

# Package ‘brightspaceR’

May 8, 2026

**Title** Access D2L 'Brightspace' Data Sets via the 'BDS' API

**Version** 0.1.0

**Description** Connect to the D2L 'Brightspace' Data Sets ('BDS') API via 'OAuth2', download all available datasets as tidy data frames with proper types, join them using convenience functions that know the foreign key relationships, and analyse student engagement, performance, and retention with ready-made analytics functions.

**Depends** R (>= 4.1.0)

**License** MIT + file LICENSE

**URL** <https://pcstrategyandopsco.github.io/brightspaceR/>,  
<https://github.com/pcstrategyandopsco/brightspaceR>

**BugReports** <https://github.com/pcstrategyandopsco/brightspaceR/issues>

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Imports** cli, config, curl, dplyr, httr2, lubridate, openssl, purrr,  
readr, rlang, stringr, tibble, tools, utils

**Suggests** httpptest2, knitr, pkgdown, rmarkdown, rstudioapi, testthat  
(>= 3.0.0), withr, yaml

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Peeyoosh Chandra [aut, cre]

**Maintainer** Peeyoosh Chandra <pc@strategyandops.co>

**Repository** CRAN

**Date/Publication** 2026-03-12 19:50:13 UTC

## Contents

bs_ads_filter . . . . .	3
bs_ads_job_status . . . . .	4
bs_api_version . . . . .	4
bs_assessment_performance . . . . .	5
bs_assignment_completion . . . . .	5
bs_auth . . . . .	6
bs_auth_refresh . . . . .	7
bs_auth_token . . . . .	8
bs_check_scopes . . . . .	9
bs_clean_names . . . . .	9
bs_config . . . . .	10
bs_config_set . . . . .	11
bs_course_engagement . . . . .	12
bs_course_summary . . . . .	12
bs_create_ads_job . . . . .	13
bs_deauth . . . . .	14
bs_download_ads . . . . .	14
bs_download_all . . . . .	15
bs_download_dataset . . . . .	15
bs_engagement_score . . . . .	16
bs_engagement_summary . . . . .	17
bs_enrich_enrollments . . . . .	18
bs_filter_test_users . . . . .	19
bs_get_ads . . . . .	20
bs_get_ads_schema . . . . .	20
bs_get_dataset . . . . .	21
bs_get_schema . . . . .	22
bs_get_timezone . . . . .	22
bs_grade_summary . . . . .	23
bs_has_token . . . . .	23
bs_identify_at_risk . . . . .	24
bs_join . . . . .	24
bs_join_content_progress . . . . .	25
bs_join_enrollments_grades . . . . .	26
bs_join_enrollments_orgunits . . . . .	27
bs_join_enrollments_roles . . . . .	27
bs_join_grades_objects . . . . .	28
bs_join_users_enrollments . . . . .	29
bs_list_ads . . . . .	29
bs_list_ads_jobs . . . . .	30
bs_list_ads_schemas . . . . .	30
bs_list_datasets . . . . .	31
bs_list_extracts . . . . .	31
bs_list_schemas . . . . .	32
bs_org_id . . . . .	32
bs_retention_summary . . . . .	33

<code>bs_ads_filter</code>	3
<code>bs_set_timezone</code> . . . . .	33
<code>bs_summarize_enrollments</code> . . . . .	34
<b>Index</b>	<b>35</b>

---

<code>bs_ads_filter</code>	<i>Build an ADS export filter</i>
----------------------------	-----------------------------------

---

### Description

Constructs a filter list for use with `bs_create_ads_job()` and `bs_get_ads()`. Produces the `[{Name, Value}]` array format that the Brightspace dataExport create endpoint expects.

### Usage

```
bs_ads_filter(
  start_date = NULL,
  end_date = NULL,
  parent_org_unit_id = NULL,
  roles = NULL
)
```

### Arguments

<code>start_date</code>	Start date (Date, POSIXct, or character in "YYYY-MM-DD" format). Optional.
<code>end_date</code>	End date (Date, POSIXct, or character in "YYYY-MM-DD" format). Optional.
<code>parent_org_unit_id</code>	Integer org unit ID to filter by. Optional.
<code>roles</code>	Integer vector of role IDs to filter by. Optional.

### Value

A list of `list(Name = ..., Value = ...)` filter objects.

