

# Package ‘nhlapi’

May 25, 2020

**Type** Package

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

**Version** 0.1.2

**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** 6.0.1

**Suggests** testthat, roxygen2, knitr

**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.

**NeedsCompilation** no

**Author** Jozef Hajnala [aut, cre]

**Repository** CRAN

**Date/Publication** 2020-05-25 13:10:02 UTC

**R topics documented:**

make_log . . . . .	3
nhl_awards . . . . .	4
nhl_conferences . . . . .	5
nhl_divisions . . . . .	5
nhl_drafts . . . . .	6
nhl_draft_prospects . . . . .	7
nhl_from_json . . . . .	7
nhl_games . . . . .	8
nhl_get_data . . . . .	10
nhl_get_data_worker . . . . .	11
nhl_make_seasons . . . . .	12
nhl_md_event_types . . . . .	12
nhl_md_game_statuses . . . . .	13
nhl_md_game_types . . . . .	13
nhl_md_play_types . . . . .	13
nhl_md_standings_types . . . . .	14
nhl_md_stat_types . . . . .	14
nhl_md_tournament_types . . . . .	14
nhl_players . . . . .	15
nhl_players_allseasons . . . . .	15
nhl_players_seasons . . . . .	16
nhl_schedule . . . . .	17
nhl_seasons . . . . .	19
nhl_standings . . . . .	20
nhl_teams . . . . .	21
nhl_teams_rosters . . . . .	22
nhl_teams_schedule_next . . . . .	23
nhl_teams_schedule_previous . . . . .	24
nhl_teams_stats . . . . .	24
nhl_tournaments . . . . .	25
nhl_url . . . . .	27
nhl_url_add_params . . . . .	28
nhl_url_add_suffixes . . . . .	28
nhl_url_awards . . . . .	29
nhl_url_conferences . . . . .	29
nhl_url_divisions . . . . .	30
nhl_url_drafts . . . . .	30
nhl_url_draft_prospects . . . . .	31
nhl_url_games . . . . .	32
nhl_url_players . . . . .	33
nhl_url_players_allseasons . . . . .	33
nhl_url_players_seasons . . . . .	34
nhl_url_players_stats . . . . .	35
nhl_url_schedule . . . . .	36
nhl_url_seasons . . . . .	37
nhl_url_standings . . . . .	38

nhl_url_teams . . . . .	39
nhl_url_tournaments . . . . .	39
nhl_url_venues . . . . .	40
nhl_venues . . . . .	41
util_attributes_to_cols . . . . .	42
util_convert_minsonice . . . . .	42
util_generate_sysdata . . . . .	43
util_inherit_attributes . . . . .	43
util_map_player_id . . . . .	44
util_map_player_ids . . . . .	44
util_md5sum_str . . . . .	45
util_prepare_player_ids . . . . .	46
util_process_copyright . . . . .	46
util_process_minsonice . . . . .	47
util_rbindlist . . . . .	47
util_report_get_data_errors . . . . .	48

<b>Index</b>	<b>49</b>
--------------	-----------

---

make_log	<i>Create a log message</i>
----------	-----------------------------

---

## 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	<i>Retrieve metadata on NHL awards from the API</i>
------------	---

---

**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 current draft  
nhl_awards()  
  
# Get information on 3 historical awards  
nhl_awards(1:3)  
  
## End(Not run)
```

---

nhl_conferences	<i>Retrieve metadata on NHL conferences from the API</i>
-----------------	--

---

**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	<i>Retrieve metadata on NHL divisions from the API</i>
---------------	--

---

**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	<i>Retrieve metadata on NHL games from the API</i>
-----------	--

---

**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	<p>numeric(), vector of one or more game ids. The game id is a 10 digit number where the</p> <ul style="list-style-type: none"> <li>• first 4 digits identify the season of the game, for instance 2017 for the 2017-2018 season.</li> <li>• next 2 digits give the type of game, where <ul style="list-style-type: none"> <li>– 01 - preseason,</li> <li>– 02 - regular season,</li> </ul> </li> </ul>
---------	---



- 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, take-aways, 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"))

## End(Not run)

## Not run:
```

```

nhl_games_content(2017010001)

## End(Not run)
## Not run:
  nhl_games_feed(2017010001)

## End(Not run)
## Not run:
  nhl_games_boxscore(2017010001)

## End(Not run)
## Not run:
  nhl_games_linescore(2017010001)

## End(Not run)

```

---

nhl_get_data	<i>Get data from the API for one or more URLs</i>
--------------	---

---

## 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	<i>Make a vector of seasons consumable by the API</i>
------------------	---

---

**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	<p>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.</p> <p>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.</p> <p>Some API endpoints, notably seasons exposed via <a href="#">nhl_seasons()</a> also allow the value "current" to be passed. This value will be returned unchanged.</p>
---------	--

**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	<i>Get event types metadata</i>
--------------------	---------------------------------

---

**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	<i>Retrieve metadata for players based on names or ids</i>
-------------	--

---

**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	<i>Retrieve all seasons statistics for players</i>
------------------------	--

---

**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 <a href="#">nhl_seasons()</a> also allow the value "current" to be 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 playerName 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\_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 <a href="#">nhl_seasons()</a> also allow the value "current" to be 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 <a href="#">nhl_md_game_types()</a> 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 <a href="#">nhl_schedule()</a> .

