Package 'hawkinR'

July 21, 2026

Title Interface to the 'Hawkin Dynamics' Force Platform API

Version 2.0.1

Description Provides a secure and configurable interface to the 'Hawkin Dynamics' API for accessing athlete performance data, tests, and metadata.
The package supports profile-based authentication with secure credential storage via the operating system keychain, automatic access token refresh, and region-aware API routing.
Designed for reproducible analysis, data synchronization workflows, and production deployment.

URL https://connect.hawkindynamics.com/r,
https://github.com/HawkinDynamics/hawkinR,
https://hawkindynamics.github.io/hawkinR/

BugReports https://github.com/HawkinDynamics/hawkinR/issues

Depends R (>= 4.1.0)

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports S7, dplyr (>= 0.7.4), jsonlite (>= 1.8.7), lubridate (>= 1.9.2), magrittr (>= 2.0.3), rlang (>= 1.1.1), keyring, janitor (>= 2.2.0), stats, httr2 (>= 1.0.1), logger (>= 0.3.0), stringr (>= 1.4.0), tidyselect (>= 1.2.1), progress

Suggests knitr, rmarkdown, testthat (>= 3.0.0), mockery, withr, crayon, httpuv, arrow

Config/testthat/edition 3

VignetteBuilder knitr

LazyData true

Collate 'MetricDictionary.R' 'utils.R' 'auth_system.R' 'create_athletes.R' 'get_athletes.R' 'get_cop.R' 'get_forcetime.R' 'get_tests.R' 'get_forcetime_bulk.R' 'get_groups.R' 'get_metrics.R' 'get_tags.R' 'get_teams.R' 'get_testTypes.R' 'logController.R' 'update_athletes.R' 'utils-pipe.R' 'zzz.R'

NeedsCompilation no

Author Lauren Green [aut, cre]

Maintainer Lauren Green <lauren@hawkindynamics.com>

Repository CRAN

Date/Publication 2026-07-21 07:20:07 UTC

Contents

`auth_system` *Hawkin Dynamics Authentication System*

Description

Defines the S7 classes and methods for managing authentication state, secure credential storage, and session configuration.

`create_athletes` *Create Athletes*

Description

Create a new athlete or athletes for an account. Bulk create up to 500 athletes at a time.

Usage

```
create_athletes(athleteData, ...)
```

Arguments

`athleteData`	A data frame of the athletes to be created. The data frame must follow the schema:
`...`	Optional arguments. • `profile`: A `HawkinAuth` object. If not provided, the active connection is used.

Details

The data frame passed as the argument must use the following schema:

Column Name	Type	Inclusion	Description
name	*chr*	**REQUIRED**	athlete's given name (First Last)
image	*chr*	*optional*	URL path to image. `default = null`
active	*logi*	*optional*	athlete is active (TRUE). `default = null`
teams	*list*	*optional*	a single team id as a string or list of team ids. `default = [defaultTeamId]`
groups	*list*	*optional*	a single group id as a string or list of group ids. `default = []`
external property	*chr*	*optional*	External properties can be added by adding any additional columns of equal len

Value

If successful, a confirmation message will be printed with the number of successful athletes created. If there are failures, a data frame containing the athletes that failed to be created will be returned with columns:

Column Name	Type	Description
reason	*chr*	Reason for failed creation
name	*chr*	Athlete's given name (First Last)

Examples

```
## Not run:
# Example data frame following the required schema
df <- data.frame(
  name = c("John Doe", "Jane Smith"),
  image = c("https://example.com/johndoe.jpg", "https://example.com/janesmith.jpg"),
  active = c(TRUE, FALSE),
  teams = I(list("team1", c("team2", "team3"))),
  groups = I(list(NULL, "group1")),
  external_property = c("value1", "value2")
)

# Create athletes using the example data frame
create_athletes(athleteData = df)

## End(Not run)
```

---

get_athletes *Get Athletes*

Description

Get the athletes for an account. Inactive players will only be included if `includeInactive` parameter is set to TRUE.