### Examples

```
bs_ads_filter(start_date = "2024-01-01", end_date = "2024-12-31")
bs_ads_filter(parent_org_unit_id = 6606)
bs_ads_filter(roles = c(110, 120))
```

---

bs_ads_job_status	<i>Check ADS export job status</i>
-------------------	------------------------------------

---

**Description**

Check ADS export job status

**Usage**

```
bs_ads_job_status(job_id)
```

**Arguments**

job\_id            Export job ID returned by [bs\\_create\\_ads\\_job\(\)](#).

**Value**

A list with export\_job\_id, name, status (integer), status\_text (character), and submit\_date.

**Examples**

```
if (bs_has_token()) {  
  status <- bs_ads_job_status("abc-123")  
  status$status_text  
}
```

---

bs_api_version	<i>Get or set the Brightspace API version</i>
----------------	---

---

**Description**

Get or set the Brightspace API version

**Usage**

```
bs_api_version(version = NULL)
```

**Arguments**

version            If provided, sets the API version. If NULL, returns the current version.

**Value**

Character string of the API version.

**Examples**

```
bs_api_version()
bs_api_version("1.49")
```

---

```
bs_assessment_performance
```

*Summarize assessment performance per user per quiz*

---

**Description**

Aggregates quiz attempt data into per-user per-quiz performance summaries including best, average, and latest scores.

**Usage**

```
bs_assessment_performance(quiz_attempts)
```

**Arguments**

`quiz_attempts` A tibble from the Quiz Attempts dataset.

**Value**

A summarised tibble with one row per user per quiz per org unit.

**Examples**

```
if (bs_has_token()) {
  attempts <- bs_get_dataset("Quiz Attempts")
  bs_assessment_performance(attempts)
}
```

---

```
bs_assignment_completion
```

*Summarize assignment submission completion*

---

**Description**

Aggregates assignment submission data per assignment per org unit, including grading rates and score statistics.

**Usage**

```
bs_assignment_completion(assignment_submissions)
```

**Arguments**

assignment\_submissions

A tibble from the Assignment Submissions dataset.

**Value**

A summarised tibble with one row per assignment per org unit.

**Examples**

```
if (bs_has_token()) {
  submissions <- bs_get_dataset("Assignment Submissions")
  bs_assignment_completion(submissions)
}
```

---

bs\_auth

*Authenticate with Brightspace*

---

**Description**

Initiates an OAuth2 Authorization Code flow with PKCE to authenticate with the Brightspace Data Hub API. The resulting token is cached to disk for reuse across sessions and automatically refreshed when expired.

**Usage**

```
bs_auth(
  client_id = "",
  client_secret = "",
  instance_url = "",
  redirect_uri = "",
  scope = ""
)
```

**Arguments**

client_id	OAuth2 client ID. Resolved in order: this argument, config.yml (if present), BRIGHTSPACE_CLIENT_ID env var.
client_secret	OAuth2 client secret. Resolved in order: this argument, config.yml (if present), BRIGHTSPACE_CLIENT_SECRET env var.
instance_url	Your Brightspace instance URL (e.g., "https://myschool.brightspace.com"). Resolved in order: this argument, config.yml (if present), BRIGHTSPACE_INSTANCE_URL env var.
redirect_uri	The registered redirect URI. Must match the URI registered in your Brightspace OAuth2 app exactly. Supports both http://localhost (automatic capture via local server) and https://localhost (browser-based with URL paste).

scope OAuth2 scope string (space-separated). Resolved from config.yml or defaults to BDS + ADS scopes.

### Details

The first authentication requires an interactive R session (browser-based login). After that, cached credentials are used automatically — including in non-interactive scripts run via Rscript.

### Value

Invisibly returns TRUE on success.

### Examples

```
if (bs_has_token()) {  
  bs_auth()  
  bs_auth(  
    client_id = "my-client-id",  
    client_secret = "my-secret",  
    instance_url = "https://myschool.brightspace.com"  
  )  
}
```

---

bs\_auth\_refresh      *Authenticate with a refresh token*

---

### Description

Authenticates using an existing refresh token, without requiring browser interaction. Ideal for non-interactive scripts and scheduled jobs.

