

Package ‘nhlscraper’

June 28, 2026

Title Scraper for National Hockey League Data

Version 0.7.0

Description

Scrapes and cleans data from the 'NHL' and 'ESPN' APIs into data.frames and lists. Wraps 125+ endpoints documented in <<https://github.com/RentoSaijo/nhlscraper/wiki>> from high-level multi-season summaries and award winners to low-level decisecond replays and bookmakers' odds, making them more accessible. Features cleaning and visualization tools, primarily for play-by-plays.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports http2 (>= 1.2.0), jsonlite (>= 2.0.0), xml2 (>= 1.5.0), arrow (>= 23.0.0), xgboost (>= 3.2.0)

Suggests testthat (>= 3.0.0), knitr (>= 1.50.0), rmarkdown (>= 2.29.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL <https://rentosaijo.github.io/nhlscraper/>,
<https://github.com/RentoSaijo/nhlscraper>

BugReports <https://github.com/RentoSaijo/nhlscraper/issues>

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 yes

Author Rento Saijo [aut, cre, cph] (ORCID:
<<https://orcid.org/0009-0008-4919-7349>>),
Lars Skytte [ctb],
Jack Pallotta [ctb] (ORCID: <<https://orcid.org/0009-0004-0744-6977>>)

Maintainer Rento Saijo <rentosaijo0527@gmail.com>

Repository CRAN

Date/Publication 2026-06-28 19:10:02 UTC

Contents

add_deltas	6
add_goalie_biometrics	7
add_shift_times	8
add_shooter_biometrics	8
attendance	9
awards	9
award_winners	10
boxscore	10
bracket	11
calculate_expected_goals	11
calculate_shift_times_by_situation	12
coaches	13
coach_career_statistics	14
coach_franchise_statistics	14
combine_reports	15
contracts	15
countries	16
drafts	16
draft_picks	17
draft_prospects	17
draft_rankings	18
draft_tracker	18
draw_NHL_rink	19
espn_futures	19
espn_games	20
espn_game_odds	20
espn_game_summary	21
espn_injuries	21
espn_players	22
espn_player_summary	22
espn_play_by_play	23
espn_teams	23
espn_team_summary	24
espn_transactions	24
expansion_drafts	25
expansion_draft_picks	25
franchises	26
franchise_playoff_situational_results	26
franchise_season_statistics	27
franchise_statistics	27
franchise_team_statistics	28
franchise_versus_franchise	28
games	29
game_odds	29
game_rosters	30
game_type_now	30

gc_play_by_play	31
gc_play_by_plays	31
gc_play_by_plays_raw	32
gc_play_by_play_raw	33
gc_summary	33
general_managers	34
get_attendance	34
get_awards	35
get_award_winners	35
get_bracket	35
get_configuration	36
get_countries	36
get_drafts	36
get_draft_picks	37
get_draft_rankings	37
get_draft_tracker	38
get_espn_athlete	38
get_espn_athletes	38
get_espn_coach	39
get_espn_coaches	39
get_espn_coach_career	39
get_espn_event	40
get_espn_events	40
get_espn_event_odds	40
get_espn_event_officials	41
get_espn_event_play_by_play	41
get_espn_event_stars	41
get_espn_futures	42
get_espn_injuries	42
get_espn_team	42
get_espn_teams	43
get_espn_transactions	43
get_franchises	43
get_franchise_season_by_season	44
get_franchise_team_totals	44
get_franchise_vs_franchise	45
get_games	45
get_game_boxscore	46
get_game_landing	46
get_game_story	47
get_gc_play_by_play	47
get_glossary	48
get_goalies	48
get_goalie_leaders	48
get_goalie_milestones	49
get_goalie_statistics	49
get_officials	49
get_partner_odds	50

get_players	50
get_player_game_log	51
get_player_landing	51
get_schedule	52
get_scoreboards	52
get_scores	53
get_seasons	53
get_season_now	53
get_series	54
get_series_schedule	54
get_shift_charts	54
get_skaters	55
get_skater_leaders	55
get_skater_milestones	56
get_skater_statistics	56
get_spotlight_players	56
get_standings	57
get_standings_information	57
get_streams	58
get_teams	58
get_team_prospects	58
get_team_roster	59
get_team_roster_statistics	59
get_team_schedule	60
get_team_scoreboard	60
get_team_seasons	61
get_team_statistics	61
get_tv_schedule	61
get_venues	62
get_wsc_play_by_play	62
glossary	63
goalie_edge_five_versus_five	63
goalie_edge_leaders	64
goalie_edge_save_percentage	65
goalie_edge_seasons	66
goalie_edge_shot_location	66
goalie_edge_summary	67
goalie_game_report	68
goalie_game_scoring	69
goalie_game_statistics	69
goalie_leaders	70
goalie_milestones	71
goalie_regular_statistics	71
goalie_report_configurations	72
goalie_scoring	72
goalie_season_report	73
goalie_season_statistics	74
goalie_series_statistics	74

goalie_statistics	75
ig_game_cumulative_expected_goals	75
ig_game_shot_locations	76
location	77
lottery_odds	77
nhlscraper	78
officials	79
penalty_shots	80
ping	80
players	81
player_game_log	81
player_seasons	82
player_summary	82
playoff_season_statistics	83
ps	83
replay	84
replays	84
roster	85
roster_statistics	86
schedule	87
scores	87
seasons	88
season_now	88
series	89
series_schedule	89
shifts	90
shift_chart	90
shift_charts	91
shift_chart_summaries	91
shift_chart_summary	92
skater_edge_leaders	92
skater_edge_seasons	93
skater_edge_shot_location	93
skater_edge_shot_speed	94
skater_edge_skating_distance	95
skater_edge_skating_speed	96
skater_edge_summary	97
skater_edge_zone_time	98
skater_game_report	99
skater_leaders	100
skater_milestones	100
skater_playoff_statistics	101
skater_regular_statistics	101
skater_report_configurations	102
skater_season_report	102
skater_season_statistics	103
skater_series_statistics	104
skater_statistics	104

spotlight_players	105
standings	105
standings_rules	106
streams	106
teams	107
team_edge_leaders	107
team_edge_seasons	108
team_edge_shot_location	108
team_edge_shot_speed	109
team_edge_skating_distance	110
team_edge_skating_speed	111
team_edge_summary	112
team_edge_zone_time	113
team_game_report	114
team_logos	115
team_month_schedule	115
team_prospects	116
team_report_configurations	117
team_seasons	117
team_season_report	118
team_season_schedule	119
team_season_statistics	119
team_week_schedule	120
tv_schedule	121
venues	121
wsc_play_by_play	122
wsc_play_by_plays	122
wsc_play_by_plays_raw	123
wsc_play_by_play_raw	124
wsc_summary	124
x_game_cumulative_expected_goals	125
x_game_shot_locations	126

Index	127
--------------	------------

add_deltas	<i>Add event-to-event deltas to a play-by-play</i>
------------	--

Description

add_deltas() adds event-to-event deltas in raw and normalized x/y, distance, and angle for a play-by-play. Sequences are bounded by faceoffs: each sequence begins at a faceoff, faceoff rows do not look backward across the boundary, and subsequent events are compared to the most recent prior valid-spatial event in the same faceoff-bounded sequence. Shootout and penalty-shot rows (0101/1010) are left as NA and do not serve as anchors for later rows. When multiple events in a sequence share the same recorded second, zero-time denominators used internally are replaced by $1 / n$, where n is the number of events in that same second within the sequence.

Usage

```
add_deltas(play_by_play)
```

Arguments

`play_by_play` data.frame of play-by-play(s) using the current public schema returned by [gc_play_by_play\(\)](#), [gc_play_by_plays\(\)](#), [wsc_play_by_play\(\)](#), or [wsc_play_by_plays\(\)](#)

Value

data.frame with one row per event (play) and added columns: `eventIdPrev`, `secondsElapsedInSequence`, `dSecondsElapsedInSequence`, `dXCoord`, `dYCoord`, `dXCoordNorm`, `dYCoordNorm`, `dDistance`, `dAngle`, `dXCoordPerSecond`, `dYCoordPerSecond`, `dXCoordNormPerSecond`, `dYCoordNormPerSecond`, `dDistancePerSecond`, `dAnglePerSecond`

`add_goalie_biometrics` *Add goalie biometrics to (a) play-by-play(s)*

Description

`add_goalie_biometrics()` adds goalie biometrics (height, weight, hand, and age at game date) to (a) play-by-play(s) for shot attempts. Goalie identity is resolved from `goalieInNetId` first and `goaliePlayerIdAgainst` second. If neither is available on a row, the added goalie biometrics are left as NA.

Usage

```
add_goalie_biometrics(play_by_play)
```

Arguments

`play_by_play` data.frame of play-by-play(s) using the current public schema returned by [gc_play_by_play\(\)](#), [gc_play_by_plays\(\)](#), [wsc_play_by_play\(\)](#), or [wsc_play_by_plays\(\)](#)

Value

data.frame with one row per event (play) and added columns: `goalieHeight`, `goalieWeight`, `goalieHandCode`, and `goalieAge`

add_shift_times	<i>Add on-ice shift times to a play-by-play</i>
-----------------	---

Description

add_shift_times() adds SecondsRemainingInShift, SecondsElapsedInShift, and SecondsElapsedInPeriodSinceLastShift columns for the on-ice goalies and skaters already present in a public play-by-play. It accepts either a single game play-by-play plus [shift_chart\(\)](#) data or a season aggregate plus [shift_charts\(\)](#) data.

Usage

```
add_shift_times(play_by_play, shift_chart)
```

Arguments

play_by_play	data.frame of play-by-play(s) using the current public schema returned by gc_play_by_play() , gc_play_by_plays() , wsc_play_by_play() , or wsc_play_by_plays()
shift_chart	data.frame returned by shift_chart() or shift_charts()

Value

data.frame with one row per event (play) and added or updated scalar on-ice shift timing columns

Examples

```
pbp <- gc_play_by_play(game = 2023030417)
sc <- shift_chart(game = 2023030417)
pbp <- add_shift_times(pbp, sc)
```

add_shooter_biometrics	<i>Add shooter biometrics to (a) play-by-play(s)</i>
------------------------	--

Description

add_shooter_biometrics() adds shooter biometrics (height, weight, hand, age at game date, and position) to (a) play-by-play(s) for shot attempts.

Usage

```
add_shooter_biometrics(play_by_play)
```

Arguments

play_by_play data.frame of play-by-play(s) using the current public schema returned by [gc_play_by_play\(\)](#), [gc_play_by_plays\(\)](#), [wsc_play_by_play\(\)](#), or [wsc_play_by_plays\(\)](#)

Value

data.frame with one row per event (play) and added columns: shooterHeight, shooterWeight, shooterHandCode, shooterAge, and shooterPositionCode

attendance	<i>Access the attendance by season and game type</i>
------------	--

Description

attendance() returns records-site attendance totals by season and game type.

Usage

attendance()

Value

data.frame with one row per season

Examples

```
all_attendance <- attendance()
```

awards	<i>Access all the awards</i>
--------	------------------------------

Description

awards() returns the records-site trophy catalog with one row per award and normalized trophy IDs, names, and short names.

Usage

awards()

Value

data.frame with one row per award

Examples

```
all_awards <- awards()
```

award_winners	<i>Access all the award winners/finalists</i>
---------------	---

Description

`award_winners()` returns historical trophy winners and finalists with one row per player/season/status entry.

Usage

```
award_winners()
```

Value

data.frame with one row per winner/finalist

Examples

```
all_award_winners <- award_winners()
```

boxscore	<i>Access the boxscore for a game, team, and position</i>
----------	---

Description

`boxscore()` returns one side of the GameCenter boxscore for one position group. Each row is a player with normalized name/position fields and the game-stat columns exposed for that group.

Usage

```
boxscore(game = 2023030417, team = "home", position = "forwards")
```

Arguments

game	integer ID (e.g., 2025020275); see games() for reference
team	character of 'h'/'home' or 'a'/'away'
position	character of 'f'/'forwards', 'd'/'defensemen', or 'g'/'goalies'

Value

data.frame with one row per player

Examples

```
boxscore_COL_forwards_Martin_Necas_legacy_game <- boxscore(
  game      = 2025020275,
  team      = 'H',
  position  = 'F'
)
```

bracket

*Access the playoff bracket for a season***Description**

`bracket()` returns the public playoff bracket for one season, with one row per series and normalized team, seed, score, and series-status fields.

Usage

```
bracket(season = season_now())
```

Arguments

`season` integer in YYYYYYYYY (e.g., 20242025); see [seasons\(\)](#) for reference

Value

data.frame with one row per series

Examples

```
bracket_20242025 <- bracket(season = 20242025)
```

calculate_expected_goals

*Calculate the expected goals for all the shots in (a) play-by-plays***Description**

`calculate_expected_goals()` scores shot events with `nhlscraper`'s rolling XGBoost expected-goals models. The package ships the small preprocessing bundle and downloads needed boosters from the companion NHLxG model store into a local user cache on first use. Each shot is routed to the matching season vintage and one of six game-state partitions: `sd` (5v5), `ev` (other even strength), `pp` (power play), `sh` (short-handed), `en` (empty net against), and `ps` (penalty shot; trained on penalty-shot and shootout-style rows). The `legacy_model` argument is accepted for backward compatibility but ignored.

Usage

```
calculate_expected_goals(play_by_play, model = NULL)
```

```
calculate_xG(play_by_play, model = NULL)
```

Arguments

play_by_play data.frame of play-by-play(s) using the current public schema returned by [gc_play_by_play\(\)](#), [gc_play_by_plays\(\)](#), [wsc_play_by_play\(\)](#), or [wsc_play_by_plays\(\)](#). Legacy alias-only columns such as typeDescKey, period, SOGFor, SOGAgainst, and SOGDifferential are no longer backfilled by the xG scorer.

model deprecated legacy model selector; ignored

Value

data.frame with one row per event (play) and added xG column

Examples

```
# May take >5s, so skip.

pbp <- gc_play_by_play()
pbp_with_xg <- calculate_expected_goals(play_by_play = pbp)
```

```
calculate_shift_times_by_situation
```

Calculate shift times by exact situation code

Description

`calculate_shift_times_by_situation()` returns one row per player per period and splits time on ice by the exact play-by-play situationCode in effect during that player's shifts. This differs from [shift_chart_summary\(\)](#), which reads the NHL HTML report's broad EV/PP/SH totals. Here, shift seconds come from [shift_chart\(\)](#) and strength state comes from [gc_play_by_play\(\)](#). When season is supplied, it overrides game and the function uses [gc_play_by_plays\(\)](#) with [shift_charts\(\)](#) for the full season.

Usage

```
calculate_shift_times_by_situation(game = 2023030417, season = NULL)
```

Arguments

game integer ID (e.g., 2023030417); ignored when season is supplied

season optional integer season ID (e.g., 20232024)

Details

In the play-by-play, the four `situationCode` digits mean away goalie count, away skater count, home skater count, and home goalie count. In this output, the four digits in each situation-specific column mean player-team goalie count, player-team skater count, opponent skater count, and opponent goalie count. For example, raw 1451 means home skaters are on a 5-on-4 power play; those seconds are written to `1541TimeOnIce` for home players and `1451TimeOnIce` for away players.

Value

data.frame with one row per player per period and situation-specific `TimeOnIce` columns

Examples

```
shift_situations_2023030417 <- calculate_shift_times_by_situation(  
  game = 2023030417  
)
```

`coaches`*Access all the coaches*

Description

`coaches()` returns the records-site coach registry with one row per coach and normalized coach IDs, names, and birth fields.

Usage

```
coaches()
```

Value

data.frame with one row per coach

Examples

```
all_coaches <- coaches()
```

`coach_career_statistics`*Access the career statistics for all the coaches*

Description

`coach_career_statistics()` returns regular-season plus playoff career coaching records with one row per coach.

Usage

```
coach_career_statistics()
```

```
coach_career_stats()
```

Value

data.frame with one row per coach

Examples

```
coach_career_stats <- coach_career_statistics()
```

`coach_franchise_statistics`*Access the statistics for all the coaches by franchise and game type*

Description

`coach_franchise_statistics()` returns coach records by franchise and game type, with one row per coach/franchise/game-type stint.