Usage

```
get_athletes(includeInactive = FALSE, ...)
```

Arguments

`includeInactive`	FALSE by default to exclude inactive players in database. Set to TRUE if you want inactive players included in the return.
`...`	Optional arguments. • `profile`: A `HawkinAuth` object. If not provided, the active connection is used.

Value

Response will be a data frame containing the athletes that match this query. Each athlete includes the following variables:

Column Name	Type	Description
id	*chr*	athlete's unique ID
name	*chr*	athlete's given name (First Last)

active	*bool*	athlete is active (TRUE)
teams	*chr*	team ids separated by ","
groups	*chr*	group ids separated by ","
image	*chr*	URL to the athlete's profile image. NA when never set or cleared.
position	*chr*	Free-text playing position (e.g. "Forward"). NA when blank.
dob	*chr*	Date of birth as an ISO-8601 date string (YYYY-MM-DD). NA when blank.
sport	*chr*	Free-text sport name (e.g. "Basketball"). NA when blank.
height	*num*	Athlete height in centimeters when present. NA otherwise.
lastTestedOn	*num*	Unix epoch **seconds** of the athlete's most recent test session. NA when no tests on file.
external	*chr*	external properties will have a column of their name with the appropriate values for the athlete of NA

The optional profile columns (image, position, dob, sport, height, lastTestedOn) and any external property columns only appear when at least one athlete in the response has that field populated.

Examples

```
## Not run:
# This is an example of how the function would be called. If you only wish to call active players,
# you don't need to provide any parameters.

df_athletes <- get_athletes()

# If you want to include all athletes, including inactive athletes, include the optional
# `includeInactive` parameter.

df_wInactive <- get_athletes( includeInactive = TRUE)


## End(Not run)
```

`get_cop` *Get Center-of-Pressure Data*

Description

Retrieves the raw Center of Pressure (COP) time-series data for a specific test trial ID. COP data is **exclusive to the "Free Run" test type** — any other test type returns a 404.

Usage

```
get_cop(testId, ...)
```

Arguments

`testId`	character. The unique identifier for the test trial.
`...`	Optional arguments. • `profile`: A `HawkinAuth` object. If not provided, the active connection is used.

Value

A HawkinCOP object. The data data frame contains the columns time_s, cop_x, cop_y, left_cop_x, left_cop_y, right_cop_x, and right_cop_y. COP values are in millimeters relative to plate center and may be NA for samples where no weight is on a given plate.

---

get_forcetime *Get Force-Time Data*

Description

Retrieves the raw force-time data for a specific test trial ID.

Usage

```
get_forcetime(testId, ...)
```

Arguments

testId	character. The unique identifier for the test trial.
...	Optional arguments. • profile: A HawkinAuth object. If not provided, the active connection is used.

Value

A HawkinForceTime object.

---

get_forcetime_bulk *Get Raw Force-Time Data (Bulk)*

Description

Retrieves raw force-time data for multiple tests. This function acts as a wrapper for both get_tests and get_forcetime.

Usage

```
get_forcetime_bulk(
  test_ids = NULL,
  export = FALSE,
  export_dir = NULL,
  format = c("csv", "json", "rds", "rda", "tsv", "parquet"),
  file_naming = c("test_id"),
  deidentify = FALSE,
  ...
)
```

Arguments

test_ids	Optional. A character vector of specific Test IDs, or a data frame containing an id column (e.g., the output of get_tests()). If NULL, get_tests(...) is called to retrieve targets.
export	logical. If TRUE, data is written to files in export_dir.
export_dir	character. The directory path to save exported files.
format	character. Options: "csv", "tsv", "json", "rds", "parquet".
file_naming	character vector. Properties to construct the filename.
deidentify	logical. If TRUE, replaces athlete_name with "De-identified".
...	Additional arguments passed to get_tests (e.g., from, typeId). Also accepts profile (HawkinAuth object).

