Package: nhlapi (via r-universe)

June 22, 2024

Type Package

Title A Minimum-Dependency 'R' Interface to the 'NHL' API

Version 0.1.4.900

Maintainer Jozef Hajnala <jozef.hajnala@gmail.com>

Description Retrieves and processes the data exposed by the open 'NHL' API. This includes information on players, teams, games, tournaments, drafts, standings, schedules and other endpoints. A lower-level interface to access the data via URLs directly is also provided.

Depends R (>= 2.10)

Imports jsonlite

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Roxygen list(markdown = TRUE)

Suggests testthat, roxygen2, knitr, rmarkdown

License AGPL-3

Language en-US

URL https://github.com/jozefhajnala/nhlapi

BugReports https://github.com/jozefhajnala/nhlapi/issues

VignetteBuilder knitr

SysDataCompression xz

Copyright NHL and the NHL Shield are registered trademarks of the National Hockey League. NHL and NHL team marks are the property of the NHL and its teams.

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

RemoteUrl https://github.com/jozefhajnala/nhlapi

RemoteRef HEAD

RemoteSha f3b40ee93b1687dfb166e3b963910c29fef72456

Contents

`make_log` *Create a log message*

Description

Create a log message

Usage

```
make_log(
  msg,
  ...,
  type = "I",
  dtFormat = getOption("nhlapi_log_datetime"),
  newLine = FALSE,
  sep = " | ",
  collapse = " ",
  lineBreak = "$",
  endNewLine = FALSE
)
```

Arguments

`msg`	`character(1)`, to be logged.
`...`	additional `character()` strings to be logged. Will be pasted to `msg` and collapsed using the `collapse` argument.
`type`	`character(1)` ideally 1 uppercase letter.

dtFormat	character(1), passed to format for [Sys.time()]
newLine	logical(1), if TRUE, new line will be pasted. to the beginning of the message.
sep	character(1) string, to separate parts of the message.
collapse	character(1), to collapse msg and ....
lineBreak	character(1), replacing line breaks in msg.
endNewLine	logical(1), if TRUE, new line will be pasted to the end of the message.

Value

character(1), constructed log message.

Examples

```
nhlapi:::make_log("Dummy warning", type = "W")
```

---

nhl_awards *Retrieve metadata on NHL awards from the API*

Description

Retrieve metadata on NHL awards from the API

Usage

```
nhl_awards(awardIds = NULL)
```

Arguments

awardIds	integer(), vector of one or more award ids or NULL (default) for all awards. The current set of valid ids seems to be 1:24.

Value

data.frame, with information on awards, one row per award.

Examples

```
## Not run:
  # Get information on all awards
  nhl_awards()

  # Get information on 3 historical awards
  nhl_awards(1:3)

## End(Not run)
```

nhl_conferences *Retrieve metadata on NHL conferences from the API*

Description

Retrieve metadata on NHL conferences from the API

Usage

```
nhl_conferences(conferenceIds = NULL)
```

Arguments

conferenceIds	integer(), ids of the conferences or NULL (default) for all conferences As of end of 2019, the valid conference ids seem to be in the 1:7 range.

Value

data.frame, with information on conferences, one row per conference.

Examples

```
## Not run:
  # Get information on all conferences
  nhl_conferences()

  # Get information on 2 selected conferences
  nhl_conferences(5:6)

## End(Not run)
```

nhl_divisions *Retrieve metadata on NHL divisions from the API*

Description

Retrieve metadata on NHL divisions from the API

Usage

```
nhl_divisions(divisionIds = NULL)
```

Arguments

divisionIds	integer(), ids of the divisions or NULL (default) for all divisions. As of end of 2019, the valid division ids seem to be in the 1:25 range.

Value

`data.frame`, with information on divisions, one row per division.

Examples

```
## Not run:
  # Get information on all divisions
  nhl_divisions()

  # Get information on 2 selected divisions
  nhl_divisions(15:16)

## End(Not run)
```

`nhl_drafts` *Retrieve metadata on NHL drafts from the API*

Description

Retrieve metadata on NHL drafts from the API

Usage

```
nhl_drafts(draftYears = NULL)
```

Arguments

`draftYears`	`integer()`, vector of one or more years in `YYYY` format or `NULL` (default) for the current year's draft. Also accepts a character vector of years in `YYYY` format.

Value

`data.frame`, with information on drafts, one row per draft year.

Examples

```
## Not run:
  # Get information on current draft
  nhl_drafts()

  # Get information on 3 historical drafts
  nhl_drafts(2015:2017)

## End(Not run)
```

nhl_draft_prospects *Retrieve metadata on NHL draft prospects from the API*

Description

Retrieve metadata on NHL draft prospects from the API

Usage

```
nhl_draft_prospects(prospectIds = NULL)
```

Arguments

prospectIds	integer(), vector of one or more ids of draft prospects or NULL (default) for all exposed prospects.

Value

data.frame, with information on draft prospects, one row per draft prospect.

Examples

```
## Not run:
  # Get information on current draft prospects
  nhl_draft_prospects()

## End(Not run)
```

nhl_from_json *Get URL using* fromJSON

Description

Get URL using fromJSON

Usage

```
nhl_from_json(
  url,
  flatten = getOption("nhlapi_flatten"),
  silent = getOption("nhlapi_try_silent"),
  retries = getOption("nhlapi_get_retries"),
  retrySleep = getOption("nhlapi_get_retry_sleep"),
  noRetryPatt = getOption("nhlapi_get_noretry")
)
```

Arguments

url	character(1), the URL to get the data from.
flatten	logical(1), if TRUE (default) automatically flattens nested data frames into a single non-nested data frame.
silent	logical(1), passed to [try()].
retries	integer(1), number of retries in case of failed data retrieval (0L for no no retries).
retrySleep	integer(1), number of seconds to [Sys.sleep()] in between retries.
noRetryPatt	character(1), string pattern. If the error condition's message contains this pattern, there will be no retries. Useful for e.g. 404 returns where retries are likely useless.

Value

list, retrieved data if succeeded, a try-error class object otherwise.

---

nhl_games *Retrieve metadata on NHL games from the API*

Description

Retrieve metadata on NHL games from the API

Usage

```
nhl_games(gameIds, element)

nhl_games_content(gameIds)

nhl_games_feed(gameIds)

nhl_games_boxscore(gameIds)

nhl_games_linescore(gameIds)
```

Arguments

gameIds	numeric(), vector of one or more game ids. The game id is a 10 digit number where the • first 4 digits identify the season of the game, for instance 2017 for the 2017-2018 season. • next 2 digits give the type of game, where – 01 - preseason, – 02 - regular season,

  - 03 - playoffs,
  - 04 - all-star.
- final 4 digits identify the specific game number
  - for regular season and preseason games, this ranges from 0001 to the number of games played. That is 1271 for seasons with 31 teams and 1230 for seasons with 30 teams.
  - for playoff games, the
    * second digit gives the round of the playoffs
    * third digit specifies the match-up
    * fourth digit specifies the game (out of 7)

`element` `character()` vector of one or more valid elements. Currently the valid elements seem to be:

- `"linescore"`
- `"boxscore"`
- `"content"`
- `"feed/live"`

Value

`list`, with information on games, one element per game and element combination.