Usage

```
coach_franchise_statistics()
```

```
coach_franchise_stats()
```

Value

data.frame with one row per franchise per coach per game type

Examples

```
coach_franchise_stats <- coach_franchise_statistics()
```

combine_reports	<i>Access the draft combine reports</i>
-----------------	---

Description

combine_reports() returns draft-combine measurements and test results with one row per player/event result.

Usage

```
combine_reports()
```

Value

data.frame with one row per player

Examples

```
combine_reports <- combine_reports()
```

contracts	<i>Access all contracts from packaged internal data</i>
-----------	---

Description

contracts() loads preprocessed contract records bundled with the package, resolves each row to an NHL player ID when the player registry has a clear match, drops unresolved/ambiguous matches, and returns normalized contract seasons and money fields.

Usage

```
contracts()
```

Value

data.frame with one row per contract

Examples

```
all_contracts <- contracts()
```

`countries`*Access all the countries*

Description

`countries()` returns the stats API country catalog with one row per country and normalized two-/three-letter country fields.

Usage

```
countries()
```

Value

data.frame with one row per country

Examples

```
all_countries <- countries()
```

`drafts`*Access all the drafts*

Description

`drafts()` merges the records draft master and stats draft-round endpoints, returning one row per draft year with IDs, dates, location, and round metadata.

Usage

```
drafts()
```

Value

data.frame with one row per draft

Examples

```
all_drafts <- drafts()
```

draft_picks	<i>Access all the draft picks</i>
-------------	-----------------------------------

Description

draft_picks() returns the historical draft-pick table with one row per pick, including draft year/round/overall slot, team, player, position, and biographical fields.

Usage

```
draft_picks()
```

Value

data.frame with one row per pick

Examples

```
# May take >5s, so skip.  
all_draft_picks <- draft_picks()
```

draft_prospects	<i>Access all the draft prospects</i>
-----------------	---------------------------------------

Description

draft_prospects() returns the records-site prospect registry with one row per prospect and normalized prospect/player IDs, names, birth data, and profile fields.

Usage

```
draft_prospects()
```

Value

data.frame with one row per player

Examples

```
# May take >5s, so skip.  
all_prospects <- draft_prospects()
```

draft_rankings	<i>Access the draft rankings for a class and category</i>
----------------	---

Description

draft_rankings() returns NHL Central Scouting rankings for one draft class and skater/goalie region group, with one row per ranked prospect.

Usage

```
draft_rankings(class = season_now()%%10000, category = 1)
```

Arguments

class	integer in YYYY (e.g., 2017); see drafts() for reference
category	integer in 1:4 (where 1 = North American Skaters, 2 = International Skaters, 3 = North American Goalies, and 4 = International Goalies) OR character of 'NAS'/'NA Skaters'/'North American Skaters', 'INTLS'/'INTL Skaters'/'International Skaters', 'NAG'/'NA Goalies'/'North American Goalies', 'INTLG'/'INTL Goalies'/'International Goalies'

Value

data.frame with one row per player

Examples

```
draft_rankings_INTL_Skaters_2017 <- draft_rankings(
  class    = 2017,
  category = 2
)
```

draft_tracker	<i>Access the real-time draft tracker</i>
---------------	---

Description

draft_tracker() returns the live NHL draft tracker picks for the current draft, with normalized player, team, round, and pick fields.

Usage

```
draft_tracker()
```

Value

data.frame with one row per player

Examples

```
draft_tracker <- draft_tracker()
```

draw_NHL_rink	<i>Draw a full NHL rink</i>
---------------	-----------------------------

Description

draw_NHL_rink() draws a full NHL rink such that the x and y coordinates span -100 to 100 and -43 to +43, respectively. Use [graphics::points\(\)](#) to create custom graphs; check out an example on the online documentation!

Usage

```
draw_NHL_rink()
```

Value

NULL

Examples

```
draw_NHL_rink()
```

espn_futures	<i>Access the ESPN futures for a season</i>
--------------	---

Description

espn_futures() returns ESPN futures markets for one season with one row per future type and nested book/odds detail retained in ESPN's payload columns.

Usage

```
espn_futures(season = season_now())
```

Arguments

season integer in YYYYYYYYY (e.g., 20242025); see [seasons\(\)](#) for reference

Value

nested data.frame with one row per type (outer) and book (inner)

Examples

```
ESPN_futures_20252026 <- espn_futures(20252026)
```

espn_games	<i>Access the ESPN games for a season</i>
------------	---

Description

`espn_games()` pages ESPN's NHL event index for a season and returns one row per event containing the ESPN game ID used by the other ESPN wrappers.

Usage

```
espn_games(season = season_now())
```

Arguments

`season` integer in YYYYYYYY (e.g., 20242025); see [seasons\(\)](#) for reference

Value

data.frame with one row per ESPN game

Examples

```
ESPN_games_20242025 <- espn_games(season = 20242025)
```

espn_game_odds	<i>Access the ESPN odds for a game</i>
----------------	--

Description

`espn_game_odds()` returns ESPN's odds items for one game with one row per provider/market entry and camelCase names for nested price, spread, and total fields.

Usage

```
espn_game_odds(game = 401777460)
```

Arguments

`game` integer ID (e.g., 401777460); see [espn_games\(\)](#) for reference

Value

data.frame with one row per provider

Examples

```
ESPN_odds_SCF_20242025 <- espn_game_odds(game = 401777460)
```

espn_game_summary	<i>Access the ESPN summary for a game</i>
-------------------	---

Description

espn_game_summary() returns ESPN's competition summary for one game after dropping large availability/link/detail blocks, leaving the compact game, competitor, venue, date, format, and series fields exposed by the core API.

Usage

```
espn_game_summary(game = 401777460)
```

Arguments

game integer ID (e.g., 401777460); see [espn_games\(\)](#) for reference

Value

list with various items

Examples

```
ESPN_summary_SCF_20242025 <- espn_game_summary(game = 401777460)
```

espn_injuries	<i>Access the real-time ESPN injury reports</i>
---------------	---

Description

espn_injuries() returns ESPN's current injury report with one outer row per team and nested player injury details where ESPN includes them.

Usage

```
espn_injuries()
```

Value

nested data.frame with one row per team (outer) and player (inner)

Examples

```
ESPN_injuries_now <- espn_injuries()
```

espn_players	<i>Access all the ESPN players</i>
--------------	------------------------------------

Description

espn_players() pages ESPN's athlete index and returns one row per athlete containing the ESPN player ID used by [espn_player_summary\(\)](#).

Usage

```
espn_players()
```

Value

data.frame with one row per ESPN player

Examples

```
all_ESPN_players <- espn_players()
```

espn_player_summary	<i>Access the ESPN summary for a player</i>
---------------------	---

Description

espn_player_summary() returns a compact one-row ESPN athlete profile with name, birth, size, handedness, draft, debut, position, experience, and active status fields.

Usage

```
espn_player_summary(player = 3988803)
```

Arguments

player integer ID (e.g., 3988803); see [espn_players\(\)](#) for reference

Value

data.frame with one row

Examples

```
ESPN_summary_Charlie_McAvoy <- espn_player_summary(player = 3988803)
```

espn_play_by_play	<i>Access the ESPN play-by-play for a game</i>
-------------------	--

Description

`espn_play_by_play()` returns ESPN's play rows for one game with one row per event, normalized event IDs, period labels, clock/text fields, team refs, and coordinate fields when ESPN supplies them.

Usage

```
espn_play_by_play(game = 401777460)
```

```
espn_pbp(game = 401777460)
```

Arguments

`game` integer ID (e.g., 401777460); see [espn_games\(\)](#) for reference

Value

data.frame with one row per event (play)

Examples

```
ESPN_pbp_SCF_20242025 <- espn_play_by_play(game = 401777460)
```

espn_teams	<i>Access all the ESPN teams</i>
------------	----------------------------------

Description

`espn_teams()` pages ESPN's team index and returns one row per team containing the ESPN team ID used by [espn_team_summary\(\)](#).

Usage

```
espn_teams()
```

Value

data.frame with one row per ESPN team

Examples

```
all_ESPN_teams <- espn_teams()
```

espn_team_summary	<i>Access the ESPN summary for a team</i>
-------------------	---

Description

`espn_team_summary()` returns a compact one-row ESPN team profile with location, display names, abbreviation, and active status.

Usage

```
espn_team_summary(team = 1)
```

Arguments

team	integer ID (e.g., 1); see espn_teams() for reference
------	--

Value

data.frame with one row

Examples

```
ESPN_summary_Boston_Bruins <- espn_team_summary(team = 1)
```

espn_transactions	<i>Access the ESPN transactions for a season</i>
-------------------	--

Description

`espn_transactions()` pages ESPN's transaction feed across a season window and returns one row per transaction with normalized team fields.

Usage

```
espn_transactions(season = season_now())
```

Arguments

season	integer in YYYYYYYYY (e.g., 20242025); the summer of the latter year is included
--------	--

Value

data.frame with one row per transaction

Examples

```
ESPN_transactions_20242025 <- espn_transactions(season = 20242025)
```

expansion_drafts	<i>Access all the expansion drafts</i>
------------------	--

Description

expansion_drafts() returns expansion-draft rule records by season, including the rule text and season identifiers exposed by the records endpoint.

Usage

```
expansion_drafts()
```

Value

data.frame with one row per expansion draft

Examples

```
all_expansion_drafts <- expansion_drafts()
```

expansion_draft_picks	<i>Access all the expansion draft picks</i>
-----------------------	---

Description

expansion_draft_picks() returns historical expansion-draft picks with one row per selected player/team/season combination.

Usage

```
expansion_draft_picks()
```

Value

data.frame with one row per pick

Examples

```
all_expansion_draft_picks <- expansion_draft_picks()
```

franchises	<i>Access all the franchises</i>
------------	----------------------------------

Description

franchises() merges records-site franchise and franchise-detail metadata, returning one row per franchise with normalized franchise/team identifiers and names.

Usage

```
franchises()
```

Value

data.frame with one row per franchise

Examples

```
all_franchises <- franchises()
```

franchise_playoff_situational_results	<i>Access the playoff series results for all the franchises by situation</i>
---------------------------------------	--

Description

franchise_playoff_situational_results() returns playoff series records by franchise and series situation, such as leading or trailing a series.

Usage

```
franchise_playoff_situational_results()
```

Value

data.frame with one row per franchise per situation

Examples

```
franchise_playoff_situational_results <-  
  franchise_playoff_situational_results()
```

`franchise_season_statistics`*Access the statistics for all the franchises by season and game type*

Description

`franchise_season_statistics()` returns records-site franchise results by season and game type, including team, standing, record, goals, and points fields.

Usage

```
franchise_season_statistics()
```

```
franchise_season_stats()
```

Value

data.frame with one row per franchise per season per game type

Examples

```
# May take >5s, so skip.  
franchise_season_stats <- franchise_season_statistics()
```

`franchise_statistics` *Access the all-time statistics for all the franchises by game type*

Description

`franchise_statistics()` returns all-time franchise totals by game type, including games, wins/losses, goals, points, and related aggregate fields.

Usage

```
franchise_statistics()
```

```
franchise_stats()
```

Value

data.frame with one row per franchise per game type

Examples

```
franchise_stats <- franchise_statistics()
```

franchise_team_statistics

Access the all-time statistics for all the franchises by team and game type

Description

franchise_team_statistics() returns all-time totals by franchise-era team and game type, preserving separate rows for teams that share a franchise.

Usage

```
franchise_team_statistics()
```

```
franchise_team_stats()
```

Value

data.frame with one row per team per franchise per game type

Examples

```
franchise_team_stats <- franchise_team_statistics()
```

franchise_versus_franchise

Access the all-time statistics versus other franchises for all the franchises by game type

Description

franchise_versus_franchise() returns all-time head-to-head records by franchise, opponent franchise, and game type.

Usage

```
franchise_versus_franchise()
```

```
franchise_vs_franchise()
```

Value

data.frame with one row per franchise per franchise per game type

Examples

```
# May take >5s, so skip.  
franchise_vs_franchise <- franchise_versus_franchise()
```

games	<i>Access all the games</i>
-------	-----------------------------

Description

games() returns the stats API game catalog with one row per NHL game, ordered by gameId and normalized to gameId, seasonId, and gameId.

Usage

```
games()
```

Value

data.frame with one row per game

Examples

```
# May take >5s, so skip.  
all_games <- games()
```

game_odds	<i>Access the real-time game odds for a country by partnered bookmaker</i>
-----------	--

Description

game_odds() returns the public partner-game feed for one country, with one row per current game and normalized team, game type, bookmaker, and odds fields when NHL exposes them.

Usage

```
game_odds(country = "US")
```

Arguments

country	two-letter code (e.g., 'CA'); see countries() for reference
---------	---

Value

data.frame with one row per game

Examples

```
game_odds_CA <- game_odds(country = 'CA')
```

game_rosters	<i>Access the rosters for a game</i>
--------------	--------------------------------------

Description

game_rosters() returns the roster spots from a GameCenter play-by-play payload with one row per dressed player, including team ID, sweater number, position, and normalized player name fields.

Usage

```
game_rosters(game = 2023030417)
```

Arguments

game integer ID (e.g., 2025020275); see [games\(\)](#) for reference

Value

data.frame with one row per player

Examples

```
rosters_Martin_Necas_legacy_game <- game_rosters(game = 2025020275)
```

game_type_now	<i>Access the game type as of now</i>
---------------	---------------------------------------

Description

game_type_now() retrieves the game type as of now and returns a scalar integer used as the current-context default in season/game-type dependent wrappers.

Usage

```
game_type_now()
```

Value

integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season)

Examples

```
game_type_now <- game_type_now()
```

gc_play_by_play	<i>Access the GameCenter (GC) play-by-play for a game</i>
-----------------	---

Description

gc_play_by_play() retrieves the GameCenter (GC) play-by-play for a game as a data.frame where each row represents an event. The returned schema is the cleaned, public-facing play-by-play schema, including canonical names such as periodNumber, eventTypeCode, eventTypeDescKey, homeShots, shotsFor, penaltyTypeDescKey, penaltyDuration, servedByPlayerId, goalieInNetId, and HTML-report-derived on-ice player ID columns such as homeGoaliePlayerId, awayGoaliePlayerId, homeSkater1PlayerId, and any additional overflow skater slots required by the game. HTML report skater and goalie IDs are returned whenever they can be matched back to a supported row, even when the raw situationCode is stale. Use add_shift_times() with shift_chart() (or shift_charts()) to add on-ice shift timing columns.

Usage

```
gc_play_by_play(game = 2023030417)

gc_pbp(game = 2023030417)
```

Arguments

game integer ID (e.g., 2025020275); see games() for reference

Value

data.frame with one row per event (play)

Examples

```
# May take >5s, so skip.
gc_pbp_Martin_Necas_legacy_game <- gc_play_by_play(game = 2025020275)
```

gc_play_by_plays	<i>Access the GameCenter (GC) play-by-plays for a season</i>
------------------	--

Description

gc_play_by_plays() downloads the cleaned season-level GameCenter parquet snapshot and pads situationCode to the four-character public format.

Usage

```
gc_play_by_plays(season = 20242025)

gc_pbps(season = 20242025)
```

Arguments

season integer in YYYYYYYYY (e.g., 20242025); see [seasons\(\)](#) for reference

Value

data.frame with one row per event (play) per game

Examples

```
# May take >5s, so skip.
gc_pbps_20212022 <- gc_play_by_plays(season = 20212022)
```

`gc_play_by_plays_raw` *Access the raw GameCenter (GC) play-by-plays for a season*

Description

`gc_play_by_plays_raw()` downloads the stored season-level GameCenter parquet snapshot and returns the raw rows without public-schema cleanup or situation-code padding.

Usage

```
gc_play_by_plays_raw(season = 20242025)

gc_pbps_raw(season = 20242025)
```

Arguments

season integer in YYYYYYYYY (e.g., 20242025); see [seasons\(\)](#) for reference

Value

data.frame with one row per raw event (play) per game

Examples

```
# May take >5s, so skip.
gc_pbps_raw_20212022 <- gc_play_by_plays_raw(season = 20212022)
```

gc_play_by_play_raw	Access the raw GameCenter (GC) play-by-play for a game
---------------------	--

Description

gc_play_by_play_raw() returns the raw flattened GameCenter play-by-play as served by the NHL API for one game. Use [gc_play_by_play\(\)](#) for the cleaned public schema that repairs common clock/order defects and appends the derived public columns.