Value

When export = FALSE (the default), a named list of raw force-time results, one element per requested test. When export = TRUE, a character vector of the file paths written to export_dir, returned invisibly. Returns NULL if no matching tests are found.

---

get_groups *Get Groups*

Description

Get the group names and ids for all the groups in the org.

Usage

```
get_groups(...)
```

Arguments

...	Optional arguments. • profile: A HawkinAuth object. If not provided, the active connection is used.

Value

Response will be a data frame containing the groups that are in the organization. Each group has the following variables:

Column Name	Type	Description
id	*chr*	group's unique ID
name	*chr*	group's given name

Examples

```
## Not run:
# This is an example of how the function would be called.

df_groups <- get_groups()


## End(Not run)
```

`get_metrics` *Get Test Metrics*

Description

Get the metrics and ids for all the metrics in the system.

Usage

```
get_metrics(testType = "all")
```

Arguments

`testType`	Supply a value of type string. Must be canonical test Id, test type name, or test name abbreviation. Names | Abbreviations: "Countermovement Jump" | "CMJ", "Squat Jump" | "SJ", "Isometric Test" | "ISO", "Drop Jump" | "DJ", "Free Run" | "FREE", "CMJ Rebound" | "CMJR", "Multi Rebound" | "MR", "Weigh In" | "WI", "Drop Landing" | "DL", "TS Isometric Test" | "TSISO", "TS Free Run" | "TSFREE"

Value

Response will be a data frame containing the tests metrics that are in the HD system. The parameter `testType` can be used to filter and return only metrics of the specified type.

The returned data frame will follow the following schema:

Column Name	Type	Description
canonicalTestTypeID	*chr*	Canonical Test Id
testTypeName	*chr*	Given Test Name
id	*chr*	camelCase Test Name
label	*chr*	Outward facing label or title
label_unit	*chr*	Outward facing label or title w/ unit of measure
header	*chr*	header of data frame output
units	*chr*	Unit of measure (if any)
description	*chr*	Description or definition of metric

Examples

```
## Not run:
# This is an example of how the function would be called.

df_testsMetrics <- get_metrics(testType = "CMJ")


## End(Not run)
```

`get_tags` *Get Tags*

Description

Get the tag names and ids for all the tags in the system.

Usage

```
get_tags(...)
```

Arguments

`...`	Optional arguments. • `profile`: A `HawkinAuth` object or a character string of the profile name. If not provided, the active connection is used.

Value

Response will be a data frame containing the tags that are in the organization. Each tag has the following variables:

Column Name	Type	Description
id	*chr*	tag's unique ID
name	*chr*	tag's given name
description	*chr*	description of tag provided by user

Examples

```
## Not run:
# Use active connection
df_tags <- get_tags()

# Use a specific profile by name
df_tags_dev <- get_tags(profile = "my_dev_profile")

## End(Not run)
```

`get_teams` *Get Teams*

Description

Get the team names and ids for all the teams in the org.

Usage

```
get_teams(...)
```

Arguments

`...`	Optional arguments. • `profile`: A `HawkinAuth` object. If not provided, the active connection is used.

Value

Response will be a data frame containing the teams that are in the organization. Each team has the following variables:

Column Name	Type	Description
id	*chr*	team's unique ID
name	*chr*	team's given name

Examples

```
## Not run:
# This is an example of how the function would be called.

df_teams <- get_teams()


## End(Not run)
```

get_tests *Get All Tests or Sync Tests*

Description

Retrieves test data from the Hawkin Dynamics API. This function replaces all previous `get_tests_*` functions. It can filter by athlete, team, group, date range, or sync timestamp.