Functions

- `nhl_games_content`: Complex endpoint returning multiple types of media relating to the game including videos of shots, goals and saves.
- `nhl_games_feed`: returns all data about a specified game id including play data with on-ice coordinates and post-game details like first, second and third stars and details about shootouts. Note that the data returned is sizable, often over 30 000 lines.
- `nhl_games_boxscore`: Returns far less detail than `nhl_games_feed()` and may be more suitable for analyzing post-game statistics including goals, shots, penalty minutes, blocked, takeaways, etc.
- `nhl_games_linescore`: Returns even fewer details than `nhl_games_boxscore()`. Has goals, shots on goal, power-play and goalie pulled status, number of skaters and shootout information if applicable.

Examples

```
## Not run:
  # Get content for one game
  nhl_games(2017010001, "content")

  # Get both box score and content for 2 games
  nhl_games(c(2017010001, 2017010002), c("content", "boxscore"))

  # Get content for a game
  nhl_games_content(2017010001)
```

```
  # Get the game feed for a game
  nhl_games_feed(2017010001)

  # Get the box score for a game
  nhl_games_boxscore(2017010001)

  # Get the line score for a game
  nhl_games_linescore(2017010001)

## End(Not run)
```

`nhl_get_data` *Get data from the API for one or more URLs*

Description

Get data from the API for one or more URLs

Usage

```
nhl_get_data(urls, flatten = getOption("nhlapi_flatten"))
```

Arguments

`urls`	`character()`, vector of URLs to retrieve the data from.
`flatten`	`logical(1)`, if `TRUE` (default) automatically flattens nested data frames into a single non-nested data frame.

Value

`list`, results retrieved using `nhl_get_data_worker()`. One element per url. The elements contain the retrieved data if retrieval succeeded, otherwise an `nhl_get_data_error` class object.

See Also

`nhl_get_data_worker()`

Examples

```
## Not run:
  nhl_get_data(c(
    "https://statsapi.web.nhl.com/api/v1/teams/1",
    "https://statsapi.web.nhl.com/api/v1/people/8477474"
  ))

  nhl_get_data(
    "https://statsapi.web.nhl.com/api/v1/teams/1",
    flatten = FALSE
```

```
  )

## End(Not run)
```

`nhl_get_data_worker` *Get data from the API for 1 URL*

Description

Gets data from the NHL API using `nhl_from_json()`.

Usage

```
nhl_get_data_worker(
  url,
  flatten = getOption("nhlapi_flatten"),
  silent = getOption("nhlapi_try_silent"),
  retries = getOption("nhlapi_get_retries"),
  retrySleep = getOption("nhlapi_get_retry_sleep")
)
```

Arguments

`url`	`character(1)`, the URL to get the data from.
`flatten`	`logical(1)`, if `TRUE` (default) automatically flattens nested data frames into a single non-nested data frame.
`silent`	`logical(1)`, passed to `[try()]`.
`retries`	`integer(1)`, number of retries in case of failed data retrieval (`0L` for no no retries).
`retrySleep`	`integer(1)`, number of seconds to `[Sys.sleep()]` in between retries.

Value

`list`, with the retrieved data or class `nhl_get_data_error`.

See Also

`nhl_from_json()`, `nhl_url()`

nhl_make_seasons *Make a vector of seasons consumable by the API*

Description

The NHL API wants seasons defined in format "YYYYZZZZ" where ZZZZ = YYYY + 1. This is a helper to take a vector of years in "YYYY" format and create a vector of such seasons to be used with the API.

Usage

```
nhl_make_seasons(seasons = 1950:2019)
```

Arguments

seasons	numeric(), integer() or character(), vector of starting years of desired seasons in YYYY format, e.g. 1995 or "1995" for season 1995-1996. Accepts vectors such as c(1995:2000, 2010) to generate multiple seasons. Alternatively, also accepts character() with seasons in the format "YYYYZZZZ", where ZZZZ = YYYY + 1, e.g. "19951996". This is the format that ultimately gets sent to the NHL API. Some API endpoints, notably seasons exposed via nhl_seasons() also allow the value "current" to passed. This value will be returned unchanged.

Value

character(), vector of seasons suited for the NHL API.

Examples

```
nhlapi:::nhl_make_seasons()
nhlapi:::nhl_make_seasons(1995:2000)
nhlapi:::nhl_make_seasons(c(1995, 2015))
nhlapi:::nhl_make_seasons(c("1995", "2015"))
```

nhl_md_event_types *Get event types metadata*

Description

Get event types metadata

Usage

```
nhl_md_event_types()
```

Value

`list`, with metadata on event types.

`nhl_md_game_statuses` *Get game status metadata*

Description

Get game status metadata

Usage

```
nhl_md_game_statuses()
```

Value

`list`, with metadata on game statuses.

`nhl_md_game_types` *Get game type metadata*

Description

Get game type metadata

Usage

```
nhl_md_game_types()
```

Value

`list`, with metadata on game types.

`nhl_md_play_types` *Get play types metadata*

Description

Get play types metadata

Usage

```
nhl_md_play_types()
```

Value

`list`, with metadata on play types.

`nhl_md_standings_types` *Get standings types metadata*

Description

Get standings types metadata

Usage

```
nhl_md_standings_types()
```

Value

`list`, with metadata on standings types.

`nhl_md_stat_types` *Get stat types metadata*

Description

Get stat types metadata

Usage

```
nhl_md_stat_types()
```

Value

`list`, with metadata on stat types.

`nhl_md_tournament_types` *Get tournament types metadata*

Description

Get tournament types metadata

Usage

```
nhl_md_tournament_types()
```

Value

`list`, with metadata on tournament types.

nhl_players *Retrieve metadata for players based on names or ids*

Description

Retrieves information on players from the NHL API based on `playerNames` or `playerIds`. If `playerNames` are provided, they take precedence over `playerIds`.

Usage

```
nhl_players(playerNames, playerIds = NULL)
```

Arguments

`playerNames`	`character()`, vector of one or more player names. Not case sensitive for convenience.
`playerIds`	`integer()`, vector of one or more ids of the players. The ids correspond to the ids expected by the NHL API people endpoint. For most cases the `playerNames` argument can be provided for more convenient usage.

Value

`data.frame`, with information on selected players.

