYYY-MM-DD"` strings. In the results archive they are tournament start dates.
`surface`	`"hard"`, `"clay"`, `"grass"` or `"indoor"`.
`level`	A tour code, as in `list_matches()`.
`last`	Only the most recent matches, this many.

Value

`search_players()` returns a tibble. The others return lists.

Examples

```
p <- search_players("alcaraz")
if (nrow(p) > 0) get_player(p$id[1])
```

Index

api_health, 2
api_usage *(*api_health*)*, 2
archive_career *(*archive_matches*)*, 3
archive_matches, 3
archive_players *(*archive_matches*)*, 3

get_match, 4
get_player *(*search_players*)*, 12
get_tournament *(*list_tournaments*)*, 7

head_to_head, 5

list_fixtures, 5
list_matches, 6
list_matches(), *4*, *12*
list_tournaments, 7
live_matches *(*list_matches*)*, 6
live_matches(), *10*
lta_get, 8
lta_key, 9
lta_key(), *2–9*, *11*, *12*
lta_meta, 10
lta_meta(), *7*

match_score *(*get_match*)*, 4

player_stats *(*search_players*)*, 12

rankings, 10

search_players, 12