Usage

```
gc_play_by_play_raw(game = 2023030417)
```

```
gc_pbp_raw(game = 2023030417)
```

Arguments

game integer ID (e.g., 2025020275); see [games\(\)](#) for reference

Value

data.frame with one row per event (play)

Examples

```
gc_raw_Martin_Necas_legacy_game <- gc_play_by_play_raw(game = 2025020275)
```

gc_summary	Access the GameCenter (GC) summary for a game
------------	---

Description

gc_summary() combines the GameCenter landing and right-rail payloads for one game. The nested list includes the scoreboard, team records, scoring summary, three stars, game videos, officials, season/game identifiers, and related GameCenter metadata when available.

Usage

```
gc_summary(game = 2023030417)
```

Arguments

game integer ID (e.g., 2025020275); see [games\(\)](#) for reference

Value

list of various items

Examples

```
gc_summary_Martin_Necas_legacy_game <- gc_summary(game = 2025020275)
```

general_managers	<i>Access all the general managers</i>
------------------	--

Description

general_managers() returns the records-site general-manager registry with one row per general manager and normalized ID/name fields.

Usage

```
general_managers()
gms()
```

Value

data.frame with one row per general manager

Examples

```
all_GMs <- general_managers()
```

get_attendance	<i>Access the attendance by season and game type</i>
----------------	--

Description

get_attendance() is deprecated. Use [attendance\(\)](#) instead.

Usage

```
get_attendance()
```

Value

data.frame with one row per season

get_awards	<i>Access all the awards</i>
------------	------------------------------

Description

get_awards() is deprecated. Use [awards\(\)](#) instead.

Usage

```
get_awards()
```

Value

data.frame with one row per award

get_award_winners	<i>Access all the award winners/finalists</i>
-------------------	---

Description

get_award_winners() is deprecated. Use [award_winners\(\)](#) instead.

Usage

```
get_award_winners()
```

Value

data.frame with one row per winner/finalist

get_bracket	<i>Access the playoff bracket for a season</i>
-------------	--

Description

get_bracket() is deprecated. Use [bracket\(\)](#) instead.

Usage

```
get_bracket(season = season_now())
```

Arguments

season	integer in YYYYYYYYY (e.g., 20242025); see seasons() for reference
--------	--

Value

data.frame with one row per series

get_configuration	<i>Access the configurations for team, skater, and goalie reports</i>
-------------------	---

Description

get_configuration() is defunct. Use [team_report_configurations\(\)](#), [skater_report_configurations\(\)](#), and/or [goalie_report_configurations\(\)](#) instead.

Usage

```
get_configuration()
```

get_countries	<i>Access all the countries</i>
---------------	---------------------------------

Description

get_countries() is deprecated. Use [countries\(\)](#) instead.

Usage

```
get_countries()
```

Value

data.frame with one row per country

get_drafts	<i>Access all the drafts</i>
------------	------------------------------

Description

get_drafts() is deprecated. Use [drafts\(\)](#) instead.

Usage

```
get_drafts()
```

Value

data.frame with one row per draft

get_draft_picks	<i>Access all the draft picks</i>
-----------------	-----------------------------------

Description

get_draft_picks() is deprecated. Use [draft_picks\(\)](#) instead.

Usage

```
get_draft_picks()
```

Value

data.frame with one row per pick

get_draft_rankings	<i>Access the draft rankings for a year and player type</i>
--------------------	---

Description

get_draft_rankings() is deprecated. Use [draft_rankings\(\)](#) instead.

Usage

```
get_draft_rankings(year = season_now()%%10000, player_type = 1)
```

Arguments

year	integer in YYYY (e.g., 2017); see drafts() for reference
player_type	integer in 1:4 (where 1 = North American Skaters, 2 = International Skaters, 3 = North American Goalies, and 4 = International Goalies)

Value

data.frame with one row per player

get_draft_tracker	<i>Access the real-time draft tracker</i>
-------------------	---

Description

get_draft_tracker() is deprecated. Use [draft_tracker\(\)](#) instead.

Usage

```
get_draft_tracker()
```

Value

data.frame with one row per player

get_espn_athlete	<i>Access the ESPN summary for an athlete (player) and season</i>
------------------	---

Description

get_espn_athlete() is defunct. Use [espn_player_summary\(\)](#) instead.

Usage

```
get_espn_athlete()
```

get_espn_athletes	<i>Access all the ESPN athletes (players)</i>
-------------------	---

Description

get_espn_athletes() is deprecated. Use [espn_players\(\)](#) instead.

Usage

```
get_espn_athletes()
```

Value

data.frame with one row per ESPN athlete (player)

get_espn_coach	<i>Access the ESPN statistics for a coach and (multiple) season(s)</i>
----------------	--

Description

get_espn_coach() is defunct. Use [coach_career_statistics\(\)](#) instead.

Usage

```
get_espn_coach()
```

get_espn_coaches	<i>Access the ESPN coaches for a season</i>
------------------	---

Description

get_espn_coaches() is defunct. Use [coaches\(\)](#) instead.

Usage

```
get_espn_coaches()
```

get_espn_coach_career	<i>Access the career ESPN statistics for a coach</i>
-----------------------	--

Description

get_espn_coach_career() is defunct. Use [coach_career_statistics\(\)](#) instead.

Usage

```
get_espn_coach_career()
```

get_espn_event	<i>Access the ESPN summary for an event (game)</i>
----------------	--

Description

get_espn_event() is deprecated. Use [espn_game_summary\(\)](#) instead.

Usage

```
get_espn_event(event = 401777460)
```

Arguments

event integer ID (e.g., 401777460); see [espn_games\(\)](#) for reference

Value

data.frame with one row per event (game)

get_espn_events	<i>Access the ESPN events (games) by start and end dates</i>
-----------------	--

Description

get_espn_events() is defunct. Use [espn_games\(\)](#) instead.

Usage

```
get_espn_events()
```

get_espn_event_odds	<i>Access the ESPN odds for an event (game)</i>
---------------------	---

Description

get_espn_event_odds() is deprecated. Use [espn_game_odds\(\)](#) instead.

Usage

```
get_espn_event_odds(event = 401777460)
```

Arguments

event integer ID (e.g., 401777460); see [espn_games\(\)](#) for reference

Value

data.frame with one row per provider

get_espn_event_officials

Access the officials for an ESPN event (game)

Description

get_espn_event_officials() is defunct. Use [gc_summary\(\)](#) and/or [wsc_summary\(\)](#) instead.

Usage

get_espn_event_officials()

get_espn_event_play_by_play

Access the ESPN play-by-play for an event (game)

Description

get_espn_event_play_by_play() is deprecated. Use [espn_play_by_play\(\)](#) instead.

Usage

get_espn_event_play_by_play(event = 401777460)

Arguments

event integer ID (e.g., 401777460); see [espn_games\(\)](#) for reference

Value

data.frame with one row per play

get_espn_event_stars

Access the three stars for an ESPN event (game)

Description

get_espn_event_stars() is defunct. Use [gc_summary\(\)](#) and/or [wsc_summary\(\)](#) instead.

Usage

get_espn_event_stars()

get_espn_futures	<i>Access the ESPN futures for a season</i>
------------------	---

Description

get_espn_futures() is defunct. Use [espn_futures\(\)](#) instead.

Usage

```
get_espn_futures()
```

get_espn_injuries	<i>Access the real-time ESPN injury reports</i>
-------------------	---

Description

get_espn_injuries() is deprecated. Use [espn_injuries\(\)](#) instead.

Usage

```
get_espn_injuries()
```

Value

nested data.frame with one row per team (outer) and player (inner)

get_espn_team	<i>Access the ESPN summary for a team and season</i>
---------------	--

Description

get_espn_team() is defunct. Use [espn_team_summary\(\)](#) instead.

Usage

```
get_espn_team()
```

get_espn_teams	<i>Access all the ESPN teams for a season</i>
----------------	---

Description

get_espn_teams() is defunct. Use [espn_teams\(\)](#) instead.

Usage

```
get_espn_teams()
```

get_espn_transactions	<i>Access the ESPN transactions by start and end dates</i>
-----------------------	--

Description

get_espn_transactions() is defunct. Use [espn_transactions\(\)](#) instead.

Usage

```
get_espn_transactions()
```

get_franchises	<i>Access all the franchises</i>
----------------	----------------------------------

Description

get_franchises() is deprecated. Use [franchises\(\)](#) instead.

Usage

```
get_franchises()
```

Value

data.frame with one row per franchise

`get_franchise_season_by_season`*Access the statistics for all the franchises by season and game type*

Description

`get_franchise_season_by_season()` is deprecated. Use [franchise_season_statistics\(\)](#) instead.

Usage

```
get_franchise_season_by_season()
```

Value

data.frame with one row per franchise per season per game type

`get_franchise_team_totals`*Access the all-time statistics for all the franchises by team and game type*

Description

`get_franchise_team_totals()` is deprecated. Use [franchise_team_statistics\(\)](#) instead.

Usage

```
get_franchise_team_totals()
```

Value

data.frame with one row per team per franchise per game type

`get_franchise_vs_franchise`*Access the all-time statistics versus other franchises for all the franchises by game type*

Description

`get_franchise_vs_franchise()` is deprecated. Use [franchise_versus_franchise\(\)](#) instead.

Usage

```
get_franchise_vs_franchise()
```

Value

data.frame with one row per franchise per franchise per game type

`get_games`*Access all the games*

Description

`get_games()` is deprecated. Use [games\(\)](#) instead.

Usage

```
get_games()
```

Value

data.frame with one row per game

get_game_boxscore	<i>Access the boxscore for a game, team, and player type</i>
-------------------	--

Description

get_game_boxscore() is deprecated. Use [boxscore\(\)](#) instead.

Usage

```
get_game_boxscore(game = 2023030417, team = "home", player_type = "forwards")
```

Arguments

game	integer ID (e.g., 2025020275); see games() for reference
team	character of 'home' or 'away'
player_type	character of 'forwards', 'defense', or 'goalies'

Value

data.frame with one row per player

get_game_landing	<i>Access the GameCenter (GC) summary for a game</i>
------------------	--

Description

get_game_landing() is deprecated. Use [gc_summary\(\)](#) instead.

Usage

```
get_game_landing(game = 2023030417)
```

Arguments

game	integer ID (e.g., 2025020275); see games() for reference
------	--

Value

list of various items

get_game_story	Access the World Showcase (WSC) summary for a game
----------------	--

Description

get_game_story() is deprecated. Use [wsc_summary\(\)](#) instead.

Usage

```
get_game_story(game = 2023030417)
```

Arguments

game	integer ID (e.g., 2025020275); see games() for reference
------	--

Value

list of various items

get_gc_play_by_play	Access the GameCenter (GC) play-by-play for a game
---------------------	--

Description

get_gc_play_by_play() is deprecated. Use [gc_play_by_play\(\)](#) instead.

Usage

```
get_gc_play_by_play(game = 2023030417)
```

Arguments

game	integer ID (e.g., 2025020275); see games() for reference
------	--

Value

data.frame with one row per event (play)

get_glossary	Access the glossary
--------------	---------------------

Description

get_glossary() is deprecated. Use [glossary\(\)](#) instead.

Usage

```
get_glossary()
```

Value

data.frame with one row per terminology

get_goalies	Access all the goalies for a range of seasons
-------------	---

Description

get_goalies() is defunct. Use [players\(\)](#) instead.

Usage

```
get_goalies()
```

get_goalie_leaders	Access the goalie statistics leaders for a season, game type, and category
--------------------	--

Description

get_goalie_leaders() is deprecated. Use [goalie_leaders\(\)](#) instead.

Usage

```
get_goalie_leaders(season = "current", game_type = "", category = "wins")
```

Arguments

- | | |
|-----------|---|
| season | integer in YYYYYYYYY (e.g., 20242025); see seasons() for reference |
| game_type | integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season)
OR character of 'pre', 'regular', or 'playoff'/'post'; see seasons() for reference;
most functions will NOT support pre-season |
| category | character of 'wins', 'shutouts', 'savePctg', or 'goalsAgainstAverage' |

Value

data.frame with one row per player

get_goalie_milestones *Access the goalies on milestone watch*

Description

get_goalie_milestones() is deprecated. Use [goalie_milestones\(\)](#) instead.

Usage

```
get_goalie_milestones()
```

Value

data.frame with one row per player

get_goalie_statistics *Access various reports for all the goalies by season or game*

Description

get_goalie_statistics() is defunct. Use [goalie_season_report\(\)](#) or [goalie_game_report\(\)](#) instead.

Usage

```
get_goalie_statistics()
```

get_officials *Access all the officials*

Description

get_officials() is deprecated. Use [officials\(\)](#) instead.

Usage

```
get_officials()
```

Value

data.frame with one row per official

get_partner_odds	<i>Access the real-time game odds for a country by partnered bookmaker</i>
------------------	--

Description

get_partner_odds() is deprecated. Use [game_odds\(\)](#) instead.

Usage

```
get_partner_odds(country = "US")
```

Arguments

country two-letter code (e.g., 'CA'); see [countries\(\)](#) for reference

Value

data.frame with one row per game

get_players	<i>Access all the players</i>
-------------	-------------------------------

Description

get_players() is deprecated. Use [players\(\)](#) instead.

Usage

```
get_players()
```

Value

data.frame with one row per player

get_player_game_log	<i>Access the game log for a player, season, and game type</i>
---------------------	--

Description

get_player_game_log() is deprecated. Use [player_game_log\(\)](#) instead.

Usage

```
get_player_game_log(player = 8478402, season = "now", game_type = "")
```

Arguments

- player integer ID (e.g., 8480039); see [players\(\)](#) for reference
- season integer in YYYYYYYY (e.g., 20242025); see [seasons\(\)](#) for reference
- game_type integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see [seasons\(\)](#) for reference; most functions will NOT support pre-season

Value

data.frame with one row per game

get_player_landing	<i>Access the summary for a player</i>
--------------------	--

Description

get_player_landing() is deprecated. Use [player_summary\(\)](#) instead.

Usage

```
get_player_landing(player = 8478402)
```

Arguments

- player integer ID (e.g., 8480039); see [players\(\)](#) for reference

Value

list with various items

get_schedule	<i>Access the schedule for a date</i>
--------------	---------------------------------------

Description

get_schedule() is deprecated. Use [schedule\(\)](#) instead.

Usage

```
get_schedule(date = "2025-01-01")
```

Arguments

date	character in 'YYYY-MM-DD' (e.g., '2025-01-01'); see seasons() for reference
------	---

Value

data.frame with one row per game

get_scoreboards	<i>Access the scoreboards for a date</i>
-----------------	--

Description

get_scoreboards() is deprecated. Use [scores\(\)](#) instead.

Usage

```
get_scoreboards(date = "now")
```

Arguments

date	character in 'YYYY-MM-DD' (e.g., '2025-01-01'); see seasons() for reference
------	---

Value

data.frame with one row per game

get_scores	<i>Access the scores for a date</i>
------------	-------------------------------------

Description

get_scores() is deprecated. Use [scores\(\)](#) instead.

Usage

```
get_scores(date = "now")
```

Arguments

date	character in 'YYYY-MM-DD' (e.g., '2025-01-01'); see seasons() for reference
------	---

Value

data.frame with one row per game

get_seasons	<i>Access all the seasons</i>
-------------	-------------------------------

Description

get_seasons() is deprecated. Use [seasons\(\)](#) instead.

Usage

```
get_seasons()
```

Value

data.frame with one row per season

get_season_now	<i>Access the season and game type as of now</i>
----------------	--

Description

get_season_now() is defunct. Use [season_now\(\)](#) and/or [game_type_now\(\)](#) instead.

Usage

```
get_season_now()
```

get_series	<i>Access the playoff series for a season and round</i>
------------	---

Description

get_series() is defunct.

Usage

```
get_series()
```

get_series_schedule	<i>Access the playoff schedule for a season and series</i>
---------------------	--

Description

get_series_schedule() is deprecated. Use [series_schedule\(\)](#) instead.