Examples

```
## Not run:
 # With player names
 nhl_players(c("joe SAKIC", "patrick roy"))

 # With playerIds
 nhl_players(playerIds = c(8451101, 8458554))

## End(Not run)
```

nhl_players_allseasons *Retrieve all seasons statistics for players*

Description

Retrieve all seasons statistics for players

Usage

```
nhl_players_allseasons(playerNames, playerIds = NULL)
```

Arguments

playerNames	character(), vector of one or more player names. Not case sensitive for convenience.
playerIds	integer(), vector of one or more ids of the players. The ids correspond to the ids expected by the NHL API people endpoint. For most cases the playerNames argument can be provided for more convenient usage.

Value

data.frame, with all season statistics for selected players.

Examples

```
## Not run:
 # With player names
 nhl_players_allseasons(c("joe sakic", "Peter Forsberg"))

 # With player ids
 nhl_players_allseasons(c(8451101, 8458554))

## End(Not run)
```

---

nhl_players_seasons *Retrieve selected seasons statistics for players*

Description

Retrieve selected seasons statistics for players

Usage

```
nhl_players_seasons(playerNames, seasons, playerIds = NULL, playoffs = FALSE)
```

Arguments

playerNames	character(), vector of one or more player names. Not case sensitive for convenience.
seasons	numeric(), integer() or character(), vector of starting years of desired seasons in YYYY format, e.g. 1995 or "1995" for season 1995-1996. Accepts vectors such as c(1995:2000, 2010) to generate multiple seasons. Alternatively, also accepts character() with seasons in the format "YYYYZZZZ", where ZZZZ = YYYY + 1, e.g. "19951996". This is the format that ultimately gets sent to the NHL API. Some API endpoints, notably seasons exposed via nhl_seasons() also allow the value "current" to passed. This value will be returned unchanged.

playerIds	integer(), vector of one or more ids of the players. The ids correspond to the ids expected by the NHL API people endpoint. For most cases the playerNames argument can be provided for more convenient usage.
playoffs	logical(1), if FALSE (default) get the regular seasons data, if TRUE, get the data for the playoffs.

Value

data.frame, with selected season statistics for selected players.

Examples

```
## Not run:
 nhl_players_seasons(
   playerIds = c(8451101, 8458554),
   seasons = "19951996",
   playoffs = TRUE
 )

## End(Not run)
```

---

nhl_plot_rink *Plot an NHL rink*

Description

Initialize a plot in base graphics with a to-scale NHL rink as the background

Usage

```
nhl_plot_rink()
```

Details

The placement of rink features & their sizes are exact according to the NHL rule book; see citation.

Examples

```
## Not run:
  # Retrieve some game feed data
  gameFeeds <- lapply(
   2019010001:2019010010,
   nhlapi::nhl_games_feed
  )

  # Create a data.frame with plays
  getPlaysDf <- function(gm) {
    playsRes <- try(gm[[1L]][["liveData"]][["plays"]][["allPlays"]])
    if (inherits(playsRes, "try-error")) data.frame() else playsRes
```

```
  }
  plays <- lapply(gameFeeds, getPlaysDf)
  plays <- nhlapi:::util_rbindlist(plays)
  plays <- plays[!is.na(plays$coordinates.x), ]

  # Move the coordinates to non-negative values before plotting
  plays$coordx <- plays$coordinates.x + abs(min(plays$coordinates.x))
  plays$coordy <- plays$coordinates.y + abs(min(plays$coordinates.y))

  # Select goals only
  goals <- plays[plays$result.event == "Goal", ]

  # Create the plot and add goals
  nhlapi::plot_rink()
  points(goals$coordinates.x, goals$coordinates.y)

## End(Not run)
```

`nhl_schedule` *Retrieve metadata on NHL schedule from the API*

Description

The general-purpose `nhl_schedule()` exposes many parameters, some useful helpers are exposed as separate functions to reflect common use cases. Arguments can be passed to these named via `...`.

- `nhl_schedule_today()`
- `nhl_schedule_seasons()`
- `nhl_schedule_date_range()`

Usage

```
nhl_schedule(
  seasons = NULL,
  teamIds = NULL,
  startDate = NULL,
  endDate = NULL,
  gameTypes = NULL,
  expand = NULL
)

nhl_schedule_today(...)

nhl_schedule_seasons(seasons, ...)

nhl_schedule_date_range(startDate, endDate, ...)
```

Arguments

seasons	numeric(), integer() or character(), vector of starting years of desired seasons in YYYY format, e.g. 1995 or "1995" for season 1995-1996. Accepts vectors such as c(1995:2000, 2010) to generate multiple seasons. Alternatively, also accepts character() with seasons in the format "YYYYZZZZ", where ZZZZ = YYYY + 1, e.g. "19951996". This is the format that ultimately gets sent to the NHL API. Some API endpoints, notably seasons exposed via nhl_seasons() also allow the value "current" to passed. This value will be returned unchanged.
teamIds	integer(), ids of the teams or NULL (default) for all teams. As of end of 2019, the valid team ids seem to be in the 1:54 range.
startDate	character(1), date in the format "YYYY-MM-DD" defining the start of the date interval for which the schedule is to be retrieved.
endDate	character(1), date in the format "YYYY-MM-DD" defining the end of the date interval for which the schedule is to be retrieved.
gameTypes	character(), defining the game types to retrieve. Valid game types are for example "R" for regular season or "P" for playoffs. See nhl_md_game_types() for all values and their descriptions.
expand	character(), of parameters passed as expand to the API URL. Some valid examples seem to be "round.series" and "schedule.broadcasts", "schedule.linescore", "schedule.ticket". NULL for no expand parameter.
...	other named parameters passed to nhl_schedule().

Value

list, with information on schedule, depending on provided arguments.

Functions

- nhl_schedule_today: Shortcut to get information on today's schedule.
- nhl_schedule_seasons: Shortcut to get information on schedule for one or more seasons.
- nhl_schedule_date_range: Shortcut to get information on schedule for a range of dates in "YYYY-MM-DD" format.