Usage

```
get_tests(
 from = NULL,
 to = NULL,
 sync = FALSE,
 athleteId = NULL,
 typeId = NULL,
 teamId = NULL,
 groupId = NULL,
 includeInactive = FALSE,
 includeEid = FALSE,
 ...
 )
```

Arguments

`from`	Optionally supply a time frame **start** value. Accepts either: • A Unix timestamp as an `integer` (e.g., `1689958617`), or • A date as a `character` string in `"YYYY-MM-DD"` format (e.g., `"2023-08-01"`). Optional. If not supplied, no lower bound is applied and all available history is returned.
`to`	Optionally supply a time frame **end** value. Accepts either: • A Unix timestamp as an `integer` (e.g., `1691207356`), or • A date as a `character` string in `"YYYY-MM-DD"` format (e.g., `"2023-08-10"`). Optional. If not supplied, no upper bound is applied (results run through the most recent test).
`sync`	The result set will include updated and newly created tests. This parameter is best suited to keep your database in sync with the Hawkin database. If you do not supply this value you will receive every test.
`athleteId`	Supply an athlete's id to receive tests for a specific athlete
`typeId`	Supply a value of type string. Must be canonical test Id, test type name, or test name abbreviation. Names | Abbreviations: "Countermovement Jump" | "CMJ", "Squat Jump" | "SJ", "Isometric Test" | "ISO", "Drop Jump" | "DJ", "Free Run" | "FREE", "CMJ Rebound" | "CMJR", "Multi Rebound" | "MR", "Weigh In" | "WI", "Drop Landing" | "DL", "TS Free Run" | "TSFR", "TS Isometric Test" | "TSISO"

`teamId` Supply a team's ID, a list of team IDs, or a string of a comma separated team id's to receive tests from the specified teams. A maximum of 10 teams can be fetched at once.

`groupId` Supply a group's ID, a list of group IDs, or a string of a comma separated group id's to receive tests from the specified groups. A maximum of 10 groups can be fetched at once.

`includeInactive` Logical. Default `FALSE`, which sends `includeInactive=false` to the API so that only active tests are returned server-side. Set to `TRUE` to include inactive (disabled) trials in the response.

`includeEid` Logical. Default `FALSE`. When `TRUE`, the API includes an `eid` (equipment ID) field on each test record identifying the hardware that produced the trial. The column will be `NA` for any tests where the equipment ID is not recorded.

`...` Optional arguments.

- `profile`: A `HawkinAuth` object or a character string of the profile name. If not provided, the active connection is used.
- `chunk_size`: **Deprecated.** Previously controlled time-window chunking. Now ignored; the API uses cursor-based pagination with a fixed page size of 1,000.

Details

Pagination: This function uses cursor-based pagination (API v1.13+) to reliably fetch all matching tests regardless of dataset size. The API returns pages of up to 1,000 tests each. The function loops automatically until all pages are retrieved.

- If `from` is provided, it is used as a time-range filter.
- If `from` is NOT provided, no lower bound is applied and all available history is returned. Pagination handles arbitrarily large result sets.
- If `to` is NOT provided, no upper bound is applied (results run through the most recent test).

Schema Handling: Because metrics may be added to the system over time, older test results might have fewer columns than newer ones. This function uses `dplyr::bind_rows` to combine pages, filling missing columns with `NA` where necessary.

Athlete columns: The athlete block is prefixed with `athlete_` and includes the core fields (`athlete_id`, `athlete_name`, `athlete_active`, `athlete_teams`, `athlete_groups`), the API v1.14 profile fields when present (`athlete_image`, `athlete_position`, `athlete_dob`, `athlete_sport`, `athlete_height`, `athlete_lastTestedOn`), and one `athlete_<key>` column per external (custom) property. Athletes missing a given external key receive `NA`.

Value

A data frame containing test trials and their metrics. Each row represents a single test trial. (if specified).

Examples

```
## Not run:
# Call for all tests (active connection)
dfAllTests <- get_tests()

# Call for all tests within a specific time frame
dfFromTo <- get_tests(from = "2023-08-01", to = "2023-08-10")

# Include inactive tests
dfAll <- get_tests(from = "2023-08-01", includeInactive = TRUE)

## End(Not run)
```

---

get_testTypes *Get Test Types*

Description

Get the test type names and ids for all the test types in the system.