Usage

```
get_series_schedule(season = season_now(), series = "a")
```

Arguments

- season integer in YYYYYYYY (e.g., 20242025); see [seasons\(\)](#) for reference
- series one-letter code (e.g., 'O'); see [series\(\)](#) and/or [bracket\(\)](#) for reference

Value

data.frame with one row per game

get_shift_charts	<i>Access the shift charts for a game</i>
------------------	---

Description

get_shift_charts() is deprecated. Use [shift_chart\(\)](#) instead.

Usage

```
get_shift_charts(game = 2023030417)
```

Arguments

game integer ID (e.g., 2025020275); see [games\(\)](#) for reference

Value

data.frame with one row per shift

get_skaters	<i>Access all the skaters for a range of seasons</i>
-------------	--

Description

get_skaters() is defunct. Use [players\(\)](#) instead.

Usage

get_skaters()

get_skater_leaders	<i>Access the skater statistics leaders for a season, game type, and category</i>
--------------------	---

Description

get_skater_leaders() is deprecated. Use [skater_leaders\(\)](#) instead.

Usage

get_skater_leaders(season = "current", game_type = "", category = "points")

Arguments

season integer in YYYYYYYY (e.g., 20242025); see [seasons\(\)](#) for reference

game_type integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season)
OR character of 'pre', 'regular', or 'playoff'/'post'; see [seasons\(\)](#) for reference;
most functions will NOT support pre-season

category character of 'assists', 'goals', 'goalsSh', 'goalsPp', 'points', 'penaltyMins', 'toi',
'plusMinus', or 'faceoffLeaders'

Value

data.frame with one row per player

`get_skater_milestones` *Access the skaters on milestone watch*

Description

`get_skater_milestones()` is deprecated. Use `skater_milestones()` instead.

Usage

```
get_skater_milestones()
```

Value

data.frame with one row per player

`get_skater_statistics` *Access various reports for all the skaters by season or game*

Description

`get_skater_statistics()` is defunct. Use `skater_season_report()` or `skater_game_report()` instead.

Usage

```
get_skater_statistics()
```

`get_spotlight_players` *Access the spotlight players*

Description

`get_spotlight_players()` is deprecated. Use `spotlight_players()` instead.

Usage

```
get_spotlight_players()
```

Value

data.frame with one row per player

get_standings	<i>Access the standings for a date</i>
---------------	--

Description

get_standings() is deprecated. Use [standings\(\)](#) instead.

Usage

```
get_standings(date = "2025-01-01")
```

Arguments

date	character in 'YYYY-MM-DD' (e.g., '2025-01-01'); see seasons() for reference
------	---

Value

data.frame with one row per team

get_standings_information	<i>Access the standings rules by season</i>
---------------------------	---

Description

get_standings_information() is deprecated. Use [standings_rules\(\)](#) instead.

Usage

```
get_standings_information()
```

Value

data.frame with one row per season

get_streams	<i>Access all the streams</i>
-------------	-------------------------------

Description

get_streams() is deprecated. Use [streams\(\)](#) instead.

Usage

```
get_streams()
```

Value

data.frame with one row per stream

get_teams	<i>Access all the teams</i>
-----------	-----------------------------

Description

get_teams() is deprecated. Use [teams\(\)](#) instead.

Usage

```
get_teams()
```

Value

data.frame with one row per team

get_team_prospects	<i>Access the prospects for a team and position</i>
--------------------	---

Description

get_team_prospects() is deprecated. Use [team_prospects\(\)](#) instead.

Usage

```
get_team_prospects(team = "NJD", player_type = "forwards")
```

Arguments

team	three-letter code (e.g., 'COL'); see teams() for reference
player_type	character of 'forwards', 'defensemen', or 'goalies'

Value

data.frame with one row per player

get_team_roster	<i>Access the roster for a team, season, and player type</i>
-----------------	--

Description

get_team_roster() is deprecated. Use [roster\(\)](#) instead.

Usage

```
get_team_roster(team = "NJD", season = "current", player_type = "forwards")
```

Arguments

- team three-letter code (e.g., 'COL'); see [teams\(\)](#) for reference
- season integer in YYYYYYYY (e.g., 20242025); see [seasons\(\)](#) for reference
- player_type character of 'forwards', 'defensemen', or 'goalies'

Value

data.frame with one row per player

get_team_roster_statistics	<i>Access the roster statistics for a team, season, game type, and player type</i>
----------------------------	--

Description

get_team_roster_statistics() is deprecated. Use [roster_statistics\(\)](#) instead.

Usage

```
get_team_roster_statistics(  
  team = "NJD",  
  season = "now",  
  game_type = 2,  
  player_type = "skaters"  
)
```

Arguments

team	three-letter code (e.g., 'COL'); see teams() for reference
season	integer in YYYYYYYY (e.g., 20242025); see seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or playoff'/'post'; see seasons() for reference; most functions will NOT support pre-season
player_type	character of 'skaters' or 'goalies'

Value

data.frame with one row per player

get_team_schedule	<i>Access the schedule for a team and season</i>
-------------------	--

Description

get_team_schedule() is deprecated. Use [team_season_schedule\(\)](#) instead.

Usage

```
get_team_schedule(team = "NJD", season = "now")
```

Arguments

team	three-letter code (e.g., 'COL'); see teams() for reference
season	integer in YYYYYYYY (e.g., 20242025); see seasons() for reference

Value

data.frame with one row per game

get_team_scoreboard	<i>Access the team scoreboard as of now</i>
---------------------	---

Description

get_team_scoreboard() is defunct.

Usage

```
get_team_scoreboard()
```

get_team_seasons	<i>Access the season(s) and game type(s) in which a team played</i>
------------------	---

Description

get_team_seasons() is deprecated. Use [team_seasons\(\)](#) instead.

Usage

```
get_team_seasons(team = "NJD")
```

Arguments

team three-letter code (e.g., 'COL'); see [teams\(\)](#) for reference

Value

data.frame with one row per season

get_team_statistics	<i>Access various reports for all the teams by season or game</i>
---------------------	---

Description

get_team_statistics() is defunct. Use [team_season_report\(\)](#) and/or [team_game_report\(\)](#) instead.

Usage

```
get_team_statistics()
```

get_tv_schedule	<i>Access the NHL Network TV schedule for a date</i>
-----------------	--

Description

get_tv_schedule() is deprecated. Use [tv_schedule\(\)](#) instead.

Usage

```
get_tv_schedule(date = "now")
```

Arguments

date character in 'YYYY-MM-DD' (e.g., '2025-01-01'); see [seasons\(\)](#) for reference

Value

data.frame with one row per program

get_venues	<i>Access all the venues</i>
------------	------------------------------

Description

get_venues() is deprecated. Use [venues\(\)](#) instead.

Usage

get_venues()

Value

data.frame with one row per venue

get_wsc_play_by_play	<i>Access the World Showcase (WSC) play-by-play for a game</i>
----------------------	--

Description

get_wsc_play_by_play() is deprecated. Use [wsc_play_by_play\(\)](#) instead.

Usage

get_wsc_play_by_play(game = 2023030417)

Arguments

game integer ID (e.g., 2025020275); see [games\(\)](#) for reference

Value

data.frame with one row per event (play)

`glossary`*Access the glossary*

Description

`glossary()` returns NHL glossary terms with one row per term and normalized terminology IDs/names.

Usage

```
glossary()
```

Value

data.frame with one row per terminology

Examples

```
glossary <- glossary()
```

`goalie_edge_five_versus_five`*Access the EDGE 5 vs. 5 statistics for a goalie, season, game type, and category*

Description

`goalie_edge_five_versus_five()` returns goalie EDGE 5-on-5 save-percentage detail for one goalie/season/game type. Use `category = 'details'` for the full split list or `category = 'last 10'` for game-level recent form.

Usage

```
goalie_edge_five_versus_five(  
  player = 8476945,  
  season = "now",  
  game_type = "",  
  category = "details"  
)
```

```
goalie_edge_5_vs_5(  
  player = 8476945,  
  season = "now",  
  game_type = "",  
  category = "details"  
)
```

Arguments

player	integer ID (e.g., 8478406)
season	integer in YYYYYYYY (e.g., 20242025); see goalie_edge_seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see goalie_edge_seasons() for reference; most functions will NOT support pre-season
category	character of 'd'/'details' or 'l'/'l10'/'last 10'

Value

list with four items (category = 'details') or data.frame with one row per game (category = 'last 10')

Examples

```
Mackenzie_Blackwood_L10_5_vs_5_regular_20242025 <- goalie_edge_five_versus_five(
  player    = 8478406,
  season    = 20242025,
  game_type = 2,
  category  = 'L'
)
```

goalie_edge_leaders	<i>Access the goalie EDGE statistics leaders for a season and game type</i>
---------------------	---

Description

goalie_edge_leaders() returns the goalie EDGE landing-page leader groups for one season and game type, such as save percentage, 5-on-5 save percentage, and shot-location leader blocks.

Usage

```
goalie_edge_leaders(season = "now", game_type = "")
```

Arguments

season	integer in YYYYYYYY (e.g., 20242025); see goalie_edge_seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see goalie_edge_seasons() for reference; most functions will NOT support pre-season

Value

list of various items

Examples

```
goalie_EDGE_leaders_regular_20242025 <- goalie_edge_leaders(
  season    = 20242025,
  game_type = 2
)
```

goalie_edge_save_percentage

Access the EDGE save percentage statistics for a goalie, season, game type, and category

Description

goalie_edge_save_percentage() returns a goalie's EDGE save-percentage detail list, or recent game rows when category = 'last 10'.

Usage

```
goalie_edge_save_percentage(
  player = 8476945,
  season = "now",
  game_type = "",
  category = "details"
)
```

Arguments

player	integer ID (e.g., 8478406)
season	integer in YYYYYYYY (e.g., 20242025); see goalie_edge_seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see goalie_edge_seasons() for reference; most functions will NOT support pre-season
category	character of 'd'/'details' or 'l'/'10'/'last 10'

Value

list with two items (category = 'details') or data.frame with one row per game (category = 'last 10')

Examples

```
Mackenzie_Blackwood_L10_sP_regular_20242025 <-
  goalie_edge_save_percentage(
    player    = 8478406,
    season    = 20242025,
    game_type = 2,
    category  = 'L'
  )
```

goalie_edge_seasons	<i>Access the season(s) and game type(s) in which there exists goalie EDGE statistics</i>
---------------------	---

Description

goalie_edge_seasons() returns the seasons and game type IDs for which the NHL EDGE goalie endpoints expose data.

Usage

```
goalie_edge_seasons()
```

Value

data.frame with one row per season

Examples

```
goalie_EDGE_seasons <- goalie_edge_seasons()
```

goalie_edge_shot_location	<i>Access the EDGE shot location statistics for a goalie, season, game type, and category</i>
---------------------------	---

Description

goalie_edge_shot_location() returns a goalie's EDGE save results by shot location, either as detailed splits or shot-location totals.

Usage

```
goalie_edge_shot_location(  
  player = 8476945,  
  season = "now",  
  game_type = "",  
  category = "details"  
)
```

Arguments

player	integer ID (e.g., 8478406)
season	integer in YYYYYYYY (e.g., 20242025); see goalie_edge_seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see goalie_edge_seasons() for reference; most functions will NOT support pre-season
category	character of 'd'/details' or 't'/totals'

Value

data.frame with one row per shot location

Examples

```
Mackenzie_Blackwood_shot_location_totals_regular_20242025 <-
goalie_edge_shot_location(
  player    = 8478406,
  season    = 20242025,
  game_type = 2,
  category  = 'T'
)
```

goalie_edge_summary	<i>Access the EDGE summary for a goalie, season, and game type</i>
---------------------	--

Description

goalie_edge_summary() returns the full NHL EDGE detail payload for one goalie, season, and game type, including goalie metadata and the available save-percentage and shot-location summary blocks.

Usage

```
goalie_edge_summary(player = 8476945, season = "now", game_type = "")
```

Arguments

player	integer ID (e.g., 8478406)
season	integer in YYYYYYYY (e.g., 20242025); see goalie_edge_seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see goalie_edge_seasons() for reference; most functions will NOT support pre-season

Value

list of various items

Examples

```
Mackenzie_Blackwood_EDGE_summary_regular_20242025 <- goalie_edge_summary(
  player      = 8478406,
  season      = 20242025,
  game_type   = 2
)
```

goalie_game_report	<i>Access various reports for a season, game type, and category for all the goalies by game</i>
--------------------	---

Description

goalie_game_report() returns the game-level version of a selected goalie stats report, with one row per goalie per game and category-specific metrics.

Usage

```
goalie_game_report(
  season = season_now(),
  game_type = game_type_now(),
  category = "summary"
)
```

Arguments

season	integer in YYYYYYYYY (e.g., 20242025); see seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see seasons() for reference; most functions will NOT support pre-season
category	character (e.g., 'advanced'); see goalie_report_configurations() for reference

Value

data.frame with one row per game per goalie

Examples

```
# May take >5s, so skip.
advanced_goalie_game_report_playoffs_20212022 <-
  goalie_game_report(
    season      = 20212022,
    game_type   = 3,
    category    = 'advanced'
  )
```

goalie_game_scoring *Access the scoring statistics for all the goalies by game*

Description

goalie_game_scoring() returns records-site goalie scoring rows by player and game.

Usage

```
goalie_game_scoring()
```

Value

data.frame with one row per player per game

Examples

```
goalie_game_scoring <- goalie_game_scoring()
```

goalie_game_statistics *Access the statistics for all the goalies by game*

Description

goalie_game_statistics() returns records-site goalie stat rows by player and game, including opponent fields and normalized team abbreviations.

Usage

```
goalie_game_statistics()
```

```
goalie_game_stats()
```

Value

data.frame with one row per goalie per game

Examples

```
goalie_game_stats <- goalie_game_statistics()
```

goalie_leaders	<i>Access the goalie statistics leaders for a season, game type, and category</i>
----------------	---

Description

goalie_leaders() returns the NHL.com leaders table for one goalie category, season, and game type, with normalized player and team fields.

Usage

```
goalie_leaders(season = "current", game_type = "", category = "wins")
```

Arguments

season	integer in YYYYYYYY (e.g., 20242025); see seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see seasons() for reference; most functions will NOT support pre-season
category	character of 'w'/'wins', 's'/'shutouts', 's%'/'sP'/'save %'/'save percentage', or 'gaa'/'goals against average'

Value

data.frame with one row per player

Examples

```
GAA_leaders_regular_20242025 <- goalie_leaders(
  season    = 20242025,
  game_type = 2,
  category  = 'GAA'
)
```

goalie_milestones	<i>Access the goalies on milestone watch</i>
-------------------	--

Description

goalie_milestones() returns NHL.com goalie milestone-watch rows, including goalie/team identifiers and the milestone/countdown fields exposed by the endpoint.

Usage

```
goalie_milestones()
```

Value

data.frame with one row per player

Examples

```
goalie_milestones <- goalie_milestones()
```

goalie_regular_statistics	<i>Access the career regular season statistics for all the goalies</i>
---------------------------	--

Description

goalie_regular_statistics() returns records-site regular-season career goalie totals, with one row per goalie.

Usage

```
goalie_regular_statistics()
```

```
goalie_regular_stats()
```

Value

data.frame with one row per goalie

Examples

```
goalie_career_regular_statistics <- goalie_regular_statistics()
```

`goalie_report_configurations`*Access the configurations for goalie reports*

Description

`goalie_report_configurations()` returns the goalie-report configuration block from the stats API, including valid report categories, fields, filters, and split options accepted by `goalie_season_report()` and `goalie_game_report()`.

Usage

```
goalie_report_configurations()
```

```
goalie_report_configs()
```

Value

list with various items

Examples

```
goalie_report_configs <- goalie_report_configurations()
```

`goalie_scoring`*Access the career scoring statistics for all the goalies*

Description

`goalie_scoring()` returns records-site career scoring totals for goalies, with one row per goalie.

Usage

```
goalie_scoring()
```

Value

data.frame with one row per player

Examples

```
goalie_scoring <- goalie_scoring()
```

goalie_season_report	<i>Access various reports for a season, game type, and category for all the goalies by season</i>
----------------------	---

Description

`goalie_season_report()` returns a season-level goalie report for a selected stats category, with one row per goalie and the metric columns defined by that category.