Examples

```
## Not run:
 # Get current schedule
 nhl_schedule()

 # Get schedule for historical seasons
 nhl_schedule(seasons = 2015:2016)

 # Get schedule for a date range
 nhl_schedule(startDate = "2018-01-02", endDate = "2018-01-02")

 # Get schedule for a date range, specific teams
```

```
 # and expand on line scores
 nhl_schedule(
   startDate = "2018-01-02",
   endDate = "2018-01-02",
   teamIds = c(29, 30),
   expand = "schedule.linescore"
 )

## End(Not run)

## Not run:
  nhl_schedule_today()

## End(Not run)
## Not run:
  # Schedule for seasons starting in 2015 and 2016
  nhl_schedule_seasons(2015:2016)

  # Schedule for seasons starting in 2015 and 2016
  # Only 1 team and expand line scores
  nhl_schedule_seasons(
    2015:2016,
    teamIds = 1,
    expand = "schedule.linescore"
  )

## End(Not run)
## Not run:
  # Schedule for October and November 2015
  nhl_schedule_date_range(
    startDate = "2015-10-01",
    endDate = "2015-11-30"
  )

  # Schedule for October and November 2015
  # Regular seasons only, specific team and expand line scores
  nhl_schedule_date_range(
    startDate = "2015-10-01", endDate = "2015-11-30",
    gameTypes = "R",
    teamIds = 2,
    expand = "schedule.linescore"
  )

## End(Not run)
```

`nhl_seasons` *Retrieve metadata on NHL seasons from the API*

Description

Retrieve metadata on NHL seasons from the API

Usage

```
nhl_seasons(seasons = NULL)
```

Arguments

seasons	numeric(), integer() or character(), vector of starting years of desired seasons in YYYY format, e.g. 1995 or "1995" for season 1995-1996. Accepts vectors such as c(1995:2000, 2010) to generate multiple seasons. Alternatively, also accepts character() with seasons in the format "YYYYZZZZ", where ZZZZ = YYYY + 1, e.g. "19951996". This is the format that ultimately gets sent to the NHL API. Some API endpoints, notably seasons exposed via nhl_seasons() also allow the value "current" to passed. This value will be returned unchanged.

Value

data.frame, with information on seasons, one row per year.

Examples

```
## Not run:
  # Get information on all seasons
  nhl_seasons()

  # Get information on 3 historical seasons
  nhl_seasons(2015:2017)

## End(Not run)
```

---

nhl_standings *Retrieve metadata on NHL standings from the API*

Description

Retrieve metadata on NHL standings from the API

Usage

```
nhl_standings(seasons = NULL, standingsTypes = NULL, expand = NULL)
```

Arguments

seasons	numeric(), integer() or character(), vector of starting years of desired seasons in YYYY format, e.g. 1995 or "1995" for season 1995-1996. Accepts vectors such as c(1995:2000, 2010) to generate multiple seasons. Alternatively, also accepts character() with seasons in the format "YYYYZZZZ", where ZZZZ = YYYY + 1, e.g. "19951996". This is the format that ultimately gets sent to the NHL API.

	Some API endpoints, notably seasons exposed via nhl_seasons() also allow the value "current" to passed. This value will be returned unchanged.
standingsTypes	character(), defining the standings types to retrieve. Valid standings types are for example "regularSeason" or "byDivision". See nhl_md_standings_types() for all values and their descriptions.
expand	character(), of parameters passed as expand to the API URL. A valid example seems to be "standings.record". NULL for no expand parameter.

Value

list, with information on standings depending on provided arguments.

Examples

```
## Not run:
 # Get current standings
 nhl_standings()

 # Get standings for historical seasons
 nhl_standings(seasons = 2015:2016)

 # Get standings for historical seasons
 nhl_standings(
   seasons = 2015:2016,
   standingsType = "byDivision",
   expand = "standings.record"
 )

## End(Not run)
```

---

nhl_teams *Retrieve metadata on NHL teams from the API*

Description

Retrieves team metadata such as the teams names, abbreviations, locations, conferences, venues, etc.

Usage

```
nhl_teams(teamIds = NULL, params = NULL)
```

Arguments

teamIds	integer(), ids of the teams or NULL (default) for all teams. As of end of 2019, the valid team ids seem to be in the 1:54 range.
params	named list(), further parameters passed to nhl_url_teams.

Details

The API allows to retrieve data on all teams at once, which is achieved by the default `NULL` value for the team id.

Value

`data.frame`, with data on teams, one row per team.

Examples

```
## Not run:
  nhl_teams()
  nhl_teams(1:3)

## End(Not run)
```

`nhl_teams_rosters` *Get rosters for teams*

Description

Get rosters for teams

Usage

```
nhl_teams_rosters(teamIds = NULL, seasons = NULL)
```

Arguments

`teamIds`	`integer()`, ids of the teams or `NULL` (default) for all teams. As of end of 2019, the valid team ids seem to be in the `1:54` range.
`seasons`	`numeric()`, `integer()` or `character()`, vector of starting years of desired seasons in `YYYY` format, e.g. `1995` or `"1995"` for season 1995-1996. Accepts vectors such as `c(1995:2000, 2010)` to generate multiple seasons. Alternatively, also accepts `character()` with seasons in the format `"YYYYZZZZ"`, where `ZZZZ = YYYY + 1`, e.g. `"19951996"`. This is the format that ultimately gets sent to the NHL API. Some API endpoints, notably `seasons` exposed via `nhl_seasons()` also allow the value `"current"` to passed. This value will be returned unchanged.

Value

`data.frame`, with an element called `roster.roster` that in itself is a `data.frame` with the roster data.

Examples

```
## Not run:
 # Current rosters for all teams
 nhl_teams_rosters()

 # Rosters for all teams for past seasons
 nhl_teams_rosters(seasons = c("19931994", "19931994"))

 # Roster for Devils and Islanders
 nhl_teams_rosters(
   teamIds = 1:2,
   seasons = c("19931994", "19931994")
 )

## End(Not run)
```

`nhl_teams_shedule_next`

Get details for the teams' upcoming game

Description

Get details for the teams' upcoming game

Usage

```
nhl_teams_shedule_next(teamIds = NULL)
```

Arguments

`teamIds`	`integer()`, ids of the teams or `NULL` (default) for all teams. As of end of 2019, the valid team ids seem to be in the `1:54` range.

Value

`data.frame`, with elements with names starting with `nextGameSchedule` that contain data on the teams' upcoming game. One row per team.

Examples

```
## Not run:
 # Next game for all teams
 nhl_teams_shedule_next()

 # Next game for selected teams
 nhl_teams_shedule_next(c(1,3,5))

## End(Not run)
```

nhl_teams_shedule_previous *Get details for the teams' previous game*

Description

Get details for the teams' previous game

Usage

```
nhl_teams_shedule_previous(teamIds = NULL)
```

Arguments

teamIds	integer(), ids of the teams or NULL (default) for all teams. As of end of 2019, the valid team ids seem to be in the 1:54 range.

Value

data.frame, with elements with names starting with previousGameSchedule that contain data on the teams' previous game. One row per team.