### Usage

```
bs_auth_refresh(  
  refresh_token,  
  client_id = Sys.getenv("BRIGHTSPACE_CLIENT_ID"),  
  client_secret = Sys.getenv("BRIGHTSPACE_CLIENT_SECRET"),  
  instance_url = Sys.getenv("BRIGHTSPACE_INSTANCE_URL"),  
  scope = ""  
)
```

### Arguments

refresh\_token The OAuth2 refresh token string.

client\_id OAuth2 client ID. Defaults to BRIGHTSPACE\_CLIENT\_ID environment variable.

client\_secret OAuth2 client secret. Defaults to BRIGHTSPACE\_CLIENT\_SECRET environment variable.

instance_url	Your Brightspace instance URL. Defaults to BRIGHTSPACE_INSTANCE_URL environment variable.
scope	OAuth2 scope.

**Value**

Invisibly returns TRUE on success.

**Examples**

```
if (bs_has_token()) {
  bs_auth_refresh(refresh_token = "my-refresh-token")
}
```

---

bs_auth_token	<i>Set Brightspace authentication token directly</i>
---------------	--

---

**Description**

Sets authentication credentials without going through the browser-based OAuth2 flow. Useful for non-interactive environments or when you already have a valid token.

**Usage**

```
bs_auth_token(
  token,
  instance_url,
  client_id = Sys.getenv("BRIGHTSPACE_CLIENT_ID"),
  client_secret = Sys.getenv("BRIGHTSPACE_CLIENT_SECRET")
)
```

**Arguments**

token	A token list with at least an access_token field. Can also include refresh_token, expires_in, etc.
instance_url	Your Brightspace instance URL.
client_id	OAuth2 client ID.
client_secret	OAuth2 client secret.

**Value**

Invisibly returns TRUE.

---

bs_check_scopes	<i>Test Brightspace API scope access</i>
-----------------	--

---

**Description**

Verifies which API capabilities are available with the current token by making lightweight test calls to each endpoint group. Useful for diagnosing 403 errors.

**Usage**

```
bs_check_scopes()
```

**Value**

A tibble with columns scope, endpoint, status ("OK" or error message), printed as a summary table.

**Examples**

```
if (bs_has_token()) {  
  bs_auth()  
  bs_check_scopes()  
}
```

---

bs_clean_names	<i>Convert column names from PascalCase to snake_case</i>
----------------	---

---

**Description**

Convert column names from PascalCase to snake\_case

**Usage**

```
bs_clean_names(df)
```

**Arguments**

df                    A data frame.

**Value**

A data frame with snake\_case column names.

**Examples**

```
df <- data.frame(UserId = 1, FirstName = "A")  
bs_clean_names(df)
```

---

**bs\_config***Read Brightspace credentials from a config file*

---

### Description

Reads Brightspace OAuth2 credentials from a YAML configuration file using the **config** package. The function looks for a brightspace key in the config file and returns the credentials as a named list.

### Usage

```
bs_config(  
  file = "config.yml",  
  profile = Sys.getenv("R_CONFIG_ACTIVE", "default")  
)
```

### Arguments

file	Path to the YAML config file. Defaults to "config.yml" in the working directory.
profile	Configuration profile to use. Defaults to the R_CONFIG_ACTIVE environment variable, or "default" if unset.

### Value

A named list with elements `client_id`, `client_secret`, `instance_url`, `redirect_uri`, and `scope`, or NULL if the file does not exist or the brightspace key is missing.

### Examples

```
if (bs_has_token()) {  
  # Read from default config.yml  
  cfg <- bs_config()  
  cfg$client_id  
  
  # Read from a custom file and profile  
  cfg <- bs_config(file = "my-config.yml", profile = "production")  
}
```

---

bs_config_set	<i>Create or update a Brightspace config file</i>
---------------	---

---

### Description

Interactively creates or updates a config.yml file with Brightspace OAuth2 credentials. If the file already exists, the brightspace section is updated while preserving other settings.

### Usage

```
bs_config_set(  
  client_id,  
  client_secret,  
  instance_url,  
  redirect_uri = "https://localhost:1410/",  
  scope = "datahub:dataexports:*",  
  file = "config.yml",  
  profile = "default"  
)
```

### Arguments

client_id	OAuth2 client ID.
client_secret	OAuth2 client secret.
instance_url	Your Brightspace instance URL (e.g., "https://myschool.brightspace.com").
redirect_uri	Redirect URI. Defaults to "https://localhost:1410/".
scope	OAuth2 scope. Defaults to "datahub:dataexports:*".
file	Path for the config file. Defaults to "config.yml".
profile	Configuration profile to write to. Defaults to "default".