Usage

```
goalie_season_report(
  season = season_now(),
  game_type = game_type_now(),
  category = "summary"
)
```

Arguments

season	integer in YYYYYYYYY (e.g., 20242025); see seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see seasons() for reference; most functions will NOT support pre-season
category	character (e.g., 'advanced'); see goalie_report_configurations() for reference

Value

data.frame with one row per player

Examples

```
# May take >5s, so skip.
advanced_goalie_season_report_playoffs_20212022 <-
  goalie_season_report(
    season = 20212022,
    game_type = 3,
    category = 'advanced'
  )
```

`goalie_season_statistics`*Access the statistics for all the goalies by season, game type, and team.*

Description

`goalie_season_statistics()` returns records-site goalie stat rows by player, team, season, and game type, preserving separate rows when a goalie changed teams.

Usage

```
goalie_season_statistics()
```

```
goalie_season_stats()
```

Value

data.frame with one row per player per season per game type, separated by team if applicable

Examples

```
goalie_season_stats <- goalie_season_statistics()
```

`goalie_series_statistics`*Access the playoff statistics for all the goalies by series*

Description

`goalie_series_statistics()` returns records-site playoff goalie totals by player and playoff series.

Usage

```
goalie_series_statistics()
```

```
goalie_series_stats()
```

Value

data.frame with one row per player per series

Examples

```
goalie_series_stats <- goalie_series_statistics()
```

goalie_statistics	<i>Access the career statistics for all the goalies</i>
-------------------	---

Description

goalie_statistics() returns records-site regular-season plus playoff career goalie totals, with one row per goalie.

Usage

```
goalie_statistics()
```

```
goalie_stats()
```

Value

data.frame with one row per player

Examples

```
goalie_stats <- goalie_statistics()
```

ig_game_cumulative_expected_goals	<i>Save an Instagram (IG) shareable cumulative expected goals (xG) plot</i>
-----------------------------------	---

Description

ig_game_cumulative_expected_goals() fetches one GameCenter play-by-play, scores shot attempts with [calculate_expected_goals\(\)](#), and renders a home/away cumulative xG time series on a 1080-by-566-style PNG canvas.

Usage

```
ig_game_cumulative_expected_goals(game = 2023030417, model = NULL, save = TRUE)
```

```
ig_game_cum_xG(game = 2023030417, model = NULL)
```

Arguments

game	integer ID (e.g., 2025020275); see games() for reference
model	deprecated legacy model selector; ignored
save	logical; use FALSE to draw on the active graphics device during tests or custom workflows

Value

NULL, invisibly

Examples

```
# May take >5s, so skip.
ig_game_cumulative_expected_goals(
  game = 2023030417,
  save = FALSE
)
```

ig_game_shot_locations

Save an Instagram (IG) shareable shot-location plot for a game

Description

ig_game_shot_locations() fetches one GameCenter play-by-play, scores its shot attempts with [calculate_expected_goals\(\)](#), filters to the selected home or away team, and renders a 1080-by-566-style PNG shot map. Points are jittered for readability, shaped by event result, and colored by capped xG.

Usage

```
ig_game_shot_locations(
  game = 2023030417,
  team = "home",
  model = NULL,
  save = TRUE
)
```

```
ig_game_shot_locs(game = 2023030417, team = "home", model = NULL)
```

Arguments

game	integer ID (e.g., 2025020275); see games() for reference
team	character of 'h'/'home' or 'a'/'away'
model	deprecated legacy model selector; ignored
save	logical; use FALSE to draw on the active graphics device during tests or custom workflows

Value

NULL, invisibly

Examples

```
# May take >5s, so skip.
ig_game_shot_locations(
  game = 2023030417,
  team = 'H',
  save = FALSE
)
```

location	<i>Access the location for a zip code</i>
----------	---

Description

location() returns the NHL postal lookup result for one ZIP/postal code, including country, region, and related local-market fields when available.

Usage

```
location(zip = 10001)
```

Arguments

zip integer (e.g., 48304)

Value

data.frame with one row per team

Examples

```
Cranbrook_Schools <- location(48304)
```

lottery_odds	<i>Access the draft lottery odds</i>
--------------	--------------------------------------

Description

lottery_odds() returns historical draft-lottery odds and outcomes by draft year and lottery slot.

Usage

```
lottery_odds()
```

Value

data.frame with one row per draft lottery

Examples

```
lottery_odds <- lottery_odds()
```

nhlscraper

nhlscraper: Scrape, clean, and visualize NHL data

Description

nhlscraper is an R package to scrape, clean, and visualize NHL data via the NHL and ESPN APIs. It primarily wraps 125+ endpoints from high-level multi-season summaries and award winners to low-level decisecond replays and bookmakers' odds, making them significantly more accessible. It also features cleaning and visualization functions, primarily for play-by-plays, to help analyze the data.

Prerequisite

- R/RStudio; you can check out my [tutorial](#) if you are not familiar.

Disclosure

Detailed documentation for each scraping function will be released gradually over time. Because each function can return a large amount of information, users are encouraged to explore the provided examples to discover what is available. Most, if not all, of the endpoints accessed by this package are unofficially documented (i.e., hidden), so it is important to use them responsibly and with respect for the NHL's data servers. Endpoints serving historical or otherwise mostly static data should ideally be queried once and then stored locally (for example, in a MySQL database) for further analysis, rather than being called repeatedly; the [load functions](#) attempt to mitigate this risk by providing pre-scraped data, so please make use of them. The exact rate limits for these APIs are not publicly known, so users are asked to avoid excessive or abusive querying to help ensure continued access for everyone.

History

Prior to the NHL API rework in 2023, Drew Hynes documented a comprehensive list of known endpoints, and several R packages, such as [nhlapi](#) and [hockeyR](#), were built to access them. However, after the NHL completely transformed its API structure, all of these packages became mostly defunct as their authors understandably chose not to continue maintaining them. The community gathered around work by Zachary Maludzinski to discover and share new endpoints, but progress naturally slowed once most of the "main" endpoints were identified. Over the summer of 2025, I began reverse-engineering many of the remaining, undocumented endpoints while searching for additional data for future research, with a particular focus on NHL EDGE and Records data. After I shared those findings, the effort to expand and refine this map of the APIs accelerated, and with the addition of David Fleischer's effort, we ultimately identified 400+ new endpoints. In parallel, I also discovered many new endpoints for the ESPN API, extending beyond what Joseph Wilson had already compiled.

Author(s)

Maintainer: Rento Saijo <rentosaijo0527@gmail.com> ([ORCID](#)) [copyright holder]

Other contributors:

- Lars Skytte <lars.sunesen.skytte@gmail.com> [contributor]
- Jack Pallotta <jackjpallotta@gmail.com> ([ORCID](#)) [contributor]

See Also

Useful links:

- <https://rentosaijo.github.io/nhlscraper/>
- <https://github.com/RentoSaijo/nhlscraper>
- Report bugs at <https://github.com/RentoSaijo/nhlscraper/issues>

officials

Access all the officials

Description

`officials()` returns the records-site official registry with one row per official and normalized official ID, name, and referee-association fields.

Usage

```
officials()
```

Value

data.frame with one row per official

Examples

```
all_officials <- officials()
```

penalty_shots	<i>Access all the penalty shots</i>
---------------	-------------------------------------

Description

penalty_shots() returns historical penalty-shot records with one row per attempt, including game, season/game type, shooter/goalie, team, and outcome fields.

Usage

```
penalty_shots()

pss()
```

Value

data.frame with one row per penalty shot

Examples

```
all_pss <- penalty_shots()
```

ping	<i>Ping</i>
------	-------------

Description

ping() is defunct.

Usage

```
ping()
```

players	<i>Access all the players</i>
---------	-------------------------------

Description

players() returns the records-site player registry with one row per player and normalized IDs, names, position, handedness, and birth fields.

Usage

```
players()
```

Value

data.frame with one row per player

Examples

```
# May take >5s, so skip.
all_players <- players()
```

player_game_log	<i>Access the game log for a player, season, and game type</i>
-----------------	--

Description

player_game_log() returns one player’s GameCenter game log for a season and game type, with one row per game and normalized opponent, score/result, and player stat fields.

Usage

```
player_game_log(player = 8478402, season = "now", game_type = "")
```

Arguments

- | | |
|-----------|---|
| player | integer ID (e.g., 8480039); see players() for reference |
| season | integer in YYYYYYYY (e.g., 20242025); see seasons() for reference |
| game_type | integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season)
OR character of 'pre', 'regular', or 'playoff'/'post'; see seasons() for reference;
most functions will NOT support pre-season |

Value

data.frame with one row per game

Examples

```
Martin_Necas_game_log_regular_20242025 <- player_game_log(
  player    = 8480039,
  season    = 20242025,
  game_type = 2
)
```

player_seasons	Access the season(s) and game type(s) in which a player played
----------------	--

Description

player_seasons() returns the seasons and game type IDs available in a player’s GameCenter game-log payload.

Usage

```
player_seasons(player = 8478402)
```

Arguments

player integer ID (e.g., 8480039); see [players\(\)](#) for reference

Value

data.frame with one row per season

Examples

```
Martin_Necas_seasons <- player_seasons(player = 8480039)
```

player_summary	Access the summary for a player
----------------	---------------------------------

Description

player_summary() returns the public player landing payload for one player, including bio/team fields, current season stats, career totals, season totals, awards, and featured media when available.

Usage

```
player_summary(player = 8478402)
```

Arguments

player integer ID (e.g., 8480039); see [players\(\)](#) for reference

Value

list with various items

Examples

```
Martin_Necas_summary <- player_summary(player = 8480039)
```

```
playoff_season_statistics
```

Access the playoff statistics by season

Description

`playoff_season_statistics()` returns league-wide playoff totals by season, including games, goals, and related year-level totals.

Usage

```
playoff_season_statistics()
```

```
playoff_season_stats()
```

Value

data.frame with one row per season

Examples

```
playoff_season_stats <- playoff_season_statistics()
```

```
ps
```

Access all the penalty shots

Description

`ps()` is deprecated. Use [penalty_shots\(\)](#) instead.

Usage

```
ps()
```

Value

data.frame with one row per penalty shot

replay	<i>Access the replay for an event</i>
--------	---------------------------------------

Description

replay() downloads NHL puck/player tracking sprite data for one goal event and returns one row per decisecond with puck coordinates plus dynamic playerN* columns for tracked skaters.

Usage

```
replay(game = 2023030417, event = 866)
```

Arguments

- game integer ID (e.g., 2025020262); see [games\(\)](#) for reference
- event integer ID (e.g., 751); see [gc_play_by_play\(\)](#) and/or [wsc_play_by_play\(\)](#) for reference; must be a 'goal' event

Value

data.frame with one row per decisecond

Examples

```
Gabriel_Landeskog_first_regular_goal_back_replay <- replay(
  game = 2025020262,
  event = 751
)
```

replays	<i>Access the replays for a season</i>
---------	--

Description

replays() downloads the stored season-level puck/player tracking parquet snapshot and returns one row per tracked decisecond.

Usage

```
replays(season = 20242025)
```

Arguments

- season integer in YYYYYYYY (e.g., 20242025); see [seasons\(\)](#) for reference

Value

data.frame with one row per decisecond

Examples

```
# May take >5s, so skip.
replays_20252026 <- replays(season = 20252026)
```

roster	<i>Access the roster for a team, season, and position</i>
--------	---

Description

roster() returns a team’s roster for one season and position group, with one row per player and normalized ID, name, sweater, position, height/weight, birth, and handedness fields when available.

Usage

```
roster(team = 1, season = "current", position = "forwards")
```

Arguments

team	integer ID (e.g., 21), character full name (e.g., 'Colorado Avalanche'), OR three-letter code (e.g., 'COL'); see teams() for reference; ID is preferable as there now exists duplicate three-letter codes (i.e., 'UTA' for 'Utah Hockey Club' and 'Utah Mammoth')
season	integer in YYYYYYYY (e.g., 20242025); see seasons() for reference
position	character of 'f'/'forwards', 'd'/'defensemen', or 'g'/'goalies'

Value

data.frame with one row per player

Examples

```
COL_defensemen_20242025 <- roster(
  team    = 21,
  season  = 20242025,
  position = 'D'
)
```

roster_statistics	<i>Access the roster statistics for a team, season, game type, and position</i>
-------------------	---

Description

roster_statistics() returns skater or goalie stat rows for one team, season, and game type, preserving the NHL API’s position-specific stat columns with normalized player name fields.

Usage

```
roster_statistics(  
  team = 1,  
  season = "now",  
  game_type = "",  
  position = "skaters"  
)  
  
roster_stats(team = 1, season = "now", game_type = "", position = "skaters")
```

Arguments

team	integer ID (e.g., 21), character full name (e.g., 'Colorado Avalanche'), OR three-letter code (e.g., 'COL'); see teams() for reference; ID is preferable as there now exists duplicate three-letter codes (i.e., 'UTA' for 'Utah Hockey Club' and 'Utah Mammoth')
season	integer in YYYYYYYY (e.g., 20242025); see seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see seasons() for reference; most functions will NOT support pre-season
position	character of 's'/'skaters' or 'g'/'goalies'

Value

data.frame with one row per player

Examples

```
COL_goalies_statistics_regular_20242025 <- roster_statistics(  
  team      = 21,  
  season    = 20242025,  
  game_type = 2,  
  position  = 'G'  
)
```

schedule	<i>Access the schedule for a date</i>
----------	---------------------------------------

Description

schedule() returns the public schedule rows for one date, including game IDs, teams, start/status, scores, venue, broadcasts, and links.

Usage

```
schedule(date = Sys.Date())
```

Arguments

date	character in 'YYYY-MM-DD' (e.g., '2025-01-01'); see seasons() for reference
------	---

Value

data.frame with one row per game

Examples

```
schedule_Halloween_2025 <- schedule(date = '2025-10-31')
```

scores	<i>Access the scores for a date</i>
--------	-------------------------------------

Description

scores() returns the public scoreboard for one date with one row per game, including team abbreviations, start/status fields, score state, venue, broadcasts, and game links when the API exposes them.

Usage

```
scores(date = "now")
```

Arguments

date	character in 'YYYY-MM-DD' (e.g., '2025-01-01'); see seasons() for reference
------	---

Value

data.frame with one row per game

Examples

```
scores_Halloween_2025 <- scores(date = '2025-10-31')
```

seasons	<i>Access all the seasons</i>
---------	-------------------------------

Description

seasons() returns the stats API season catalog with one row per season, including season IDs, date boundaries, and game-type metadata.

Usage

```
seasons()
```

Value

data.frame with one row per season

Examples

```
all_seasons <- seasons()
```

season_now	<i>Access the season as of now</i>
------------	------------------------------------

Description

season_now() retrieves the season as of now and returns a scalar integer used as the current-context default in season/game-type dependent wrappers.

Usage

```
season_now()
```

Value

integer in YYYYYYYY (e.g., 20242025)

Examples

```
season_now <- season_now()
```

series	<i>Access all the playoff series by game</i>
--------	--

Description

series() returns the records-site playoff-series game table with one row per game/series entry, including season, series letter, round, teams, and game ID fields.

Usage

```
series()
```

Value

data.frame with one row per game per series

Examples

```
# May take >5s, so skip.
all_series <- series()
```

series_schedule	<i>Access the playoff schedule for a season and series</i>
-----------------	--

Description

series_schedule() returns the public schedule for one playoff series, with one row per game and normalized game, team, score/status, venue, and broadcast fields.

Usage

```
series_schedule(season = season_now() - 10001, series = "a")
```

Arguments

- | | |
|--------|--|
| season | integer in YYYYYYYY (e.g., 20242025); see seasons() for reference |
| series | one-letter code (e.g., 'O'); see series() and/or bracket() for reference |

Value

data.frame with one row per game

Examples

```
SCF_schedule_20212022 <- series_schedule(
  season = 20212022,
  series = 'O'
)
```

shifts	<i>Access the shift charts for a game</i>
--------	---

Description

shifts() is deprecated. Use [shift_chart\(\)](#) instead.