Usage

```
get_testTypes()
```

Value

Response will be a data frame containing the tests that are in the HD system. Each test type includes the following variables:

Column Name	Type	Description
canonicalId	*chr*	test's unique canonical ID
name	*chr*	test's given name
abbreviation	*chr*	test given name abbreviation

Examples

```
## Not run:
# This is an example of how the function would be called.

df_tests <- get_testTypes()


## End(Not run)
```

HawkinAuth *Hawkin Authentication Class*

Description

Manages the API session state, including the access token lifecycle and regional endpoints.

Usage

```
HawkinAuth(
  config = HawkinConfig(),
  access_token = NULL,
  expires_at = NULL,
  region = "Americas"
)
```

Arguments

config	HawkinConfig. The configuration settings.
access_token	character. The ephemeral Bearer token.
expires_at	POSIXct. The timestamp when the access token expires.
region	character. The data region ("Americas", "Europe", "APAC").

Value

An S7 object of class HawkinAuth representing the API session state, including its config, access_token, expires_at, region, and the computed read-only base_url.

Properties

- base_url: The computed API base URL for the region (read-only).

HawkinConfig *Hawkin Configuration Class*

Description

Stores environment and profile configuration settings.

Usage

```
HawkinConfig(
  profile = "default",
  org_id = "v1",
  environment = "development",
  log_level = "INFO"
)
```

Arguments

profile	character. The profile name used for credential lookup (default: "default").
org_id	character. The organization ID for API paths (default: "v1").
environment	character. "development" (keyring) or "production" (env vars).
log_level	character. Logging verbosity ("INFO", "DEBUG", "WARN").

Value

An S7 object of class HawkinConfig storing environment and profile configuration settings (profile, org_id, environment, and log_level).

---

hd_auth_reset *Remove Hawkin Credentials*

Description

Deletes a stored Refresh Token from the system keychain.

Usage

```
hd_auth_reset(profile = "default")
```

Arguments

profile	character. The name of the profile to remove.

Value

No return value, called for side effects. Deletes the stored refresh token for profile from the operating system credential store.

---

hd_auth_store *Store Hawkin Credentials Securely*

Description

Securely saves your Hawkin Dynamics Refresh Token to your operating system's credential store (Keychain on macOS, Credential Manager on Windows).

Usage

```
hd_auth_store(profile = "default", token = NULL)
```

Arguments

profile	character. A name for this set of credentials (default: "default").
token	character. Optional. If NULL (default), a secure prompt will appear.

Value

No return value, called for side effects. Stores the refresh token for profile in the operating system credential store.

Note

When using the token parameter directly (e.g., hd_auth_store(token = "...")), be aware that the token string may be recorded in your .Rhistory file. For maximum security, omit the token parameter to use the secure OS prompt instead.

---

hd_connect *Connect to Hawkin Dynamics API*

Description

Initializes a connection to the Hawkin Dynamics Cloud. This function creates the session object, performs the initial authentication, and sets it as the active connection for subsequent data calls.

Usage

```
hd_connect(
  profile = "default",
  org_id = "v1",
  environment = "development",
  region = "Americas",
  log_level = "INFO"
)
```

Arguments

profile	character. The name of the stored profile credential to use (default: "default").
org_id	character. Your Organization ID. Defaults to "v1" for standard users.
environment	character. "development" to use local keychain, or "production" to use Environment Variables.
region	character. The API region: "Americas" (default), "Europe", or "APAC".
log_level	character. Logging verbosity: "INFO", "DEBUG", or "WARN".

Value

Invisibly returns the authenticated HawkinAuth object.

`initialize_logger` *Logger Settings Controller*

Description

This function initializes the logger with customization parameters, including the output destination and log thresholds for both stdout and log file.