### Value

Invisibly returns the file path.

### Examples

```
if (bs_has_token()) {  
  bs_config_set(  
    client_id = "my-client-id",  
    client_secret = "my-secret",  
    instance_url = "https://myschool.brightspace.com"  
  )  
}
```

bs\_course\_engagement *Calculate per-user per-course engagement metrics*

---

### Description

Computes engagement metrics from Learner Usage ADS data including progress percentage, days since last visit, and passes through all raw activity counts. No composite score is computed here — use `bs_engagement_score()` to add one.

### Usage

```
bs_course_engagement(learner_usage, tz = NULL)
```

### Arguments

`learner_usage` A tibble from the Learner Usage ADS.  
`tz` Timezone for date conversion. Defaults to `bs_get_timezone()`.

### Value

A tibble with all `learner_usage` identity columns plus computed metrics: `progress_pct`, `days_since_visit`.

### Examples

```
if (bs_has_token()) {  
  usage <- bs_get_ads("Learner Usage")  
  engagement <- bs_course_engagement(usage)  
}
```

---

bs\_course\_summary *Summarize course effectiveness*

---

### Description

Creates a per-course dashboard view with engagement, progress, and optionally award-based completion metrics. `completion_rate_progress` uses content progress (available from Learner Usage alone); `completion_rate_awards` uses certificate issuance (more authoritative but requires Awards Issued dataset).

### Usage

```
bs_course_summary(learner_usage, awards = NULL)
```

**Arguments**

learner\_usage A tibble from the Learner Usage ADS.  
 awards Optional tibble from the Awards Issued ADS. When provided, adds award-based completion rate.

**Value**

A summarised tibble with one row per course, sorted by n\_learners descending.

**Examples**

```
if (bs_has_token()) {
  usage <- bs_get_ads("Learner Usage")
  bs_course_summary(usage)

  awards <- bs_get_ads("Awards Issued")
  bs_course_summary(usage, awards = awards)
}
```

---

bs\_create\_ads\_job      *Create an ADS export job*

---

**Description**

Submits a new export job for the named ADS dataset. Use [bs\\_ads\\_job\\_status\(\)](#) to poll for completion, then [bs\\_download\\_ads\(\)](#) to retrieve the result.

**Usage**

```
bs_create_ads_job(name, filters = list())
```

**Arguments**

name Dataset name (case-insensitive). For example, "Learner Usage".  
 filters Optional filter list from [bs\\_ads\\_filter\(\)](#).

**Value**

A tibble with one row containing export\_job\_id, dataset\_id, name, status, status\_text, submit\_date.

**Examples**

```
if (bs_has_token()) {
  job <- bs_create_ads_job("Learner Usage")
  job$export_job_id
}
```

---

bs_deauth	<i>Clear Brightspace authentication</i>
-----------	---

---

**Description**

Removes cached credentials from the current session and optionally from disk.

**Usage**

```
bs_deauth(clear_cache = TRUE)
```

**Arguments**

clear\_cache      If TRUE (default), also removes the cached token from disk.

**Value**

Invisibly returns TRUE.

**Examples**

```
if (bs_has_token()) {  
  bs_deauth()  
}
```

---

bs_download_ads	<i>Download a completed ADS export</i>
-----------------	--

---

**Description**

Downloads the result of a completed ADS export job, unzips it, and returns a tidy tibble with proper types and snake\_case names.

**Usage**

```
bs_download_ads(job_id, dataset_name = NULL)
```

**Arguments**

job\_id            Export job ID.  
dataset\_name     Optional dataset name (used for schema lookup).

**Value**

A tibble of the dataset contents.

**Examples**

```
if (bs_has_token()) {  
  result <- bs_download_ads("abc-123", "Learner Usage")  
}
```

---

bs_download_all	<i>Download all available datasets</i>
-----------------	--

---

**Description**

Downloads all available datasets and returns them as a named list of tibbles. Names are snake\_case versions of the dataset names.

**Usage**

```
bs_download_all(extract_type = c("full", "diff"))
```

**Arguments**

extract\_type    Type of extract: "full" or "diff". Default "full".

**Value**

A named list of tibbles.