Usage

```
shifts(game = 2023030417)
```

Arguments

game integer ID (e.g., 2025020275); see [games\(\)](#) for reference

Value

data.frame with one row per shift

shift_chart	<i>Access the shift chart for a game</i>
-------------	--

Description

shift_chart() returns one row per player shift for a game with gameId, teamId, playerId, shift number, period, start/end clocks, elapsed period/game seconds, and duration. It uses the stats API when available and falls back to the NHL HTML shift reports for older or missing games.

Usage

```
shift_chart(game = 2023030417)
```

Arguments

game integer ID (e.g., 2025020275); see [games\(\)](#) for reference

Value

data.frame with one row per shift

Examples

```
shifts_Martin_Necas_legacy_game <- shift_chart(game = 2025020275)
```

shift_charts	<i>Access the shift charts for a season</i>
--------------	---

Description

shift_charts() downloads the stored season-level shift-chart parquet snapshot, removes the storage-only id column, and returns one row per parsed shift.

Usage

```
shift_charts(season = 20242025)
```

Arguments

season integer in YYYYYYYYY (e.g., 20242025); see [seasons\(\)](#) for reference

Value

data.frame with one row per event (play) per game

Examples

```
# May take >5s, so skip.  
shift_charts_20212022 <- shift_charts(season = 20212022)
```

shift_chart_summaries	<i>Access the shift chart summaries for a season</i>
-----------------------	--

Description

shift_chart_summaries() downloads the stored season-level shift-summary parquet snapshot with per-player, per-period time-on-ice splits.

Usage

```
shift_chart_summaries(season = 20242025)
```

Arguments

season integer in YYYYYYYYY (e.g., 20242025); see [seasons\(\)](#) for reference

Value

data.frame with one row per player per period

Examples

```
# May take >5s, so skip.  
shift_chart_summaries_20212022 <- shift_chart_summaries(season = 20212022)
```

`shift_chart_summary` *Access the shift chart time-on-ice summary for a game*

Description

`shift_chart_summary()` retrieves per-player, per-period time-on-ice splits from the HTML shift reports.

Usage

```
shift_chart_summary(game = 2023030417)
```

Arguments

`game` integer ID (e.g., 2025020275); see [games\(\)](#) for reference

Value

data.frame with one row per player per period

Examples

```
shift_summary_2023030417 <- shift_chart_summary(game = 2023030417)
```

`skater_edge_leaders` *Access the skater EDGE statistics leaders for a season and game type*

Description

`skater_edge_leaders()` returns the skater EDGE landing-page leader groups for one season and game type, such as speed, distance, shot speed, shot location, and zone-time leader blocks.

Usage

```
skater_edge_leaders(season = "now", game_type = "")
```

Arguments

`season` integer in YYYYYYYYY (e.g., 20242025); see [skater_edge_seasons\(\)](#) for reference

`game_type` integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see [skater_edge_seasons\(\)](#) for reference; most functions will NOT support pre-season

Value

list of various items

Examples

```
skater_EDGE_leaders_regular_20242025 <- skater_edge_leaders(
  season    = 20242025,
  game_type = 2
)
```

skater_edge_seasons	<i>Access the season(s) and game type(s) in which there exists skater EDGE statistics</i>
---------------------	---

Description

skater_edge_seasons() returns the seasons and game type IDs for which the NHL EDGE skater endpoints expose data.

Usage

```
skater_edge_seasons()
```

Value

data.frame with one row per season

Examples

```
skater_EDGE_seasons <- skater_edge_seasons()
```

skater_edge_shot_location	<i>Access the EDGE shot location statistics for a skater, season, game type, and category</i>
---------------------------	---

Description

skater_edge_shot_location() returns a skater's EDGE shot-location detail by rink region, or totals by shot-location bucket when category = 'totals'.

Usage

```
skater_edge_shot_location(
  player = 8478402,
  season = "now",
  game_type = "",
  category = "details"
)
```

Arguments

player	integer ID (e.g., 8480039); see players() for reference
season	integer in YYYYYYYY (e.g., 20242025); see skater_edge_seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see skater_edge_seasons() for reference; most functions will NOT support pre-season
category	character of 'd'/details' or 't'/'totals'

Value

data.frame with one row per shot location

Examples

```
Martin_Necas_shot_location_totals_regular_20242025 <-
  skater_edge_shot_location(
    player = 8480039,
    season = 20242025,
    game_type = 2,
    category = 'T'
  )
```

skater_edge_shot_speed

Access the EDGE shot speed statistics for a skater, season, game type, and category

Description

skater_edge_shot_speed() returns a skater's EDGE shot-speed detail list, or hardest-shot rows when category = 'hardest'.

Usage

```
skater_edge_shot_speed(
  player = 8478402,
  season = "now",
  game_type = "",
  category = "details"
)
```

Arguments

player	integer ID (e.g., 8480039); see players() for reference
season	integer in YYYYYYYY (e.g., 20242025); see skater_edge_seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see skater_edge_seasons() for reference; most functions will NOT support pre-season
category	character of 'd'/'details' or 'h'/'hardest'

Value

list with six items (category = 'details') or data.frame with one row per shot (category = 'hardest')

Examples

```
Martin_Necas_hardest_shots_regular_20242025 <- skater_edge_shot_speed(
  player = 8480039,
  season = 20242025,
  game_type = 2,
  category = 'H'
)
```

skater_edge_skating_distance

Access the EDGE skating distance statistics for a skater, season, game type, and category

Description

skater_edge_skating_distance() returns a skater's EDGE distance detail by strength state, or recent game rows when category = 'last 10'.

Usage

```
skater_edge_skating_distance(
  player = 8478402,
  season = "now",
  game_type = "",
  category = "details"
)
```

Arguments

player	integer ID (e.g., 8480039); see players() for reference
season	integer in YYYYYYYY (e.g., 20242025); see skater_edge_seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see skater_edge_seasons() for reference; most functions will NOT support pre-season
category	character of 'd'/'details' or 'l'/'l10'/'last 10'

Value

data.frame with one row per strength state (category = 'details') or game (category = 'last 10')

Examples

```
Martin_Necas_L10_skating_distance_regular_20242025 <-
  skater_edge_skating_distance(
    player = 8480039,
    season = 20242025,
    game_type = 2,
    category = 'L'
  )
```

skater_edge_skating_speed

Access the EDGE skating speed statistics for a skater, season, game type, and category

Description

skater_edge_skating_speed() returns a skater's EDGE skating-speed detail list, or top-speed burst rows when category = 'top speeds'.

Usage

```
skater_edge_skating_speed(
  player = 8478402,
  season = "now",
  game_type = "",
  category = "details"
)
```

Arguments

player	integer ID (e.g., 8480039); see players() for reference
season	integer in YYYYYYYYY (e.g., 20242025); see skater_edge_seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see skater_edge_seasons() for reference; most functions will NOT support pre-season
category	character of 'd'/'details' or 't'/'top'/'top speeds'

Value

list with four items (category = 'details') or data.frame with one row per burst (category = 'top speeds')

Examples

```
Martin_Necas_top_speeds_regular_20242025 <- skater_edge_skating_speed(
  player    = 8480039,
  season    = 20242025,
  game_type = 2,
  category  = 'T'
)
```

skater_edge_summary	<i>Access the EDGE summary for a skater, season, and game type</i>
---------------------	--

Description

skater_edge_summary() returns the full NHL EDGE detail payload for one skater, season, and game type, including player metadata and the available skating, shot, and zone-time summary blocks.

Usage

```
skater_edge_summary(player = 8478402, season = "now", game_type = "")
```

Arguments

player	integer ID (e.g., 8480039); see players() for reference
season	integer in YYYYYYYYY (e.g., 20242025); see skater_edge_seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see skater_edge_seasons() for reference; most functions will NOT support pre-season

Value

list of various items

Examples

```
Martin_Necas_EDGE_summary_regular_20242025 <- skater_edge_summary(
  player    = 8480039,
  season    = 20242025,
  game_type = 2
)
```

skater_edge_zone_time	<i>Access the EDGE zone time statistics for a skater, season, game type, and category</i>
-----------------------	---

Description

skater_edge_zone_time() returns a skater’s EDGE zone-time detail table by strength state, or the zone-start split list when category = 'starts'.

Usage

```
skater_edge_zone_time(
  player = 8478402,
  season = "now",
  game_type = "",
  category = "details"
)
```

Arguments

player	integer ID (e.g., 8480039); see players() for reference
season	integer in YYYYYYYY (e.g., 20242025); see skater_edge_seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see skater_edge_seasons() for reference; most functions will NOT support pre-season
category	character of 'd'/'details' or 's'/'starts'

Value

data.frame with one row per strength state (category = 'details') or list with six items (category = 'starts')

Examples

```
Martin_Necas_starts_regular_20242025 <- skater_edge_zone_time(
  player   = 8480039,
  season   = 20242025,
  game_type = 2,
  category  = 'S'
)
```

skater_game_report	<i>Access various reports for a season, game type, and category for all the skaters by game</i>
--------------------	---

Description

skater_game_report() returns the game-level version of a selected skater stats report, with one row per player per game and category-specific metrics. Regular-season calls are paged by month to avoid oversized API requests.

Usage

```
skater_game_report(
  season = season_now(),
  game_type = game_type_now(),
  category = "summary"
)
```

Arguments

season	integer in YYYYYYYYY (e.g., 20242025); see seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see seasons() for reference; most functions will NOT support pre-season
category	character (e.g., 'puckPossessions'); see skater_report_configurations() for reference

Value

data.frame with one row per game per player

Examples

```
# May take >5s, so skip.
possession_skater_game_report_playoff_20212022 <-
  skater_game_report(
    season   = 20212022,
    game_type = 3,
    category  = 'puckPossessions'
  )
```

skater_leaders	<i>Access the skater statistics leaders for a season, game type, and category</i>
----------------	---

Description

skater_leaders() returns the NHL.com leaders table for one skater category, season, and game type, with normalized player and team fields.

Usage

```
skater_leaders(season = "current", game_type = "", category = "points")
```

Arguments

season	integer in YYYYYYYYY (e.g., 20242025); see seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or playoff'/'post'; see seasons() for reference; most functions will NOT support pre-season
category	string of 'a'/'assists', 'g'/'goals', 'shg'/'shorthanded goals', 'ppg'/'powerplay goals', 'p'/'points', 'pim'/'penalty minutes'/'penalty infraction minutes', 'toi'/'time on ice', 'pm'/'plus minus', or 'f'/'faceoffs'

Value

data.frame with one row per player

Examples

```
TOI_leaders_regular_20242025 <- skater_leaders(  
  season    = 20242025,  
  game_type = 2,  
  category  = 'TOI'  
)
```

skater_milestones	<i>Access the skaters on milestone watch</i>
-------------------	--

Description

skater_milestones() returns NHL.com skater milestone-watch rows, including player/team identifiers and the milestone/countdown fields exposed by the endpoint.

Usage

```
skater_milestones()
```

Value

data.frame with one row per player

Examples

```
skater_milestones <- skater_milestones()
```

```
skater_playoff_statistics
```

Access the career playoff statistics for all the skaters

Description

skater_playoff_statistics() returns records-site playoff career scoring totals for skaters, with one row per player.

Usage

```
skater_playoff_statistics()
```

```
skater_playoff_stats()
```

Value

data.frame with one row per player

Examples

```
skater_playoff_stats <- skater_playoff_statistics()
```

```
skater_regular_statistics
```

Access the career regular season statistics for all the skaters

Description

skater_regular_statistics() returns records-site regular-season career scoring totals for skaters, with one row per player.

Usage

```
skater_regular_statistics()
```

```
skater_regular_stats()
```

Value

data.frame with one row per player

Examples

```
skater_regular_stats <- skater_regular_statistics()
```

```
skater_report_configurations
```

Access the configurations for skater reports

Description

`skater_report_configurations()` returns the skater-report configuration block from the stats API, including valid report categories, fields, filters, and split options accepted by `skater_season_report()` and `skater_game_report()`.

Usage

```
skater_report_configurations()
```

```
skater_report_configs()
```

Value

list with various items

Examples

```
skater_report_configs <- skater_report_configurations()
```

```
skater_season_report
```

Access various reports for a season, game type, and category for all the skaters by season

Description

`skater_season_report()` returns a season-level skater report for a selected stats category, with one row per player and the metric columns defined by that category.

Usage

```
skater_season_report(  
  season = season_now(),  
  game_type = game_type_now(),  
  category = "summary"  
)
```

Arguments

season	integer in YYYYYYYY (e.g., 20242025); see seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or playoff'/'post'; see seasons() for reference; most functions will NOT support pre-season
category	character (e.g., 'puckPossessions'); see skater_report_configurations() for reference

Value

data.frame with one row per player

Examples

```
# May take >5s, so skip.
possession_skater_season_report_playoff_20212022 <-
  skater_season_report(
    season    = 20212022,
    game_type = 3,
    category  = 'puckPossessions'
  )
```

skater_season_statistics

Access the statistics for all the skaters by season, game type, and team

Description

skater_season_statistics() returns records-site player stat rows by player, team, season, and game type, preserving separate rows when a player changed teams.

Usage

```
skater_season_statistics()
```

```
skater_season_stats()
```

Value

data.frame with one row per player per season per game type, separated by team if applicable

Examples

```
# May take >5s, so skip.
skater_season_stats <- skater_season_statistics()
```

`skater_series_statistics`*Access the playoff statistics for all the skaters by series*

Description

`skater_series_statistics()` returns records-site playoff skater totals by player and playoff series.

Usage

```
skater_series_statistics()
```

```
skater_series_stats()
```

Value

data.frame with one row per player per series

Examples

```
# May take >5s, so skip.  
skater_series_stats <- skater_series_statistics()
```

`skater_statistics`*Access the career statistics for all the skaters*

Description

`skater_statistics()` returns records-site regular-season plus playoff career scoring totals for skaters, with one row per player.

Usage

```
skater_statistics()
```

```
skater_stats()
```

Value

data.frame with one row per player

Examples

```
skater_stats <- skater_statistics()
```

spotlight_players	<i>Access the spotlight players</i>
-------------------	-------------------------------------

Description

spotlight_players() returns NHL.com player-spotlight entries with normalized player, position, and team fields.

Usage

```
spotlight_players()
```

Value

data.frame with one row per player

Examples

```
spotlight_players <- spotlight_players()
```

standings	<i>Access the standings for a date</i>
-----------	--

Description

standings() returns public standings for one date with one row per team, including rank, record, points, streak, division/conference, and normalized team fields.

Usage

```
standings(date = "now")
```

Arguments

date	character in 'YYYY-MM-DD' (e.g., '2025-01-01'); see seasons() for reference
------	---

Value

data.frame with one row per team

Examples

```
standings_Halloween_2025 <- standings(date = '2025-10-31')
```

standings_rules	<i>Access the standings rules by season</i>
-----------------	---

Description

`standings_rules()` returns the public standings-season catalog with one row per season and the rules/metadata used by standings endpoints.

Usage

```
standings_rules()
```

Value

data.frame with one row per season

Examples

```
standings_rules <- standings_rules()
```

streams	<i>Access all the streams</i>
---------	-------------------------------

Description

`streams()` returns the public "where to watch" payload for the current region, including broadcast/streaming providers when the endpoint is available.

Usage

```
streams()
```

Value

data.frame with one row per stream

Examples

```
all_streams <- streams()
```

teams	<i>Access all the teams</i>
-------	-----------------------------

Description

teams() returns the stats API team catalog with one row per franchise-era team entry and normalized identifiers, full names, and tri-codes.

Usage

```
teams()
```

Value

data.frame with one row per team

Examples

```
all_teams <- teams()
```

team_edge_leaders	<i>Access the team EDGE statistics leaders for a season and game type</i>
-------------------	---

Description

team_edge_leaders() returns the team EDGE landing-page leader groups for one season and game type, such as skating, shot, and zone-time leader blocks.

Usage

```
team_edge_leaders(season = "now", game_type = "")
```

Arguments

- | | |
|-----------|---|
| season | integer in YYYYYYYY (e.g., 20242025); see team_edge_seasons() for reference |
| game_type | integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see team_edge_seasons() for reference; most functions will NOT support pre-season |

Value

list of various items

Examples

```
team_EDGE_leaders_regular_20242025 <- team_edge_leaders(
  season    = 20242025,
  game_type = 2
)
```

team_edge_seasons	<i>Access the season(s) and game type(s) in which there exists team EDGE statistics</i>
-------------------	---

Description

team_edge_seasons() returns the seasons and game type IDs for which the NHL EDGE team endpoints expose data.