**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 be passed. This value will be returned unchanged.

**Value**

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

**Examples**

```
## Not run:
# Get information on current draft
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 <code>nhl_seasons()</code> also allow the value "current" to be passed. This value will be returned unchanged.
<code>standingsTypes</code>	<code>character()</code> , defining the standings types to retrieve. Valid standings types are for example "regularSeason" or "byDivision". See <code>nhl_md_standings_types()</code> for all values and their descriptions.
<code>expand</code>	<code>character()</code> , 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**

<code>teamIds</code>	<code>integer()</code> , 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.
<code>params</code>	<code>named list()</code> , further parameters passed to <code>nhl_url_teams</code> .

**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_rovers	<i>Get rosters for teams</i>
------------------	------------------------------

---

**Description**

Get rosters for teams

**Usage**

```
nhl_teams_rovers(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 <a href="#">nhl_seasons()</a> also allow the value "current" to be 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_schedule_next
```

*Get details for the teams' upcoming game*

---

**Description**

Get details for the teams' upcoming game

**Usage**

```
nhl_teams_schedule_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_schedule_next()

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

## End(Not run)
```

nhl\_teams\_schedule\_previous

*Get details for the teams' previous game*

---

### Description

Get details for the teams' previous game

### Usage

```
nhl_teams_schedule_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_schedule_previous()  
  
# Next game for selected teams  
nhl_teams_schedule_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 <code>nhl_seasons()</code> also allow the value "current" to be 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	<i>Retrieve data on tournaments from the API</i>
-----------------	--

---

**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	<p>character(), vector of one or more tournament types. Currently supported types seem to be</p> <ul style="list-style-type: none"> <li>• "playoffs"</li> <li>• "olympics"</li> <li>• "worldCup"</li> </ul> <p>Those are exposed via shorthand functions</p> <ul style="list-style-type: none"> <li>• <code>nhl_tournaments_playoffs()</code></li> <li>• <code>nhl_tournaments_olympics()</code></li> <li>• <code>nhl_tournaments_worldcups()</code></li> </ul>
seasons	<p>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 <code>c(1995:2000, 2010)</code> to generate multiple seasons.</p> <p>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.</p> <p>Some API endpoints, notably seasons exposed via <code>nhl_seasons()</code> also allow the value "current" to be passed. This value will be returned unchanged.</p>
expand	<p>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.</p>

**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	<i>Create an NHL API URL</i>
---------	------------------------------

---

## 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	<i>Create an NHL API URL for awards</i>
----------------	---

---

**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	<i>Create an NHL API URL for conferences</i>
---------------------	--

---

**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**

- |                      |   |
|----------------------|---|
| <code>gameIds</code> | <p><code>numeric()</code>, vector of one or more game ids. The game id is a 10 digit number where the</p> <ul style="list-style-type: none"><li>• first 4 digits identify the season of the game, for instance 2017 for the 2017-2018 season.</li><li>• next 2 digits give the type of game, where<ul style="list-style-type: none"><li>– 01 - preseason,</li><li>– 02 - regular season,</li><li>– 03 - playoffs,</li><li>– 04 - all-star.</li></ul></li><li>• final 4 digits identify the specific game number<ul style="list-style-type: none"><li>– 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.</li><li>– for playoff games, the<ul style="list-style-type: none"><li>* second digit gives the round of the playoffs</li><li>* third digit specifies the match-up</li><li>* fourth digit specifies the game (out of 7)</li></ul></li></ul></li></ul> |
| <code>element</code> | <p><code>character()</code> vector of one or more valid elements. Currently the valid elements seem to be:</p> <ul style="list-style-type: none"><li>• "linescore"</li><li>• "boxscore"</li><li>• "content"</li><li>• "feed/live"</li></ul>   |

**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	<i>Create an NHL API URL for players</i>
-----------------	--

---

**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	<i>Create an NHL API URL for all players' seasons statistics</i>
----------------------------	--

---

**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 be 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 <a href="#">nhl_seasons()</a> also allow the value "current" to be 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 <a href="#">nhl_md_game_types()</a> 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	<i>Create an NHL API URL for seasons</i>
-----------------	--

---

## 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 be 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 <a href="#">nhl_seasons()</a> also allow the value "current" to be 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 <a href="#">nhl_md_standings_types()</a> 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	<p>character(), vector of one or more tournament types. Currently supported types seem to be</p> <ul style="list-style-type: none"> <li>• "playoffs"</li> <li>• "olympics"</li> <li>• "worldCup"</li> </ul> <p>Those are exposed via shorthand functions</p> <ul style="list-style-type: none"> <li>• <a href="#">nhl_tournaments_playoffs()</a></li> <li>• <a href="#">nhl_tournaments_olympics()</a></li> <li>• <a href="#">nhl_tournaments_worldcups()</a></li> </ul>
seasons	<p>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.</p> <p>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.</p> <p>Some API endpoints, notably seasons exposed via <a href="#">nhl_seasons()</a> also allow the value "current" to be passed. This value will be returned unchanged.</p>
expand	<p>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.</p>

**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 `attrs` from object `lst` and adds their value into columns with the same name in `df`.

