Package: hawkinR (via r-universe)

October 18, 2026

Title hawkinR - Hawkin Dynamics API Client

Version 1.2.4

Description Provides simple functionality with Hawkin Dynamics API. These functions are for use with 'Hawkin Dynamics Beta API' version 1.8-beta. You must be an Hawkin Dynamics user with active integration account to utilize functions within the package. This API is designed to get data out of your Hawkin Dynamics database into your own database. It is not designed to be accessed from client applications directly. There is a limit on the amount of data that can be returned in a single request. As your database grows, it will be necessary to use the from and to parameters to limit the size of the responses. Responses that exceed the memory limit will fail.

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

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

Depends R (>= 3.5.0)

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports dplyr (>= 0.7.4), jsonlite (>= 1.8.7), lubridate (>= 1.9.2), magrittr (>= 2.0.3), rlang (>= 1.1.1), janitor(>= 2.2.0), stats, httr2 (>= 1.0.1), logger (>= 0.3.0), tidyselect(>= 1.2.1), later(>= 0.7.3)

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

LazyData true

Config/pak/sysreqs libicu-dev libssl-dev

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

Date/Publication 2026-07-15 23:53:36 UTC

RemoteUrl https://github.com/hawkindynamics/hawkinr

RemoteRef HEAD

RemoteSha bac9d28a93fe42e84b7cec239af98b6c1992aa2f

Contents

create_athletes . . . 2
get_access . . . 3
get_athletes . . . 5
get_forcetime . . . 6
get_groups . . . 7
get_metrics . . . 7
get_tags . . . 8
get_teams . . . 9
get_tests . . . 10
get_tests_ath . . . 12
get_tests_group . . . 14
get_tests_team . . . 16
get_tests_type . . . 18
get_testTypes . . . 20
initialize_logger . . . 21
MetricDictionary . . . 22
update_athletes . . . 23

Index **25**

`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:

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("http://example.com/johndoe.jpg", "http://example.com/janesmith.jpg"),
  active = c(TRUE, FALSE),
  teams = c("team1", c("team2", "team3"))),
  groups = c(NULL, "group1")),
  external_property = c("value1", "value2")
)

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

## End(Not run)
```

`get_access` *Get Access Token*

Description

Use the Refresh Token generated to get a valid Access Token. Only the organization administrator account has the ability to generate API tokens.

Usage

```
get_access(`refreshToken`, `region` = "Americas", `org_name` = NULL)
```

Arguments

refreshToken	Use the Refresh Token generated from 'https://cloud.hawkindynamics.com/integrations'.
region	The region to define the URL to be used. Options: "Americas" (default), "Europe", "Asia/Pacific".
org_name	The organization endpoint created by our tech support team.

Details

Use this function to initiate access to your data in the cloud. All other hawkinR functions will depend on the values returned from this function.

When correct inputs are passed through the region and refreshToken parameters, the returned access token, expiration time, and regional URL will be stored in the system for use by other functions during this session.

The accessToken is set to expire every 60 minutes. If the token has expired, and you attempt to use a dependent function, you will be prompted to run this function again to receive a new access token.

Shown as <org_name> in the API docs, the org_name parameter allows for custom configurations to take effect.

Value

A data frame with necessary information for accessing API (access token, token expiration, URL region). The contents of the data frame are stored in the system environment.

Examples

```
## Not run:
# This is an example of how the function would be called with the region defaulting to "Americas".
# Replace 'refresh token' with an actual authentication token.

get_access('refreshToken')

# If you are in a different region and use one of the other URLs, declare your region by using the
#`region` parameter.

get_access('refreshToken', region = "Europe", org_name = "MyOrgName")

## 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.

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 athlete's profile image
position	*chr*	athlete's position
dob	*chr*	athlete's date of birth
sport	*chr*	athlete's sport
height	*chr*	athlete's height
lastTestedOn	*chr*	date of athlete's last test
external	*chr*	external properties will have a column of their name with the appropriate values for the athlete of NA

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_forcetime` *Get Force-Time Data*

Description

Get the force-time data for a specific test by id. This includes both left, right and combined force data at 1000hz (per millisecond). Calculated velocity, displacement, and power at each time interval will also be included.

Usage

```
get_forcetime(testId)
```

Arguments

`testId`	Give the unique test id of the trial you want to be called.