Examples

```
 ## Not run:
  # Next game for all teams
  nhl_teams_shedule_previous()

  # Next game for selected teams
  nhl_teams_shedule_previous(c(1,3,5))

## End(Not run)
```

nhl_teams_stats *Get team statistics per seasons*

Description

Get team statistics per seasons

Usage

```
nhl_teams_stats(teamIds = NULL, seasons = NULL)
```

Arguments

teamIds	integer(), ids of the teams or NULL (default) for all teams. As of end of 2019, the valid team ids seem to be in the 1:54 range.
seasons	numeric(), integer() or character(), vector of starting years of desired seasons in YYYY format, e.g. 1995 or "1995" for season 1995-1996. Accepts vectors such as c(1995:2000, 2010) to generate multiple seasons. Alternatively, also accepts character() with seasons in the format "YYYYZZZZ", where ZZZZ = YYYY + 1, e.g. "19951996". This is the format that ultimately gets sent to the NHL API. Some API endpoints, notably seasons exposed via nhl_seasons() also allow the value "current" to passed. This value will be returned unchanged.

Value

data.frame, with seasons statistics for the selected team(s), one row per each team and season combination.

Examples

```
## Not run:
  # All teams, current seasons
  nhl_teams_stats()

  # 2 teams, 3 seasons
  nhl_teams_stats(1:2, c("20052006", "20062007", "20072008"))

## End(Not run)
```

---

nhl_tournaments *Retrieve data on tournaments from the API*

Description

Retrieve data on tournaments from the API

Usage

```
nhl_tournaments(tournamentTypes, seasons = NULL, expand = NULL)

nhl_tournaments_playoffs(seasons = NULL, expand = NULL)

nhl_tournaments_olympics(seasons = NULL, expand = NULL)

nhl_tournaments_worldcups(seasons = NULL, expand = NULL)
```

Arguments

tournamentTypes

character(), vector of one or more tournament types. Currently supported types seem to be

- "playoffs"
- "olympics"
- "worldCup"

Those are exposed via shorthand functions

- nhl_tournaments_playoffs()
- nhl_tournaments_olympics()
- nhl_tournaments_worldcups()

seasons

numeric(), integer() or character(), vector of starting years of desired seasons in YYYY format, e.g. 1995 or "1995" for season 1995-1996. Accepts vectors such as c(1995:2000, 2010) to generate multiple seasons.

Alternatively, also accepts character() with seasons in the format "YYYYZZZZ", where ZZZZ = YYYY + 1, e.g. "19951996". This is the format that ultimately gets sent to the NHL API.

Some API endpoints, notably seasons exposed via nhl_seasons() also allow the value "current" to passed. This value will be returned unchanged.

expand

character(), of parameters passed as expand to the API URL. Two valid examples seem to be "round.series" and "schedule.game.seriesSummary". NULL for no expand parameter.

Value

list, with information on tournaments, one element per tournamentTypes and parameters (seasons and expand) combinations.

Functions

- nhl_tournaments_playoffs: Shortcut to get information on playoffs.
- nhl_tournaments_olympics: Shortcut to get information on Olympics.
- nhl_tournaments_worldcups: Shortcut to get information on world cups.

Examples

```
## Not run:
  # Get info on playoffs in one season
  nhl_tournaments("playoffs", 2015)

  # Get info on playoffs in 2 seasons, expand rounds
  nhl_tournaments("playoffs", 2015:2016, "round.series")

## End(Not run)

## Not run:
```

```
  nhl_tournaments_playoffs(2015:2016, "round.series")

## End(Not run)
## Not run:
  nhl_tournaments_olympics(2009, "round.series")

## End(Not run)
## Not run:
  nhl_tournaments_worldcups(2003)

## End(Not run)
```

`nhl_url` *Create an NHL API URL*

Description

Create an NHL API URL

Usage

```
nhl_url(
  endPoint = NULL,
  suffixes = NULL,
  params = NULL,
  baseUrl = getOption("nhlapi_baseurl")
)
```

Arguments

`endPoint`	`character(1)`, the API endpoint.
`suffixes`	`list()`, of suffixes that will be concatenated to the end of the URLs, separated by `/`.
`params`	`named list()` of parameters that will be concatenated to the end of the URLs after `?`. Parameters can have multiple values, in which case multiple URLs are created. Multiple parameters are separated by `&`.
`baseUrl`	`character(1)`, URL of the NHL API base location.

Value

`character()`, the created URLs.

Examples

```
  nhlapi:::nhl_url("people", "8477474")
```

nhl_url_add_params *Add parameters to URLs*

Description

Add parameters to URLs

Usage

```
nhl_url_add_params(url, params = NULL)
```

Arguments

url	character(), vector of URLs.
params	named list() of parameters that will be concatenated to the end of the URLs after ?. Parameters can have multiple values, in which case multiple URLs are created. Multiple parameters are separated by &.

Value

character(), URLs with parameters added. Same length as all the combinations of url and params.

nhl_url_add_suffixes *Add suffixes to URLs*

Description

Add suffixes to URLs

Usage

```
nhl_url_add_suffixes(url, suffixes)
```

Arguments

url	character(), vector of URLs.
suffixes	list(), of suffixes that will be concatenated to the end of the URLs, separated by /.

Value

character(), URLs with suffixes added. Same length as all the combinations of url and suffixes.

nhl_url_awards *Create an NHL API URL for awards*

Description

Create an NHL API URL for awards

Usage

```
nhl_url_awards(awardIds = NULL)
```

Arguments

awardIds	integer(), vector of one or more award ids or NULL (default) for all awards. The current set of valid ids seems to be 1:24.

Value

character(), API URLs, same length as awardIds or length 1 if awardIds is NULL.

Examples

```
nhlapi:::nhl_url_awards()
nhlapi:::nhl_url_awards(1:3)
```

nhl_url_conferences *Create an NHL API URL for conferences*

Description

Create an NHL API URL for conferences

Usage

```
nhl_url_conferences(conferenceIds = NULL)
```

Arguments

conferenceIds	integer(), ids of the conferences or NULL (default) for all conferences As of end of 2019, the valid conference ids seem to be in the 1:7 range.

Value

character(), API URLs, same length as teamIds or length 1 if teamIds is NULL.

Examples

```
nhlapi:::nhl_url_conferences()
nhlapi:::nhl_url_conferences(1:3)
```

`nhl_url_divisions` *Create an NHL API URL for divisions*

Description

Create an NHL API URL for divisions

Usage

```
nhl_url_divisions(divisionIds = NULL)
```

Arguments

`divisionIds`	`integer()`, ids of the divisions or `NULL` (default) for all divisions. As of end of 2019, the valid division ids seem to be in the `1:25` range.

Value

`character()`, of same length as `teamIds` or length `1` if `teamIds` is `NULL`.

Examples

```
nhlapi:::nhl_url_divisions()
nhlapi:::nhl_url_divisions(1:3)
```

`nhl_url_drafts` *Create an NHL API URL for drafts*

Description

Create an NHL API URL for drafts

Usage

```
nhl_url_drafts(draftYears = NULL)
```

Arguments

`draftYears`	`integer()`, vector of one or more years in `YYYY` format or `NULL` (default) for the current year's draft. Also accepts a character vector of years in `YYYY` format.