**Examples**

```
if (bs_has_token()) {  
  all_data <- bs_download_all()  
  all_data$users  
  all_data$org_units  
}
```

---

bs_download_dataset	<i>Download a dataset extract</i>
---------------------	-----------------------------------

---

**Description**

Downloads a specific dataset extract as a ZIP file, unzips it, reads the CSV, and returns a tidy tibble with proper types.

**Usage**

```
bs_download_dataset(  
  schema_id,  
  plugin_id,  
  extract_type = c("full", "diff"),  
  extract_id = NULL,  
  dataset_name = NULL  
)
```

**Arguments**

schema_id	Schema ID of the dataset.
plugin_id	Plugin ID of the dataset.
extract_type	Type of extract: "full" or "diff". Default "full".
extract_id	Specific extract ID. If NULL, downloads the latest.
dataset_name	Optional name of the dataset (used for schema lookup).

**Value**

A tibble of the dataset contents.

**Examples**

```
if (bs_has_token()) {  
  datasets <- bs_list_datasets()  
  users <- bs_download_dataset(  
    datasets$schema_id[1],  
    datasets$plugin_id[1]  
  )  
}
```

---

bs\_engagement\_score *Add a composite engagement score*

---

**Description**

Adds a weighted composite engagement\_score column to any tibble that has raw activity count columns. Default weights reflect relative effort/depth (login is passive, assignment is active). Users should override for their context.

**Usage**

```
bs_engagement_score(df, weights = list())
```

**Arguments**

df	A tibble with activity count columns.
weights	A named list of column-weight pairs to override defaults. Defaults: login_count = 1, quiz_completed = 3, assignment_completed = 5, discussion_posts_created = 2.

**Value**

The input tibble with an engagement\_score column appended.

**Examples**

```
if (bs_has_token()) {  
  usage <- bs_get_ads("Learner Usage")  
  scored <- bs_engagement_score(usage)  
  scored <- bs_engagement_score(usage, weights = list(login_count = 2))  
}
```

---

bs\_engagement\_summary *Summarize engagement by grouping dimension*

---

**Description**

Aggregates engagement metrics from Learner Usage data by course, department, or user.

**Usage**

```
bs_engagement_summary(learner_usage, by = c("course", "department", "user"))
```

**Arguments**

learner_usage	A tibble from the Learner Usage ADS.
by	Grouping dimension: "course", "department", or "user".

**Value**

A summarised tibble sorted by mean\_progress descending (or last\_activity for user grouping).

**Examples**

```
if (bs_has_token()) {  
  usage <- bs_get_ads("Learner Usage")  
  bs_engagement_summary(usage, by = "course")  
  bs_engagement_summary(usage, by = "department")  
  bs_engagement_summary(usage, by = "user")  
}
```

---

bs\_enrich\_enrollments *Enrich enrollments with org unit and user details*

---

### Description

Builds the enriched enrollment table by joining enrollments with org units and users, adding analysis-friendly column aliases. Original column names are preserved for compatibility; friendly aliases are added for readability.

### Usage

```
bs_enrich_enrollments(  
  enrollments,  
  org_units,  
  users,  
  course_type = "Course Offering"  
)
```

### Arguments

enrollments	A tibble from the Enrollments and Withdrawals dataset.
org_units	A tibble from the Org Units dataset.
users	A tibble from the Users dataset.
course_type	Org unit type to filter to (default "Course Offering"). Set to NULL to keep all org unit types.

### Value

A tibble with both original and friendly column names, filtered to the specified course type.

### Examples

```
if (bs_has_token()) {  
  enroll <- bs_get_dataset("Enrollments and Withdrawals")  
  org_units <- bs_get_dataset("Org Units")  
  users <- bs_get_dataset("Users")  
  enriched <- bs_enrich_enrollments(enroll, org_units, users)  
}
```

---

bs\_filter\_test\_users *Filter test users from a dataset*

---

## Description

Removes test/system accounts using a two-layer filter: ID string length and an optional exclusion list. This eliminates the repeated boilerplate of removing test users before analysis.

## Usage

```
bs_filter_test_users(  
  df,  
  min_id_length = 30,  
  exclusion_list = NULL,  
  id_col = "org_defined_id"  
)
```

## Arguments

df	A tibble containing user data.
min_id_length	Minimum character length of a real user ID (default 30). IDs shorter than this are assumed to be test accounts.
exclusion_list	Optional character vector of specific IDs to exclude.
id_col	Name of the column containing user IDs (default "org_defined_id").