Value

Response will be a data frame containing the following:

Column Name	Type	Description
time_s	*int*	Elapsed time in seconds, starting from end of identified quiet phase
force_right	*int*	Force recorded from the RIGHT platform coinciding with time point from `time_s`, measured in N
force_Left	*int*	Force recorded from the LEFT platform coinciding with time point from `time_s`, measured in Ne
force_combined	*int*	Sum of forces from LEFT and RIGHT, coinciding with time point from `time_s`, measured in New
velocity_m.s	*int*	Calculated velocity of center of mass at time interval, measured in meters per second (m/s)
displacement_m	*int*	Calculated displacement of center of mass at time interval, measured in meters (m)
power_w	*int*	Calculated power of mass at time interval, measured in watts (W)

Examples

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

df_ft <- get_forcetime(testId = `stringId`)


## End(Not run)
```

get_groups *Get Groups*

Description

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

Usage

```
get_groups()
```

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()
```

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:
# This is an example of how the function would be called.

df_tags <- get_tags()


## End(Not run)
```

`get_teams` *Get Teams*

Description

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

Usage

```
get_teams()
```

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

Get the tests for an account. You can specify a time frame from, or to, which the tests should come (or be synced).

Usage

```
get_tests(
 from = NULL,
 to = NULL,
 sync = FALSE,
 athleteId = NULL,
 typeId = NULL,
 teamId = NULL,
 groupId = NULL,
 includeInactive = 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"`). If not supplied, all available tests from the earliest record will be returned. Use this parameter for bulk exports or to define a starting point for data retrieval.
`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"`). If not supplied, all available tests up to the latest record will be returned, or up to the `from` parameter if specified. Use this parameter to limit the range of historical data retrieved.

`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`

There was a change to the default API configuration to reflect the majority of users API configuration. Inactive tests or tests where `active = false` are returned in these configuration. Be default, `includeInactive` is set to `FALSE`. To return all tests, including disabled trials, set `includeInactive` to `TRUE`.

Value

Response will be a data frame containing the trials within the time range (if specified).

Column Name	Type	Description
id	*str*	Test trial unique ID
active	*logi*	The trial is active and not disabled
timestamp	*int*	UNIX time stamp of trial
segment	*chr*	Description of the test type and trial number of the session (testType:trialNo)
test_type_id	*chr*	Id of the test type of the trial
test_type_name	*chr*	Name of the test type of the trial
test_type_canonicalId	*chr*	Canonical Id of the test type of the trial
test_type_tag_ids	*chr*	String of Ids associated with tags used during the test trial
test_type_tag_names	*chr*	String of names of tags used during the test trial
test_type_tag_desc	*chr*	String of descriptions of tags used during the test trial
athlete_id	*chr*	Unique Id of the athlete
athlete_name	*chr*	Athlete given name
athlete_active	*logi*	The athlete is active
athlete_teams	*list*	List containing Ids of each team the athlete is on
athlete_groups	*list*	List containing Ids of each group the athlete is in
athlete_image	*chr*	URL to athlete's profile image

athlete_position	*chr*	Athlete's position
athlete_dob	*chr*	Athlete's date of birth
athlete_sport	*chr*	Athlete's sport
athlete_height	*chr*	Athlete's height

All metrics from each test type are included as the remaining variables unless `typeId` is provided, then only the metrics of that test type will be returned. If a trial does not have data for a variable it is returned NA.

Examples

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

# Call for all tests

dfAllTests <- get_tests()


# Call for all tests within a specific time frame

dfFromTo <- get_tests(from = 1689958617, to = 1691207356)


# Call for all new tests since a specific date, or any tests that have been
updated/changed since that date

dfSync <- get_tests(from = 1689958617, sync=TRUE)


## End(Not run)
```

`get_tests_ath` *Get Test Trials By Athlete*

Description

Deprecated: Use `get_tests` instead, which has been expanded to handle all requests.

Get only tests of the specified athlete for an account.

Usage