Value

`character()`, API URLs, same length as `draftYears` or length `1` if `draftYears` is `NULL`.

Examples

```
nhlapi:::nhl_url_drafts()
nhlapi:::nhl_url_drafts(2015:2017)
```

`nhl_url_draft_prospects`

Create an NHL API URL for draft prospects

Description

Create an NHL API URL for draft prospects

Usage

```
nhl_url_draft_prospects(prospectIds = NULL)
```

Arguments

`prospectIds`	`integer()`, vector of one or more ids of draft prospects or `NULL` (default) for all exposed prospects.

Value

`character()`, API URLs, same length as `prospectIds` or length `1` if `prospectIds` is `NULL`.

Examples

```
nhlapi:::nhl_url_draft_prospects()
```

`nhl_url_games` *Create an NHL API URL for games*

Description

Create an NHL API URL for games

Usage

```
nhl_url_games(gameIds, element)
```

Arguments

`gameIds` `numeric()`, vector of one or more game ids. The game id is a 10 digit number where the

- first 4 digits identify the season of the game, for instance 2017 for the 2017-2018 season.
- next 2 digits give the type of game, where
  - 01 - preseason,
  - 02 - regular season,
  - 03 - playoffs,
  - 04 - all-star.
- final 4 digits identify the specific game number
  - for regular season and preseason games, this ranges from 0001 to the number of games played. That is 1271 for seasons with 31 teams and 1230 for seasons with 30 teams.
  - for playoff games, the
    - second digit gives the round of the playoffs
    - third digit specifies the match-up
    - fourth digit specifies the game (out of 7)

`element` `character()` vector of one or more valid elements. Currently the valid elements seem to be:

- `"linescore"`
- `"boxscore"`
- `"content"`
- `"feed/live"`

Value

`character()`, of same length as `gameIds`.

Examples

```
nhlapi:::nhl_url_games(2017010001, "content")
nhlapi:::nhl_url_games(
  c(2017010001, 2017010002),
  c("content", "boxscore")
)
```

`nhl_url_players` *Create an NHL API URL for players*

Description

Create an NHL API URL for players

Usage

```
nhl_url_players(playerIds)
```

Arguments

`playerIds`	`integer()`, vector of one or more ids of the players. The ids correspond to the ids expected by the NHL API people endpoint. For most cases the `playerNames` argument can be provided for more convenient usage.

Value

`character()`, API URLs, same length as `playerIds`.

Examples

```
nhlapi:::nhl_url_players(playerIds = c(8477474, 8477475))
```

`nhl_url_players_allseasons`
Create an NHL API URL for all players' seasons statistics

Description

Create an NHL API URL for all players' seasons statistics

Usage

```
nhl_url_players_allseasons(playerIds)
```

Arguments

playerIds	integer(), vector of one or more ids of the players. The ids correspond to the ids expected by the NHL API people endpoint. For most cases the playerNames argument can be provided for more convenient usage.

Examples

```
# Joe Sakic, all seasons
nhlapi:::nhl_url_players_allseasons(8451101L)
```

---

nhl_url_players_seasons

Create an NHL API URL for players' seasons statistics

Description

Create an NHL API URL for players' seasons statistics

Usage

```
nhl_url_players_seasons(playerIds, seasons, playoffs = FALSE)
```

Arguments

playerIds	integer(), vector of one or more ids of the players. The ids correspond to the ids expected by the NHL API people endpoint. For most cases the playerNames argument can be provided for more convenient usage.
seasons	numeric(), integer() or character(), vector of starting years of desired seasons in YYYY format, e.g. 1995 or "1995" for season 1995-1996. Accepts vectors such as c(1995:2000, 2010) to generate multiple seasons. Alternatively, also accepts character() with seasons in the format "YYYYZZZZ", where ZZZZ = YYYY + 1, e.g. "19951996". This is the format that ultimately gets sent to the NHL API. Some API endpoints, notably seasons exposed via nhl_seasons() also allow the value "current" to passed. This value will be returned unchanged.
playoffs	logical(1), if FALSE (default) get the regular seasons data, if TRUE, get the data for the playoffs.

Details

If multiple players and seasons are provided, URLs will be created for all combinations of players and seasons.

Examples

```
# Joe Sakic, regular season 1995/1996
nhlapi:::nhl_url_players_seasons(8451101L, 1995)

# Joe Sakic, playoffs 1995/1996, 1996/1997 and 1997/1998
nhlapi:::nhl_url_players_seasons(
  8451101L,
  1995:1997,
  playoffs = TRUE
 )
```

---

nhl_url_players_stats *Create an NHL API stats URL for players*

Description

Create an NHL API stats URL for players

Usage

```
nhl_url_players_stats(playerIds, params = NULL)
```

Arguments

playerIds	integer(), vector of one or more ids of the players. The ids correspond to the ids expected by the NHL API people endpoint. For most cases the playerNames argument can be provided for more convenient usage.
params	named list() of parameters that will be concatenated to the end of the URLs after ?. Parameters can have multiple values, in which case multiple URLs are created. Multiple parameters are separated by &.

Value

character(), of API URLs, same length as playerIds.

Examples

```
nhlapi:::nhl_url_players_stats(8477474)
```

`nhl_url_schedule` *Create an NHL API URL for schedules*

Description

Create an NHL API URL for schedules

Usage

```
nhl_url_schedule(
  seasons = NULL,
  teamIds = NULL,
  startDate = NULL,
  endDate = NULL,
  gameTypes = NULL,
  expand = NULL
)
```

Arguments

`seasons`	`numeric()`, `integer()` or `character()`, vector of starting years of desired seasons in `YYYY` format, e.g. `1995` or `"1995"` for season 1995-1996. Accepts vectors such as `c(1995:2000, 2010)` to generate multiple seasons. Alternatively, also accepts `character()` with seasons in the format `"YYYYZZZZ"`, where `ZZZZ = YYYY + 1`, e.g. `"19951996"`. This is the format that ultimately gets sent to the NHL API. Some API endpoints, notably `seasons` exposed via `nhl_seasons()` also allow the value `"current"` to passed. This value will be returned unchanged.
`teamIds`	`integer()`, ids of the teams or `NULL` (default) for all teams. As of end of 2019, the valid team ids seem to be in the `1:54` range.
`startDate`	`character(1)`, date in the format `"YYYY-MM-DD"` defining the start of the date interval for which the schedule is to be retrieved.
`endDate`	`character(1)`, date in the format `"YYYY-MM-DD"` defining the end of the date interval for which the schedule is to be retrieved.
`gameTypes`	`character()`, defining the game types to retrieve. Valid game types are for example `"R"` for regular season or `"P"` for playoffs. See `nhl_md_game_types()` for all values and their descriptions.
`expand`	`character()`, of parameters passed as `expand` to the API URL. Some valid examples seem to be `"round.series"` and `"schedule.broadcasts"`, `"schedule.linescore"`, `"schedule.ticket"`. `NULL` for no expand parameter.