Usage

```
team_edge_seasons()
```

Value

data.frame with one row per season

Examples

```
team_EDGE_seasons <- team_edge_seasons()
```

team_edge_shot_location	<i>Access the EDGE shot location statistics for a team, season, game type, and category</i>
-------------------------	---

Description

team_edge_shot_location() returns team EDGE shot-location detail by rink region, or totals by strength state and position when category = 'totals'.

Usage

```
team_edge_shot_location(
  team = 1,
  season = "now",
  game_type = "",
  category = "details"
)
```

Arguments

team	integer ID (e.g., 21), character full name (e.g., 'Colorado Avalanche'), OR three-letter code (e.g., 'COL'); see teams() for reference; ID is preferable as there now exists duplicate three-letter codes (i.e., 'UTA' for 'Utah Hockey Club' and 'Utah Mammoth')
season	integer in YYYYYYYY (e.g., 20242025); see team_edge_seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see team_edge_seasons() for reference; most functions will NOT support pre-season
category	character of 'd'/'details' or 't'/'totals'

Value

data.frame with one row per location (category = 'details') or combination of strength state and position (category = 'totals')

Examples

```
COL_shot_location_totals_regular_20242025 <- team_edge_shot_location(
  team      = 21,
  season    = 20242025,
  game_type = 2,
  category  = 'T'
)
```

team_edge_shot_speed	<i>Access the EDGE shot speed statistics for a team, season, game type, and category</i>
----------------------	--

Description

team_edge_shot_speed() returns team EDGE shot-speed detail by position, or hardest-shot rows when category = 'hardest'.

Usage

```
team_edge_shot_speed(
  team = 1,
  season = "now",
  game_type = "",
  category = "details"
)
```

Arguments

team	integer ID (e.g., 21), character full name (e.g., 'Colorado Avalanche'), OR three-letter code (e.g., 'COL'); see teams() for reference; ID is preferable as there now exists duplicate three-letter codes (i.e., 'UTA' for 'Utah Hockey Club' and 'Utah Mammoth')
season	integer in YYYYYYYY (e.g., 20242025); see team_edge_seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see team_edge_seasons() for reference; most functions will NOT support pre-season
category	character of 'd'/'details' or 'h'/'hardest'

Value

data.frame with one row per position (category = 'details') or shot (category = 'hardest')

Examples

```
COL_hardest_shots_regular_20242025 <- team_edge_shot_speed(
  team      = 21,
  season    = 20242025,
  game_type = 2,
  category  = 'H'
)
```

team_edge_skating_distance

Access the EDGE skating distance statistics for a team, season, game type, and category

Description

team_edge_skating_distance() returns team EDGE skating-distance detail by strength state and position, or recent game rows when category = 'last 10'.

Usage

```
team_edge_skating_distance(
  team = 1,
  season = "now",
  game_type = "",
  category = "details"
)
```

Arguments

team	integer ID (e.g., 21), character full name (e.g., 'Colorado Avalanche'), OR three-letter code (e.g., 'COL'); see teams() for reference; ID is preferable as there now exists duplicate three-letter codes (i.e., 'UTA' for 'Utah Hockey Club' and 'Utah Mammoth')
season	integer in YYYYYYYY (e.g., 20242025); see team_edge_seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see team_edge_seasons() for reference; most functions will NOT support pre-season
category	character of 'd'/'details' or 'l'/'10'/'last 10'

Value

data.frame with one row per combination of strength state and position (category = 'details') or game (category = 'last 10') game

Examples

```
COL_L10_skating_distance_regular_20242025 <- team_edge_skating_distance(  
  team      = 21,  
  season    = 20242025,  
  game_type = 2,  
  category  = 'L'  
)
```

team_edge_skating_speed	<i>Access the EDGE skating speed statistics for a team, season, game type, and category</i>
-------------------------	---

Description

team_edge_skating_speed() returns team EDGE skating-speed detail by position, or top-speed burst rows when category = 'top speeds'.

Usage

```
team_edge_skating_speed(  
  team = 1,  
  season = "now",  
  game_type = "",  
  category = "details"  
)
```

Arguments

team	integer ID (e.g., 21), character full name (e.g., 'Colorado Avalanche'), OR three-letter code (e.g., 'COL'); see teams() for reference; ID is preferable as there now exists duplicate three-letter codes (i.e., 'UTA' for 'Utah Hockey Club' and 'Utah Mammoth')
season	integer in YYYYYYYY (e.g., 20242025); see team_edge_seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see team_edge_seasons() for reference; most functions will NOT support pre-season
category	character of 'd'/'details' or 't'/'top'/'top speeds'

Value

data.frame with one row per position (category = 'details') or burst (category = 'top speeds')

Examples

```
COL_top_speeds_regular_20242025 <- team_edge_skating_speed(
  team      = 21,
  season    = 20242025,
  game_type = 2,
  category  = 'T'
)
```

team_edge_summary	<i>Access the EDGE summary for a team, season, and game type</i>
-------------------	--

Description

team_edge_summary() returns the full NHL EDGE detail payload for one team, season, and game type, including team metadata and the available skating, shot, and zone-time summary blocks.

Usage

```
team_edge_summary(team = 1, season = "now", game_type = "")
```

Arguments

team	integer ID (e.g., 21), character full name (e.g., 'Colorado Avalanche'), OR three-letter code (e.g., 'COL'); see teams() for reference; ID is preferable as there now exists duplicate three-letter codes (i.e., 'UTA' for 'Utah Hockey Club' and 'Utah Mammoth')
season	integer in YYYYYYYY (e.g., 20242025); see team_edge_seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see team_edge_seasons() for reference; most functions will NOT support pre-season

Value

list of various items

Examples

```
COL_EDGE_summary_regular_20242025 <- team_edge_summary(  
  team      = 21,  
  season    = 20242025,  
  game_type = 2  
)
```

team_edge_zone_time	<i>Access the EDGE zone time statistics for a team, season, game type, and category</i>
---------------------	---

Description

team_edge_zone_time() returns a team’s EDGE zone-time detail table by strength state, or the shot-differential split list when requested.

Usage

```
team_edge_zone_time(  
  team = 1,  
  season = "now",  
  game_type = "",  
  category = "details"  
)
```

Arguments

team	integer ID (e.g., 21), character full name (e.g., 'Colorado Avalanche'), OR three-letter code (e.g., 'COL'); see teams() for reference; ID is preferable as there now exists duplicate three-letter codes (i.e., 'UTA' for 'Utah Hockey Club' and 'Utah Mammoth')
season	integer in YYYYYYYY (e.g., 20242025); see team_edge_seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see team_edge_seasons() for reference; most functions will NOT support pre-season
category	character of 'd'/'details' or 'dS'/'dSOG'/'dShot'/'shot differential'

Value

data.frame with one row per strength state (category = 'details') or list with four items (category = 'shot differential')

Examples

```
COL_dS_regular_20242025 <- team_edge_zone_time(  
  team      = 21,  
  season    = 20242025,  
  game_type = 2,  
  category  = 'dS'  
)
```

team_game_report	<i>Access various reports for a season, game type, and category for all the teams by game</i>
------------------	---

Description

team_game_report() returns the game-level version of a selected team stats report, with one row per team per game and the category-specific metric columns exposed by the stats API.

Usage

```
team_game_report(  
  season = season_now(),  
  game_type = game_type_now(),  
  category = "summary"  
)
```

Arguments

- season integer in YYYYYYYYY (e.g., 20242025); see [seasons\(\)](#) for reference
- game_type integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see [seasons\(\)](#) for reference; most functions will NOT support pre-season
- category character (e.g., 'leadingtrailing'); see [team_report_configurations\(\)](#) for reference

Value

data.frame with one row per game per team

Examples

```
situational_team_game_report_playoffs_20212022 <- team_game_report(  
  season    = 20212022,  
  game_type = 3,  
  category  = 'leadingtrailing'  
)
```

team_logos	<i>Access all the team logos</i>
------------	----------------------------------

Description

team_logos() returns records-site logo metadata with one row per team logo interval, including team ID, logo URL fields, and start/end season IDs.

Usage

```
team_logos()
```

Value

data.frame with one row per logo

Examples

```
all_team_logos <- team_logos()
```

team_month_schedule	<i>Access the schedule for a team and month</i>
---------------------	---

Description

team_month_schedule() returns the same normalized club-schedule rows as [team_season_schedule\(\)](#), restricted to the requested YYYY-MM month.

Usage

```
team_month_schedule(team = 1, month = "now")
```

Arguments

- | | |
|-------|---|
| team | integer ID (e.g., 21), character full name (e.g., 'Colorado Avalanche'), OR three-letter code (e.g., 'COL'); see teams() for reference; ID is preferable as there now exists duplicate three-letter codes (i.e., 'UTA' for 'Utah Hockey Club' and 'Utah Mammoth') |
| month | character in 'YYYY-MM' (e.g., '2025-01'); see seasons() for reference |

Value

data.frame with one row per game

Examples

```
COL_schedule_December_2025 <- team_month_schedule(  
  team = 21,  
  month = '2025-12'  
)
```

team_prospects	<i>Access the prospects for a team and position</i>
----------------	---

Description

team_prospects() returns a team’s prospect list for one position group, including prospect/player IDs, names, position, size, birth data, and prospect status fields when available.

Usage

```
team_prospects(team = 1, position = "forwards")
```

Arguments

team	integer ID (e.g., 21), character full name (e.g., 'Colorado Avalanche'), OR three-letter code (e.g., 'COL'); see teams() for reference; ID is preferable as there now exists duplicate three-letter codes (i.e., 'UTA' for 'Utah Hockey Club' and 'Utah Mammoth')
position	character of 'f'/'forwards', 'd'/'defensemen', or 'g'/'goalies'

Value

data.frame with one row per player

Examples

```
COL_forward_prospects <- team_prospects(  
  team = 21,  
  position = 'F'  
)
```

`team_report_configurations`*Access the configurations for team reports*

Description

`team_report_configurations()` returns the team-report configuration block from the stats API, including valid report categories, fields, filters, and split options accepted by [team_season_report\(\)](#) and [team_game_report\(\)](#).

Usage

```
team_report_configurations()
```

```
team_report_configs()
```

Value

list with various items

Examples

```
team_report_configs <- team_report_configurations()
```

`team_seasons`*Access the season(s) and game type(s) in which a team played*

Description

`team_seasons()` returns the seasons and game type IDs available for a team in the public club-stats API.

Usage

```
team_seasons(team = 1)
```

Arguments

<code>team</code>	integer ID (e.g., 21), character full name (e.g., 'Colorado Avalanche'), OR three-letter code (e.g., 'COL'); see teams() for reference; ID is preferable as there now exists duplicate three-letter codes (i.e., 'UTA' for 'Utah Hockey Club' and 'Utah Mammoth')
-------------------	---

Value

data.frame with one row per season

Examples

```
COL_seasons <- team_seasons(team = 21)
```

team_season_report	<i>Access various reports for a season, game type, and category for all the teams by season</i>
--------------------	---

Description

team_season_report() returns a season-level team report for a selected stats category, with one row per team and the metric columns defined by that category.

Usage

```
team_season_report(
  season = season_now(),
  game_type = game_type_now(),
  category = "summary"
)
```

Arguments

season	integer in YYYYYYYY (e.g., 20242025); see seasons() for reference
game_type	integer in 1:3 (where 1 = pre-season, 2 = regular season, 3 = playoff/post-season) OR character of 'pre', 'regular', or 'playoff'/'post'; see seasons() for reference; most functions will NOT support pre-season
category	character (e.g., 'leadingtrailing'); see team_report_configurations() for reference

Value

data.frame with one row per team

Examples

```
situational_team_season_report_playoffs_20212022 <- team_season_report(
  season = 20212022,
  game_type = 3,
  category = 'leadingtrailing'
)
```

team_season_schedule	<i>Access the schedule for a team and season</i>
----------------------	--

Description

team_season_schedule() returns one team’s season schedule with one row per game and normalized game, opponent, score/status, venue, broadcast, and link fields.

Usage

```
team_season_schedule(team = 1, season = "now")
```

Arguments

- team integer ID (e.g., 21), character full name (e.g., ‘Colorado Avalanche’), OR three-letter code (e.g., ‘COL’); see [teams\(\)](#) for reference; ID is preferable as there now exists duplicate three-letter codes (i.e., ‘UTA’ for ‘Utah Hockey Club’ and ‘Utah Mammoth’)
- season integer in YYYYYYYY (e.g., 20242025); see [seasons\(\)](#) for reference

Value

data.frame with one row per game

Examples

```
COL_schedule_20252026 <- team_season_schedule(  
  team = 21,  
  season = 20252026  
)
```

team_season_statistics	<i>Access the statistics for all the teams by season and game type</i>
------------------------	--

Description

team_season_statistics() returns records-site team totals by team, season, and game type, including win/loss, goal, shot, standings-point, and related season-total fields.

Usage

```
team_season_statistics()  
  
team_season_stats()
```

Value

data.frame with one row per team per season per game type

Examples

```
# May take >5s, so skip.
team_season_statistics <- team_season_statistics()
```

team_week_schedule	<i>Access the schedule for a team and week since a date</i>
--------------------	---

Description

team_week_schedule() returns the same normalized club-schedule rows as [team_season_schedule\(\)](#), restricted to the API's week window starting from the requested date.

Usage

```
team_week_schedule(team = 1, date = "now")
```

Arguments

team	integer ID (e.g., 21), character full name (e.g., 'Colorado Avalanche'), OR three-letter code (e.g., 'COL'); see teams() for reference; ID is preferable as there now exists duplicate three-letter codes (i.e., 'UTA' for 'Utah Hockey Club' and 'Utah Mammoth')
date	character in 'YYYY-MM-DD' (e.g., '2025-01-01'); see seasons() for reference

Value

data.frame with one row per game

Examples

```
COL_schedule_Family_Week_2025 <- team_week_schedule(
  team = 21,
  date = '2025-10-06'
)
```

tv_schedule	<i>Access the NHL Network TV schedule for a date</i>
-------------	--

Description

tv_schedule() returns NHL Network broadcast schedule rows for one date, with one row per program/broadcast item.

Usage

```
tv_schedule(date = "now")
```

Arguments

date	character in 'YYYY-MM-DD' (e.g., '2025-01-01'); see seasons() for reference
------	---

Value

data.frame with one row per program

Examples

```
tv_schedule_Halloween_2025 <- tv_schedule(date = '2025-10-31')
```

venues	<i>Access all the venues</i>
--------	------------------------------

Description

venues() returns the records-site venue catalog with one row per venue and location/name fields.

Usage

```
venues()
```

Value

data.frame with one row per venue

Examples

```
all_venues <- venues()
```

wsc_play_by_play	<i>Access the World Showcase (WSC) play-by-play for a game</i>
------------------	--

Description

wsc_play_by_play() retrieves the World Showcase (WSC) play-by-play for a game as a `data.frame` where each row represents an event. The returned schema follows the same cleaned public-facing naming as `gc_play_by_play()`, including `servedByPlayerId`, `goalieInNetId`, and `utc` immediately after `secondsElapsedInGame` while omitting GC-only clip fields. It also includes the same HTML-report-derived on-ice player ID columns added to the GC output, including dynamically expanded overflow skater slots when needed. HTML report skater and goalie IDs are returned whenever they can be matched back to a supported row, even when the raw `situationCode` is stale. Use `add_shift_times()` with `shift_chart()` (or `shift_charts()`) to add on-ice shift timing columns.

Usage

```
wsc_play_by_play(game = 2023030417)
```

```
wsc_pbp(game = 2023030417)
```

Arguments

game integer ID (e.g., 2025020275); see `games()` for reference

Value

`data.frame` with one row per event (play)

Examples

```
# May take >5s, so skip.
wsc_pbp_Martin_Necas_legacy_game <- wsc_play_by_play(game = 2025020275)
```

wsc_play_by_plays	<i>Access the World Showcase (WSC) play-by-plays for a season</i>
-------------------	---

Description

wsc_play_by_plays() downloads the cleaned season-level World Showcase parquet snapshot and pads `situationCode` to the four-character public format.