Usage

```
initialize_logger(
 log_output = "stdout",
 log_threshold_stdout = "INFO",
 log_file = "hawkinRlog.log",
 log_threshold_file = "INFO"
 )
```

Arguments

`log_output`	A character string specifying the log output destination. Options are "stdout", "file", or "both". Default is "stdout".
`log_threshold_stdout`	A character string specifying the log threshold for stdout. Options are "TRACE", "DEBUG", "INFO", "WARN", "ERROR", "FATAL". Default is "INFO".
`log_file`	A character string specifying the name of the custom log file.
`log_threshold_file`	A character string specifying the log threshold for the log file. Options are "TRACE", "DEBUG", "INFO", "WARN", "ERROR", "FATAL". Default is "INFO".

Details

The function sets up the logging configuration with the specified output destination and log thresholds. If "stdout" is chosen, logs will be output to the console. If "file" is chosen, logs will be written to a file named "hawkinRlog.log" by default. The `log_file` parameter can be used to use a different file name and location. If "both" is chosen, logs will be output to both the console and the log file.

Users can specify different log thresholds for stdout and log file.

Value

No return value, called for side effects. Configures the `logger` package's layout, threshold, and appender according to the arguments.

Examples

```
## Not run:
# Log messages as stdout with a DEBUG threshold
initialize_logger(log_output = "stdout", log_threshold_stdout = "DEBUG")

# Log to file in 'app' directory with a TRACE threshold
initialize_logger(
  log_output = "file", log_file = "app/mylog", log_threshold_file = "TRACE"
)

## End(Not run)
```

`MetricDictionary` *Metric Dictionary*

Description

This data frame contains metrics from all tests.

Usage

```
MetricDictionary
```

Format

A data frame with 484 rows and 7 variables:

canonicalTestTypeId The unique id of the canonical test type (character).

testTypeName The given / common name of the test type (character).

id PascalCase formatted name of the metric (character).

label The metric label found in app and cloud UI (character).

label_unit The metric name returned from the API (character).

header header of data frame output (character).

units Metric unit of measure (character).

description A verbose description of the metric (character).

Source

Generated for demonstration purposes.

`update_athletes` *Update Athletes*

Description

Update an athlete or athletes for an account. Bulk update up to 500 athletes at a time.

Usage

```
update_athletes(athleteData, ...)
```

Arguments

`athleteData`	Provide a data frame of the athlete or athletes to be updated.
`...`	Optional arguments. • `profile`: A `HawkinAuth` object. If not provided, the active connection is used.

Details

The data frame passed as the argument must use the following schema:

Column Name	Type	Inclusion	Description
id	*chr*	**REQUIRED**	athlete's Hawkin Dynamics unique ID
name	*chr*	*optional*	athlete's given name (First Last)
image	*chr*	*optional*	URL path to image. `default = null`
active	*logi*	*optional*	athlete is active (TRUE). `default = null`
teams	*list*	*optional*	a single team id as a string or list of team ids. `default = [defaultTeamId]`
groups	*list*	*optional*	a single group id as a string or list of group ids. `default = []`
external property	*chr*	*optional*	External properties can be added by adding any additional columns of equal len

If optional fields are not present in an update request, those properties will be left unchanged. However, when updating external properties, custom properties that are not present will be removed.

Value

If successful, a confirmation message will be printed with the number of successful athletes updated. If there are failures, a data frame containing the athletes that failed to be updated will be returned with columns:

Column Name	Type	Description
reason	*chr*	Reason for failed update
name	*chr*	Athlete's given name (First Last)

Examples

```
## Not run:
# Example data frame following the required schema
df <- data.frame(
  id = c("uniqueAthleteIdOne", "uniqueAthleteIdTwo"),
  name = c("John Doe", "Jane Smith"),
  image = c("https://example.com/johndoe.jpg", "https://example.com/janesmith.jpg"),
  active = c(TRUE, FALSE),
  teams = I(list("team1", c("team2", "team3"))),
  groups = I(list(NULL, "group1")),
  external_property = c("value1", "value2")
)

# Update athletes using the example data frame
update_athletes(athleteData = df)

## End(Not run)
```

Index