Value

`character()`, vector of URLs.

Examples

```
nhlapi:::nhl_url_schedule(seasons = 2015:2016)
nhlapi:::nhl_url_schedule(
  startDate = "2018-01-02",
  endDate = "2018-01-02"
)
nhlapi:::nhl_url_schedule(
  startDate = "2018-01-02",
  endDate = "2018-01-02",
  teamIds = c(29, 30),
  expand = "schedule.linescore"
)
```

`nhl_url_seasons` *Create an NHL API URL for seasons*

Description

Create an NHL API URL for seasons

Usage

```
nhl_url_seasons(seasons = NULL)
```

Arguments

seasons	`numeric()`, `integer()` or `character()`, vector of starting years of desired seasons in `YYYY` format, e.g. `1995` or `"1995"` for season 1995-1996. Accepts vectors such as `c(1995:2000, 2010)` to generate multiple seasons. Alternatively, also accepts `character()` with seasons in the format `"YYYYZZZZ"`, where `ZZZZ = YYYY + 1`, e.g. `"19951996"`. This is the format that ultimately gets sent to the NHL API. Some API endpoints, notably `seasons` exposed via `nhl_seasons()` also allow the value `"current"` to passed. This value will be returned unchanged.

Value

`character()`, of API URLs, same length as `seasons` or length `1` if `seasons` is `NULL`.

Examples

```
nhlapi:::nhl_url_seasons()
nhlapi:::nhl_url_seasons(2015:2017)
nhlapi:::nhl_url_seasons("20152016")
```

`nhl_url_standings` *Create an NHL API URL for standings*

Description

Create an NHL API URL for standings

Usage

```
nhl_url_standings(seasons = NULL, standingsTypes = NULL, expand = NULL)
```

Arguments

`seasons`	`numeric()`, `integer()` or `character()`, vector of starting years of desired seasons in `YYYY` format, e.g. `1995` or `"1995"` for season 1995-1996. Accepts vectors such as `c(1995:2000, 2010)` to generate multiple seasons. Alternatively, also accepts `character()` with seasons in the format `"YYYYZZZZ"`, where `ZZZZ = YYYY + 1`, e.g. `"19951996"`. This is the format that ultimately gets sent to the NHL API. Some API endpoints, notably `seasons` exposed via `nhl_seasons()` also allow the value `"current"` to passed. This value will be returned unchanged.
`standingsTypes`	`character()`, defining the standings types to retrieve. Valid standings types are for example `"regularSeason"` or `"byDivision"`. See `nhl_md_standings_types()` for all values and their descriptions.
`expand`	`character()`, of parameters passed as `expand` to the API URL. A valid example seems to be `"standings.record"`. `NULL` for no expand parameter.

Value

`character()`, vector of URLs.

Examples

```
nhlapi:::nhl_url_standings(seasons = 2015:2016)
nhlapi:::nhl_url_standings(
  standingsType = "byDivision",
  expand = "standings.record"
)
```

nhl_url_teams *Create an NHL API URL for teams*

Description

Create an NHL API URL for teams

Usage

```
nhl_url_teams(teamIds = NULL, params = NULL)
```

Arguments

teamIds	integer(), ids of the teams or NULL (default) for all teams. As of end of 2019, the valid team ids seem to be in the 1:54 range.
params	named list() of parameters that will be concatenated to the end of the URLs after ?. Parameters can have multiple values, in which case multiple URLs are created. Multiple parameters are separated by &.

Value

character(), API URLs, same length as teamIds or length 1 if teamIds is NULL.

Examples

```
nhlapi:::nhl_url_teams()
nhlapi:::nhl_url_teams(1:3)
```

nhl_url_tournaments *Create an NHL API URL for tournaments*

Description

Create an NHL API URL for tournaments

Usage

```
nhl_url_tournaments(tournamentTypes, seasons = NULL, expand = NULL)
```

Arguments

tournamentTypes

character(), vector of one or more tournament types. Currently supported types seem to be

- "playoffs"
- "olympics"
- "worldCup"

Those are exposed via shorthand functions

- nhl_tournaments_playoffs()
- nhl_tournaments_olympics()
- nhl_tournaments_worldcups()

seasons

numeric(), integer() or character(), vector of starting years of desired seasons in YYYY format, e.g. 1995 or "1995" for season 1995-1996. Accepts vectors such as c(1995:2000, 2010) to generate multiple seasons.

Alternatively, also accepts character() with seasons in the format "YYYYZZZZ", where ZZZZ = YYYY + 1, e.g. "19951996". This is the format that ultimately gets sent to the NHL API.

Some API endpoints, notably seasons exposed via nhl_seasons() also allow the value "current" to passed. This value will be returned unchanged.

expand

character(), of parameters passed as expand to the API URL. Two valid examples seem to be "round.series" and "schedule.game.seriesSummary". NULL for no expand parameter.

Value

character(), API URLs, same length as combinations of tournamentTypes, seasons and expand.

See Also

nhl_md_tournament_types()

Examples

```
nhlapi:::nhl_url_tournaments("olympics")
nhlapi:::nhl_url_tournaments("playoffs", 2015:2016)
nhlapi:::nhl_url_tournaments("playoffs", 2015:2016, "round.series")
```

---

nhl_url_venues *Create an NHL API URL for venues*

Description

Create an NHL API URL for venues

Usage

```
nhl_url_venues(venueIds = NULL)
```

Arguments

venueIds	integer(), vector of one or more venue ids or NULL (default) for all currently exposed venues. The exported values seem incomplete, so it may be worth it to investigate other ids.

Value

character(), API URLs, same length as venueIds or length 1 if venueIds is NULL.

Examples

```
nhlapi:::nhl_url_venues()
nhlapi:::nhl_url_venues(5000:5006)
```

---

nhl_venues *Retrieve metadata on NHL venues from the API*

Description

Retrieve metadata on NHL venues from the API

Usage

```
nhl_venues(venueIds = NULL)
```

Arguments

venueIds	integer(), vector of one or more venue ids or NULL (default) for all currently exposed venues. The exported values seem incomplete, so it may be worth it to investigate other ids.

Value

data.frame, with information on venues, one row per venue.

Examples

```
## Not run:
  # Get information on currently exposed venues
  nhl_venues()

  # Get information on 3 historical venues
  nhl_venues(5000:5006)

## End(Not run)
```

util_attributes_to_cols

Add attributes as data frame columns

Description

Take attributes with names specified by `atrs` from object `lst` and adds their value into columns with the same name in `df`.