```
get_tests_ath(athleteId, from = NULL, to = NULL, sync = FALSE, includeInactive = FALSE)
```

Arguments

`athleteId`	Supply an athlete's id to receive tests for a specific athlete
`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"`). If not supplied, all available tests from the earliest record will be returned. Use this parameter for bulk exports or to define a starting point for data retrieval.
`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"`). If not supplied, all available tests up to the latest record will be returned, or up to the `from` parameter if specified. Use this parameter to limit the range of historical data retrieved.
`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.
`includeInactive`	There was a change to the default API configuration to reflect the majority of users API configuration. Inactive tests or tests where `active = false` are returned in these configuration. Be default, `includeInactive` is set to `FALSE`. To return all tests, including disabled trials, set `includeInactive` to `TRUE`.

Value

Response will be a data frame containing the trials within the time range (if specified).

Column Name	Type	Description
id	*str*	Test trial unique ID
active	*logi*	The trial is active and not disabled
timestamp	*int*	UNIX time stamp of trial
segment	*chr*	Description of the test type and trial number of the session (testType:trialNo)
test_type_id	*chr*	Id of the test type of the trial
test_type_name	*chr*	Name of the test type of the trial
test_type_canonicalId	*chr*	Canonical Id of the test type of the trial
test_type_tag_ids	*chr*	String of Ids associated with tags used during the test trial
test_type_tag_names	*chr*	String of names of tags used during the test trial
test_type_tag_desc	*chr*	String of descriptions of tags used during the test trial
athlete_id	*chr*	Unique Id of the athlete
athlete_name	*chr*	Athlete given name
athlete_active	*logi*	The athlete is active
athlete_teams	*list*	List containing Ids of each team the athlete is on
athlete_groups	*list*	List containing Ids of each group the athlete is in

All metrics from each test type are included as the remaining variables. If a trial does not have data for a variable it is returned NA.

See Also

get_tests

Examples

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


## Call for all tests from a specified athlete
df_cmj <- get_tests_ath(athleteId = "athleteId")


## Call for all tests within a specific time frame
dfFromTo <- get_tests_ath(athleteId = "athleteId", from = 1689958617, to = 1691207356)


## Call for all tests since a specific date
dfSince <- get_tests_ath("athleteId", from = 1689958617)

## End(Not run)
```

---

get_tests_group *Get Test Trials By Groups*

Description

Deprecated: Use `get_tests` instead, which has been expanded to handle all requests.

Get only tests of the specified group for an account.

Usage

```
get_tests_group(groupId, from = NULL, to = NULL, sync = FALSE, includeInactive = FALSE)
```

Arguments

`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.
`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"`). If not supplied, all available tests from the earliest record will be returned. Use this parameter for bulk exports or to define a starting point for data retrieval.
`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"`).

If not supplied, all available tests up to the latest record will be returned, or up to the `from` parameter if specified. Use this parameter to limit the range of historical data retrieved.

`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.

`includeInactive`

There was a change to the default API configuration to reflect the majority of users API configuration. Inactive tests or tests where `active = false` are returned in these configuration. Be default, `includeInactive` is set to `FALSE`. To return all tests, including disabled trials, set `includeInactive` to `TRUE`.

Value

Response will be a data frame containing the trials within the time range (if specified).

Column Name	Type	Description
id	*str*	Test trial unique ID
active	*logi*	The trial is active and not disabled
timestamp	*int*	UNIX time stamp of trial
segment	*chr*	Description of the test type and trial number of the session (testType:trialNo)
test_type_id	*chr*	Id of the test type of the trial
test_type_name	*chr*	Name of the test type of the trial
test_type_canonicalId	*chr*	Canonical Id of the test type of the trial
test_type_tag_ids	*chr*	String of Ids associated with tags used during the test trial
test_type_tag_names	*chr*	String of names of tags used during the test trial
test_type_tag_desc	*chr*	String of descriptions of tags used during the test trial
athlete_id	*chr*	Unique Id of the athlete
athlete_name	*chr*	Athlete given name
athlete_active	*logi*	The athlete is active
athlete_teams	*list*	List containing Ids of each team the athlete is on
athlete_groups	*list*	List containing Ids of each group the athlete is in

All metrics from each test type are included as the remaining variables. If a trial does not have data for a variable it is returned NA.

See Also

`get_tests`

Examples

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

```
## Call for all tests by Group 1
dfGroup1 <- get_tests_group(groupId = "group1")


## Call for all tests from Groups 1 & 2
dfGroups_1_2 <- get_tests_group(groupId = paste0("group1","group2"))


## Call for all Group 1 tests since a specific date
df_Group1_Since <- get_tests_group("group1", from = 1689958617)


## End(Not run)
```

`get_tests_team` *Get Test Trials By Teams*

Description

Deprecated: Use `get_tests` instead, which has been expanded to handle all requests.

Get only tests of the specified team for an account.

Usage

```
get_tests_team(teamId, from = NULL, to = NULL, sync = FALSE, includeInactive = FALSE)
```

Arguments

`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.
`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"`). If not supplied, all available tests from the earliest record will be returned. Use this parameter for bulk exports or to define a starting point for data retrieval.
`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"`). If not supplied, all available tests up to the latest record will be returned, or up to the `from` parameter if specified. Use this parameter to limit the range of historical data retrieved.

`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.

`includeInactive`
There was a change to the default API configuration to reflect the majority of users API configuration. Inactive tests or tests where `active = false` are returned in these configuration. Be default, `includeInactive` is set to `FALSE`. To return all tests, including disabled trials, set `includeInactive` to `TRUE`.

Value

Response will be a data frame containing the trials within the time range (if specified).