### Usage

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

### Arguments

<code>lst</code>	list, with attributes to be added as columns to <code>df</code> .
<code>df</code>	data.frame, onto which new columns containing attributes of <code>lst</code> should be added.
<code>attrs</code>	character(), vector of names of attributes of <code>lst</code> .

### 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

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

### 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.

**Examples**

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

---

util_md5sum_str	<i>Get MD5 hash for a character vector</i>
-----------------	--

---

**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: <ul style="list-style-type: none"><li>• nameMd5: character() of hashed player names</li><li>• id: integer() of player ids used by the NHL API</li></ul>

### 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 <code>do.call(rbind, lst)</code> .

**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

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

make\_log, 3

nhl\_awards, 4  
nhl\_conferences, 5  
nhl\_divisions, 5  
nhl\_draft\_prospects, 7  
nhl\_drafts, 6  
nhl\_from\_json, 7  
nhl\_from\_json(), 11  
nhl\_games, 8  
nhl\_games\_boxscore (nhl\_games), 8  
nhl\_games\_content (nhl\_games), 8  
nhl\_games\_feed (nhl\_games), 8  
nhl\_games\_linescore (nhl\_games), 8  
nhl\_get\_data, 10  
nhl\_get\_data(), 48  
nhl\_get\_data\_worker, 11  
nhl\_get\_data\_worker(), 10  
nhl\_make\_seasons, 12  
nhl\_md\_event\_types, 12  
nhl\_md\_game\_statuses, 13  
nhl\_md\_game\_types, 13  
nhl\_md\_game\_types(), 18, 36  
nhl\_md\_play\_types, 13  
nhl\_md\_standings\_types, 14  
nhl\_md\_standings\_types(), 21, 38  
nhl\_md\_stat\_types, 14  
nhl\_md\_tournament\_types, 14  
nhl\_md\_tournament\_types(), 40  
nhl\_players, 15  
nhl\_players\_allseasons, 15  
nhl\_players\_seasons, 16  
nhl\_schedule, 17  
nhl\_schedule(), 18  
nhl\_schedule\_date\_range (nhl\_schedule), 17  
nhl\_schedule\_date\_range(), 17  
nhl\_schedule\_seasons (nhl\_schedule), 17  
nhl\_schedule\_seasons(), 17  
nhl\_schedule\_today (nhl\_schedule), 17  
nhl\_schedule\_today(), 17  
nhl\_seasons, 19  
nhl\_seasons(), 12, 16, 18, 20–22, 25, 26, 34, 36–38, 40  
nhl\_standings, 20  
nhl\_teams, 21  
nhl\_teams\_rosters, 22  
nhl\_teams\_schedule\_next, 23  
nhl\_teams\_schedule\_previous, 24  
nhl\_teams\_stats, 24  
nhl\_tournaments, 25  
nhl\_tournaments\_olympics (nhl\_tournaments), 25  
nhl\_tournaments\_olympics(), 26, 40  
nhl\_tournaments\_playoffs (nhl\_tournaments), 25  
nhl\_tournaments\_playoffs(), 26, 40  
nhl\_tournaments\_worldcups (nhl\_tournaments), 25  
nhl\_tournaments\_worldcups(), 26, 40  
nhl\_url, 27  
nhl\_url(), 11  
nhl\_url\_add\_params, 28  
nhl\_url\_add\_suffixes, 28  
nhl\_url\_awards, 29  
nhl\_url\_conferences, 29  
nhl\_url\_divisions, 30  
nhl\_url\_draft\_prospects, 31  
nhl\_url\_drafts, 30  
nhl\_url\_games, 32  
nhl\_url\_players, 33  
nhl\_url\_players\_allseasons, 33  
nhl\_url\_players\_seasons, 34  
nhl\_url\_players\_stats, 35  
nhl\_url\_schedule, 36  
nhl\_url\_seasons, 37  
nhl\_url\_standings, 38  
nhl\_url\_teams, 39  
nhl\_url\_tournaments, 39

nhl\_url\_venues, 40  
nhl\_venues, 41

util\_attributes\_to\_cols, 42  
util\_convert\_minsonice, 42  
util\_generate\_sysdata, 43  
util\_inherit\_attributes, 43  
util\_map\_player\_id, 44  
util\_map\_player\_ids, 44  
util\_md5sum\_str, 45  
util\_prepare\_player\_ids, 46  
util\_process\_copyright, 46  
util\_process\_minsonice, 47  
util\_rbindlist, 47  
util\_report\_get\_data\_errors, 48

writeChar(), 45