Usage

```
util_attributes_to_cols(lst, df, atrs = c("url", "copyright"))
```

Arguments

`lst`	`list`, with attributes to be added as columns to `df`.
`df`	`data.frame`, onto which new columns containing attributes of `lst` should be added.
`atrs`	`character()`, vector of names of attributes of `lst`.

Value

`data.frame`, `df` with added columns.

util_convert_minsonice

Convert "mm:ss" character to numeric minutes

Description

Convert `"mm:ss"` character to numeric minutes

Usage

```
util_convert_minsonice(chr, splitter = ":")
```

Arguments

`chr`	`character()`, vector in format `"mins:secs"`.
`splitter`	`character(1)`, that splits minutes and seconds in elements of `chr`.

Value

`numeric()`, vector of times in minutes. Same length as `chr`.

Examples

```
nhlapi:::util_convert_minsonice(c("20:00", "1500:30"))
```

util_generate_sysdata *Generate the* sysdata.rda *file*

Description

Generate the sysdata.rda file

Usage

```
util_generate_sysdata(playerIds = 8444849L:8490000L, tgtPath = "sysdata.rda")
```

Arguments

playerIds	integer(), vector of playerIds.
tgtPath	character(1), path where to save the generated object, NULL to not save.

Value

data.frame, with player name hashes and ids.

util_inherit_attributes *Inherit attributes from another object*

Description

Take attributes with names specified by atrs from object src and add them as the same attributes to tgt.

Usage

```
util_inherit_attributes(src, tgt, atrs = c("url", "copyright"))
```

Arguments

src	object, with attributes to be inherited by tgt.
tgt	object, onto which attributes of src should be added.
atrs	character(), vector of names of attributes of src to be added to tgt.

Value

object, same as tgt with attributes added.

util_map_player_id *Retrieve a player id from the name*

Description

Using a table of hashed names and ids, get a player id based on the name.

Usage

```
util_map_player_id(x, map = getOption("nhlapi_player_map"))
```

Arguments

x	`character(1)` a player's name, not case sensitive for convenience.
map	`data.frame`, with 2 columns: • `nameMd5`: `character()` of hashed player names • `id`: `integer()` of player ids used by the NHL API

Value

`integer(1)`, id of the player or `NA_integer` if not found.

Examples

```
 nhlapi:::util_map_player_id(
  "Joe Sakic",
  data.frame(
    nameMd5 = "9d2a915c8610dbc524c1bc800e010fcc",
    id = 19L,
    stringsAsFactors = FALSE
  )
)
```

util_map_player_ids *Retrieve a player ids from their names*

Description

Retrieve a player ids from their names

Usage

```
util_map_player_ids(playerNames, map = getOption("nhlapi_player_map"))
```

Arguments

playerNames	character(), vector of one or more player names. Not case sensitive for convenience.
map	data.frame, with 2 columns: • nameMd5: character() of hashed player names • id: integer() of player ids used by the NHL API

Value

integer(), named vector of player ids, 'NA_integer" for those names where id was not found. In case a player name has multiple ids, all of them are returned.

Examples

```
nhlapi:::util_map_player_ids(
  c("Joe SAKIC", "peter Forsberg", "test")
)
```

---

util_md5sum_str *Get MD5 hash for a character vector*

Description

Writes x to a temporary file using writeChar() and computes the md5sum() on that file, removing the file afterwards.

Usage

```
util_md5sum_str(x)
```

Arguments

x	character(), vector to compute the MD5 for.

Value

character(1), MD5 hash of a text file created from x using writeChar().

Examples

```
nhlapi:::util_md5sum_str("test")
```

`util_prepare_player_ids` *Prepare player ids based on player names*

Description

Prepare player ids based on player names

Usage

```
util_prepare_player_ids(playerNames, map = getOption("nhlapi_player_map"))
```

Arguments

`playerNames`	`character()`, vector of one or more player names. Not case sensitive for convenience.
`map`	`data.frame`, with 2 columns: • `nameMd5`: `character()` of hashed player names • `id`: `integer()` of player ids used by the NHL API

Value

`integer()`, named vector of found valid player ids, those not found omitted.

Examples

```
nhlapi:::util_prepare_player_ids(c("joe sakic", "fake player"))
```

`util_process_copyright` *Move copyright information to attribute*

Description

Removes the element named `el` from `x` if present and keeps the information as an equally named attribute.

Usage

```
util_process_copyright(x, el = "copyright")
```

Arguments

`x`	`list()`, to be processed.
`el`	`character(1)`, name of the element to remove. Defaults to `"copyright"` as this is the intended use of the function.

Value

`list`, with the `el` element removed and added as attribute, if it is present in `x`. Unchanged `x` otherwise.

`util_process_minsonice` *Convert time columns from* `"mm:ss"` *to numeric minutes*

Description

Convert time columns from `"mm:ss"` to numeric minutes

Usage

```
util_process_minsonice(df, patt = "timeOn|TimeOn")
```

Arguments

`df`	`data.frame`, data to examine.
`patt`	`character(1)`, pattern to match column names that contain time information in `"mm:ss"` format.

Value

`data.frame`, with time columns converted from `"mm:ss"` characters to numeric minutes.

`util_rbindlist` *Safely* `rbind` *multiple data.frames*

Description

Attempts to replace `do.call(rbind, lst)` taking into consideration that some data frames in `lst` can have missing columns. Those are filled by `NA` values.

Usage

```
util_rbindlist(lst, fill = TRUE)
```

Arguments

`lst`	`list()`, of data frames to be `rbind`-ed into one.
`fill`	`logical(1)`, if `FALSE`, this function just returns `do.call(rbind, lst)`.

Value

`data.frame`, the elements of `lst`, `rbind`-ed into one.

Examples

```
nhlapi:::util_rbindlist(list(
  datasets::mtcars[1, 2:3],
  datasets::mtcars[2, 4:5]
))
```

`util_report_get_data_errors` *Report errors encountered during* `nhl_get_data`

Description

Report errors encountered during `nhl_get_data`

Usage

```
util_report_get_data_errors(x, reporter = log_e, ...)
```

Arguments

`x`	`list`, results created by `nhl_get_data()`.
`reporter`	`function`, used to report the constructed error message, e.g. `message`, `warning`, `writeLines`, etc.
`...`	further arguments passed to `reporter`, e.g. `con = file("~/log.txt")` in case `writeLines` is the `reporter`.

Value

`character()`, URLs for which the retrieval resulted in an error, invisibly. Optional side-effects.

Examples

```
## Not run:
  # Write errors to a temporary text file
  tmpFile <- tempfile()
  util_report_get_data_errors(
    nhl_get_data(nhl_url_players(c("none", "8451101", "some"))),
    reporter = writeLines,
    con = tmpFile
  )

## End(Not run)
```

Index