Column Name	Type	Description
id	*str*	Test trial unique ID
active	*logi*	The trial is active and not disabled
timestamp	*int*	UNIX time stamp of trial
segment	*chr*	Description of the test type and trial number of the session (testType:trialNo)
test_type_id	*chr*	Id of the test type of the trial
test_type_name	*chr*	Name of the test type of the trial
test_type_canonicalId	*chr*	Canonical Id of the test type of the trial
test_type_tag_ids	*chr*	String of Ids associated with tags used during the test trial
test_type_tag_names	*chr*	String of names of tags used during the test trial
test_type_tag_desc	*chr*	String of descriptions of tags used during the test trial
athlete_id	*chr*	Unique Id of the athlete
athlete_name	*chr*	Athlete given name
athlete_active	*logi*	The athlete is active
athlete_teams	*list*	List containing Ids of each team the athlete is on
athlete_groups	*list*	List containing Ids of each group the athlete is in

All metrics from each test type are included as the remaining variables. If a trial does not have data for a variable it is returned NA.

See Also

`get_tests`

Examples

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


## Call for all tests by Group 1
dfteam1 <- get_tests_team(teamId = "team1")


## Call for all tests from Groups 1 & 2
dfteam_1_2 <- get_tests_team(teamId = paste0("team1","team2"))
```

```
## Call for all Group 1 tests since a specific date
df_team1_Since <- get_tests_team("team1", from = 1689958617)


## End(Not run)
```

`get_tests_type` *Get Test Trials By Test Type*

Description

Deprecated: Use `get_tests` instead, which has been expanded to handle all requests.

Get only tests of the specified type for an account.

Usage

```
get_tests_type(typeId, from = NULL, to = NULL, sync = FALSE, includeInactive = FALSE)
```

Arguments

`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"
`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"`). If not supplied, all available tests from the earliest record will be returned. Use this parameter for bulk exports or to define a starting point for data retrieval.
`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"`). If not supplied, all available tests up to the latest record will be returned, or up to the `from` parameter if specified. Use this parameter to limit the range of historical data retrieved.
`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.

`includeInactive`

There was a change to the default API configuration to reflect the majority of users API configuration. Inactive tests or tests where `active = false` are returned in these configuration. Be default, `includeInactive` is set to `FALSE`. To return all tests, including disabled trials, set `includeInactive` to `TRUE`.

Value

Response will be a data frame containing the trials within the time range (if specified).

Column Name	Type	Description
id	*str*	Test trial unique ID
active	*logi*	The trial is active and not disabled
timestamp	*int*	UNIX time stamp of trial
segment	*chr*	Description of the test type and trial number of the session (testType:trialNo)
test_type_id	*chr*	Id of the test type of the trial
test_type_name	*chr*	Name of the test type of the trial
test_type_canonicalId	*chr*	Canonical Id of the test type of the trial
test_type_tag_ids	*chr*	String of Ids associated with tags used during the test trial
test_type_tag_names	*chr*	String of names of tags used during the test trial
test_type_tag_desc	*chr*	String of descriptions of tags used during the test trial
athlete_id	*chr*	Unique Id of the athlete
athlete_name	*chr*	Athlete given name
athlete_active	*logi*	The athlete is active
athlete_teams	*list*	List containing Ids of each team the athlete is on
athlete_groups	*list*	List containing Ids of each group the athlete is in

Only metrics of the given test type are included as the remaining variables.

See Also

`get_tests`

Examples

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


## Call for all CMJ tests
df_cmj <- get_tests_type(typeId = "7nNduHeM5zETPjHxvm7s")


## Call for Free Run tests within a  specific time frame
df_free <- get_tests_type(typeId = "5pRSUQVSJVnxijpPMck3", from = 1689958617, to = 1691207356)


## Call for Squat Jump tests since a specific date
df_sjSince <- get_tests_type("QEG7m7DhYsD6BrcQ8pic", from = 1689958617)
```

```
## 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)
```

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.

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.

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 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(
  id = c("uniqueAthleteIdOne", "uniqueAthleteIdTwo"),
  name = c("John Doe", "Jane Smith"),
  image = c("http://example.com/johndoe.jpg", "http://example.com/janesmith.jpg"),
  active = c(TRUE, FALSE),
  teams = c("team1", c("team2", "team3"))),
  groups = c(NULL, "group1")),
  external_property = c("value1", "value2")
)

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

## End(Not run)
```

Index

∗ **datasets**
MetricDictionary, 22

create_athletes, 2

get_access, 3
get_athletes, 5
get_forcetime, 6
get_groups, 7
get_metrics, 7
get_tags, 8
get_teams, 9
get_tests, 10
get_tests_ath, 12
get_tests_group, 14
get_tests_team, 16
get_tests_type, 18
get_testTypes, 20

initialize_logger, 21

MetricDictionary, 22

update_athletes, 23