Usage

```
wsc_play_by_plays(season = 20242025)
```

```
wsc_pbps(season = 20242025)
```

Arguments

season integer in YYYYYYYYY (e.g., 20242025); see [seasons\(\)](#) for reference

Value

data.frame with one row per event (play) per game

Examples

```
# May take >5s, so skip.  
wsc_pbps_20212022 <- wsc_play_by_plays(season = 20212022)
```

wsc_play_by_plays_raw *Access the raw World Showcase (WSC) play-by-plays for a season*

Description

wsc_play_by_plays_raw() downloads the stored season-level World Showcase parquet snapshot and returns the raw rows without public-schema cleanup or situation-code padding.

Usage

```
wsc_play_by_plays_raw(season = 20242025)  
  
wsc_pbps_raw(season = 20242025)
```

Arguments

season integer in YYYYYYYYY (e.g., 20242025); see [seasons\(\)](#) for reference

Value

data.frame with one row per raw event (play) per game

Examples

```
# May take >5s, so skip.  
wsc_pbps_raw_20212022 <- wsc_play_by_plays_raw(season = 20212022)
```

wsc_play_by_play_raw	<i>Access the raw World Showcase (WSC) play-by-play for a game</i>
----------------------	--

Description

wsc_play_by_play_raw() returns the raw flattened World Showcase play-by-play as served by the NHL API for one game. Use [wsc_play_by_play\(\)](#) for the cleaned public schema that repairs common clock/order defects and appends the derived public columns.

Usage

```
wsc_play_by_play_raw(game = 2023030417)
```

```
wsc_pbp_raw(game = 2023030417)
```

Arguments

game	integer ID (e.g., 2025020275); see games() for reference
------	--

Value

data.frame with one row per event (play)

Examples

```
wsc_raw_Martin_Necas_legacy_game <- wsc_play_by_play_raw(game = 2025020275)
```

wsc_summary	<i>Access the World Showcase (WSC) summary for a game</i>
-------------	---

Description

wsc_summary() returns the World Showcase game-story payload for one game, including recap/story blocks, team metadata, scoring state, and supporting media/link fields when that endpoint provides them.

Usage

```
wsc_summary(game = 2023030417)
```

Arguments

game	integer ID (e.g., 2025020275); see games() for reference
------	--

Value

list of various items

Examples

```
wsc_summary_Martin_Necas_legacy_game <- wsc_summary(game = 2025020275)
```

```
x_game_cumulative_expected_goals
```

Save an X (Twitter) shareable cumulative expected goals (xG) plot

Description

`x_game_cumulative_expected_goals()` is the X/Twitter-sized companion to [ig_game_cumulative_expected_goals\(\)](#). It uses the same xG-scored cumulative series on a 1200-by-675-style canvas.

Usage

```
x_game_cumulative_expected_goals(game = 2023030417, model = NULL, save = TRUE)
```

```
x_game_cum_xG(game = 2023030417, model = NULL)
```

Arguments

<code>game</code>	integer ID (e.g., 2025020275); see games() for reference
<code>model</code>	deprecated legacy model selector; ignored
<code>save</code>	logical; use FALSE to draw on the active graphics device during tests or custom workflows

Value

NULL, invisibly

Examples

```
# May take >5s, so skip.
x_game_cumulative_expected_goals(
  game = 2023030417,
  save = FALSE
)
```

x_game_shot_locations *Save an X (Twitter) shareable shot-location plot for a game*

Description

x_game_shot_locations() is the X/Twitter-sized companion to [ig_game_shot_locations\(\)](#). It uses the same xG-scored shot map and legend logic on a 1200-by-675-style canvas.

Usage

```
x_game_shot_locations(  
  game = 2023030417,  
  team = "home",  
  model = NULL,  
  save = TRUE  
)
```

```
x_game_shot_locs(game = 2023030417, team = "home", model = NULL)
```

Arguments

game	integer ID (e.g., 2025020275); see games() for reference
team	character of 'h'/'home' or 'a'/'away'
model	deprecated legacy model selector; ignored
save	logical; use FALSE to draw on the active graphics device during tests or custom workflows

Value

NULL, invisibly

Examples

```
# May take >5s, so skip.  
x_game_shot_locations(  
  game = 2023030417,  
  team = 'H',  
  save = FALSE  
)
```

Index

add_deltas, 6
add_goalie_biometrics, 7
add_shift_times, 8
add_shift_times(), 31, 122
add_shooter_biometrics, 8
attendance, 9
attendance(), 34
award_winners, 10
award_winners(), 35
awards, 9
awards(), 35

boxscore, 10
boxscore(), 46
bracket, 11
bracket(), 35, 54, 89

calculate_expected_goals, 11
calculate_expected_goals(), 75, 76
calculate_shift_times_by_situation, 12
calculate_xG
 (calculate_expected_goals), 11
coach_career_statistics, 14
coach_career_statistics(), 39
coach_career_stats
 (coach_career_statistics), 14
coach_franchise_statistics, 14
coach_franchise_stats
 (coach_franchise_statistics),
 14
coaches, 13
coaches(), 39
combine_reports, 15
contracts, 15
countries, 16
countries(), 29, 36, 50

draft_picks, 17
draft_picks(), 37
draft_prospects, 17

draft_rankings, 18
draft_rankings(), 37
draft_tracker, 18
draft_tracker(), 38
drafts, 16
drafts(), 18, 36, 37
draw_NHL_rink, 19

espn_futures, 19
espn_futures(), 42
espn_game_odds, 20
espn_game_odds(), 40
espn_game_summary, 21
espn_game_summary(), 40
espn_games, 20
espn_games(), 20, 21, 23, 40, 41
espn_injuries, 21
espn_injuries(), 42
espn_pbp (espn_play_by_play), 23
espn_play_by_play, 23
espn_play_by_play(), 41
espn_player_summary, 22
espn_player_summary(), 22, 38
espn_players, 22
espn_players(), 22, 38
espn_team_summary, 24
espn_team_summary(), 23, 42
espn_teams, 23
espn_teams(), 24, 43
espn_transactions, 24
espn_transactions(), 43
expansion_draft_picks, 25
expansion_drafts, 25

franchise_playoff_situational_results,
 26
franchise_season_statistics, 27
franchise_season_statistics(), 44
franchise_season_stats
 (franchise_season_statistics),

- 27
- franchise_statistics, 27
- franchise_stats (franchise_statistics), 27
- franchise_team_statistics, 28
- franchise_team_statistics(), 44
- franchise_team_stats (franchise_team_statistics), 28
- franchise_versus_franchise, 28
- franchise_versus_franchise(), 45
- franchise_vs_franchise (franchise_versus_franchise), 28
- franchises, 26
- franchises(), 43
- game_odds, 29
- game_odds(), 50
- game_rosters, 30
- game_type_now, 30
- game_type_now(), 53
- games, 29
- games(), 10, 30, 31, 33, 45–47, 55, 62, 75, 76, 84, 90, 92, 122, 124–126
- gc_pbp (gc_play_by_play), 31
- gc_pbp_raw (gc_play_by_play_raw), 33
- gc_pbps (gc_play_by_plays), 31
- gc_pbps_raw (gc_play_by_plays_raw), 32
- gc_play_by_play, 31
- gc_play_by_play(), 7–9, 12, 33, 47, 84
- gc_play_by_play_raw, 33
- gc_play_by_plays, 31
- gc_play_by_plays(), 7–9, 12
- gc_play_by_plays_raw, 32
- gc_summary, 33
- gc_summary(), 41, 46
- general_managers, 34
- get_attendance, 34
- get_award_winners, 35
- get_awards, 35
- get_bracket, 35
- get_configuration, 36
- get_countries, 36
- get_draft_picks, 37
- get_draft_rankings, 37
- get_draft_tracker, 38
- get_drafts, 36
- get_espn_athlete, 38
- get_espn_athletes, 38
- get_espn_coach, 39
- get_espn_coach_career, 39
- get_espn_coaches, 39
- get_espn_event, 40
- get_espn_event_odds, 40
- get_espn_event_officials, 41
- get_espn_event_play_by_play, 41
- get_espn_event_stars, 41
- get_espn_events, 40
- get_espn_futures, 42
- get_espn_injuries, 42
- get_espn_team, 42
- get_espn_teams, 43
- get_espn_transactions, 43
- get_franchise_season_by_season, 44
- get_franchise_team_totals, 44
- get_franchise_vs_franchise, 45
- get_franchises, 43
- get_game_boxscore, 46
- get_game_landing, 46
- get_game_story, 47
- get_games, 45
- get_gc_play_by_play, 47
- get_glossary, 48
- get_goalie_leaders, 48
- get_goalie_milestones, 49
- get_goalie_statistics, 49
- get_goalies, 48
- get_officials, 49
- get_partner_odds, 50
- get_player_game_log, 51
- get_player_landing, 51
- get_players, 50
- get_schedule, 52
- get_scoreboards, 52
- get_scores, 53
- get_season_now, 53
- get_seasons, 53
- get_series, 54
- get_series_schedule, 54
- get_shift_charts, 54
- get_skater_leaders, 55
- get_skater_milestones, 56
- get_skater_statistics, 56
- get_skaters, 55
- get_spotlight_players, 56
- get_standings, 57
- get_standings_information, 57

get_streams, 58
get_team_prospects, 58
get_team_roster, 59
get_team_roster_statistics, 59
get_team_schedule, 60
get_team_scoreboard, 60
get_team_seasons, 61
get_team_statistics, 61
get_teams, 58
get_tv_schedule, 61
get_venues, 62
get_wsc_play_by_play, 62
glossary, 63
glossary(), 48
gms (general_managers), 34
goalie_edge_5_vs_5
 (goalie_edge_five_versus_five),
 63
goalie_edge_five_versus_five, 63
goalie_edge_leaders, 64
goalie_edge_save_percentage, 65
goalie_edge_seasons, 66
goalie_edge_seasons(), 64, 65, 67
goalie_edge_shot_location, 66
goalie_edge_summary, 67
goalie_game_report, 68
goalie_game_report(), 49, 72
goalie_game_scoring, 69
goalie_game_statistics, 69
goalie_game_stats
 (goalie_game_statistics), 69
goalie_leaders, 70
goalie_leaders(), 48
goalie_milestones, 71
goalie_milestones(), 49
goalie_regular_statistics, 71
goalie_regular_stats
 (goalie_regular_statistics), 71
goalie_report_configs
 (goalie_report_configurations),
 72
goalie_report_configurations, 72
goalie_report_configurations(), 36, 68,
 73
goalie_scoring, 72
goalie_season_report, 73
goalie_season_report(), 49, 72
goalie_season_statistics, 74
goalie_season_stats
 (goalie_season_statistics), 74
goalie_series_statistics, 74
goalie_series_stats
 (goalie_series_statistics), 74
goalie_statistics, 75
goalie_stats (goalie_statistics), 75
graphics::points(), 19

ig_game_cum_xG
 (ig_game_cumulative_expected_goals),
 75
ig_game_cumulative_expected_goals, 75
ig_game_cumulative_expected_goals(),
 125
ig_game_shot_locations, 76
ig_game_shot_locations(), 126
ig_game_shot_locs
 (ig_game_shot_locations), 76

location, 77
lottery_odds, 77

nhlscraper, 78
nhlscraper-package (nhlscraper), 78

officials, 79
officials(), 49

penalty_shots, 80
penalty_shots(), 83
ping, 80
player_game_log, 81
player_game_log(), 51
player_seasons, 82
player_summary, 82
player_summary(), 51
players, 81
players(), 48, 50, 51, 55, 81, 82, 94–98
playoff_season_statistics, 83
playoff_season_stats
 (playoff_season_statistics), 83
ps, 83
pss (penalty_shots), 80

replay, 84
replays, 84
roster, 85
roster(), 59
roster_statistics, 86

- roster_statistics(), 59
- roster_stats(roster_statistics), 86
- schedule, 87
- schedule(), 52
- scores, 87
- scores(), 52, 53
- season_now, 88
- season_now(), 53
- seasons, 88
- seasons(), 11, 19, 20, 32, 35, 48, 51–55, 57, 59, 60, 62, 68, 70, 73, 81, 84–87, 89, 91, 99, 100, 103, 105, 114, 115, 118–121, 123
- series, 89
- series(), 54, 89
- series_schedule, 89
- series_schedule(), 54
- shift_chart, 90
- shift_chart(), 8, 12, 31, 54, 90, 122
- shift_chart_summaries, 91
- shift_chart_summary, 92
- shift_chart_summary(), 12
- shift_charts, 91
- shift_charts(), 8, 12, 31, 122
- shifts, 90
- skater_edge_leaders, 92
- skater_edge_seasons, 93
- skater_edge_seasons(), 92, 94–98
- skater_edge_shot_location, 93
- skater_edge_shot_speed, 94
- skater_edge_skating_distance, 95
- skater_edge_skating_speed, 96
- skater_edge_summary, 97
- skater_edge_zone_time, 98
- skater_game_report, 99
- skater_game_report(), 56, 102
- skater_leaders, 100
- skater_leaders(), 55
- skater_milestones, 100
- skater_milestones(), 56
- skater_playoff_statistics, 101
- skater_playoff_stats
 - (skater_playoff_statistics), 101
- skater_regular_statistics, 101
- skater_regular_stats
 - (skater_regular_statistics), 101
- skater_report_configs
 - (skater_report_configurations), 102
- skater_report_configurations, 102
- skater_report_configurations(), 36, 99, 103
- skater_season_report, 102
- skater_season_report(), 56, 102
- skater_season_statistics, 103
- skater_season_stats
 - (skater_season_statistics), 103
- skater_series_statistics, 104
- skater_series_stats
 - (skater_series_statistics), 104
- skater_statistics, 104
- skater_stats(skater_statistics), 104
- spotlight_players, 105
- spotlight_players(), 56
- standings, 105
- standings(), 57
- standings_rules, 106
- standings_rules(), 57
- streams, 106
- streams(), 58
- team_edge_leaders, 107
- team_edge_seasons, 108
- team_edge_seasons(), 107, 109–113
- team_edge_shot_location, 108
- team_edge_shot_speed, 109
- team_edge_skating_distance, 110
- team_edge_skating_speed, 111
- team_edge_summary, 112
- team_edge_zone_time, 113
- team_game_report, 114
- team_game_report(), 61, 117
- team_logos, 115
- team_month_schedule, 115
- team_prospects, 116
- team_prospects(), 58
- team_report_configs
 - (team_report_configurations), 117
- team_report_configurations, 117
- team_report_configurations(), 36, 114, 118
- team_season_report, 118
- team_season_report(), 61, 117
- team_season_schedule, 119

`team_season_schedule()`, [60](#), [115](#), [120](#)
`team_season_statistics`, [119](#)
`team_season_stats`
 (`team_season_statistics`), [119](#)
`team_seasons`, [117](#)
`team_seasons()`, [61](#)
`team_week_schedule`, [120](#)
`teams`, [107](#)
`teams()`, [58–61](#), [85](#), [86](#), [109–113](#), [115–117](#),
 [119](#), [120](#)
`tv_schedule`, [121](#)
`tv_schedule()`, [61](#)

`venues`, [121](#)
`venues()`, [62](#)

`wsc_pbp` (`wsc_play_by_play`), [122](#)
`wsc_pbp_raw` (`wsc_play_by_play_raw`), [124](#)
`wsc_pbps` (`wsc_play_by_plays`), [122](#)
`wsc_pbps_raw` (`wsc_play_by_plays_raw`),
 [123](#)
`wsc_play_by_play`, [122](#)
`wsc_play_by_play()`, [7–9](#), [12](#), [62](#), [84](#), [124](#)
`wsc_play_by_play_raw`, [124](#)
`wsc_play_by_plays`, [122](#)
`wsc_play_by_plays()`, [7–9](#), [12](#)
`wsc_play_by_plays_raw`, [123](#)
`wsc_summary`, [124](#)
`wsc_summary()`, [41](#), [47](#)

`x_game_cum_xG`
 (`x_game_cumulative_expected_goals`),
 [125](#)
`x_game_cumulative_expected_goals`, [125](#)
`x_game_shot_locations`, [126](#)
`x_game_shot_locs`
 (`x_game_shot_locations`), [126](#)