## Value

A filtered tibble with test users removed.

## Examples

```
if (bs_has_token()) {  
  users <- bs_get_dataset("Users")  
  real_users <- bs_filter_test_users(users)  
  real_users <- bs_filter_test_users(users, exclusion_list = c("testuser01"))  
}
```

---

bs_get_ads	<i>Get an ADS dataset by name (convenience wrapper)</i>
------------	---

---

**Description**

High-level function that finds the dataset by name, creates an export job, polls until complete, downloads the result, and returns a tidy tibble. Intended for interactive use.

**Usage**

```
bs_get_ads(name, filters = list(), poll_interval = 5, timeout = 300)
```

**Arguments**

name	Dataset name (case-insensitive). For example, "Learner Usage".
filters	Optional filter list from <a href="#">bs_ads_filter()</a> .
poll_interval	Seconds between status checks. Default 5.
timeout	Maximum seconds to wait for completion. Default 300.

**Value**

A tibble of the dataset contents.

**Examples**

```
if (bs_has_token()) {  
  usage <- bs_get_ads("Learner Usage")  
  usage <- bs_get_ads("Learner Usage",  
    filters = bs_ads_filter(start_date = "2024-01-01"))  
}
```

---

bs_get_ads_schema	<i>Get the schema for an ADS dataset</i>
-------------------	--

---

**Description**

Get the schema for an ADS dataset

**Usage**

```
bs_get_ads_schema(dataset_name)
```

**Arguments**

dataset_name	Name of the dataset (will be normalized to snake_case).
--------------	---

**Value**

A schema list, or NULL if no schema is registered.

**Examples**

```
bs_get_ads_schema("Learner Usage")
```

---

bs_get_dataset	<i>Get a dataset by name</i>
----------------	------------------------------

---

**Description**

Convenience wrapper that finds a dataset by name, downloads the latest full extract, and returns a tidy tibble.

**Usage**

```
bs_get_dataset(name, extract_type = c("full", "diff"))
```

**Arguments**

name	Dataset name (case-insensitive partial match). For example, "Users", "Grade Results", "Org Units".
extract_type	Type of extract: "full" or "diff". Default "full".

**Value**

A tibble of the dataset contents.

**Examples**

```
if (bs_has_token()) {  
  users <- bs_get_dataset("Users")  
  grades <- bs_get_dataset("Grade Results")  
}
```

---

bs_get_schema	<i>Get the schema for a dataset</i>
---------------	-------------------------------------

---

**Description**

Get the schema for a dataset

**Usage**

```
bs_get_schema(dataset_name)
```

**Arguments**

dataset\_name    Name of the dataset (will be normalized to snake\_case).

**Value**

A schema list, or NULL if no schema is registered.

**Examples**

```
bs_get_schema("Users")  
bs_get_schema("Grade Results")
```

---

bs_get_timezone	<i>Get the current Brightspace analytics timezone</i>
-----------------	---

---

**Description**

Returns the timezone set by [bs\\_set\\_timezone\(\)](#), defaulting to "UTC".

**Usage**

```
bs_get_timezone()
```

**Value**

Character string of the timezone.

**Examples**

```
bs_get_timezone()
```

---

bs_grade_summary	<i>Summarize grades with percentages</i>
------------------	--

---

**Description**

Joins grade results with grade object definitions and calculates grade percentages.

**Usage**

```
bs_grade_summary(grade_results, grade_objects)
```

**Arguments**

`grade_results` A tibble from the Grade Results dataset.  
`grade_objects` A tibble from the Grade Objects dataset.

**Value**

A joined tibble with grade object name, type, max points, and calculated `grade_pct` and `grade_label`.

**Examples**

```
if (bs_has_token()) {  
  grades <- bs_get_dataset("Grade Results")  
  objects <- bs_get_dataset("Grade Objects")  
  bs_grade_summary(grades, objects)  
}
```

---

bs_has_token	<i>Check if authenticated with Brightspace</i>
--------------	--

---

**Description**

Check if authenticated with Brightspace

**Usage**

```
bs_has_token()
```

**Value**

Logical; TRUE if a token is available.

**Examples**

```
bs_has_token()
```

---

bs\_identify\_at\_risk     *Identify at-risk students*

---

### Description

Flags at-risk students from Learner Usage data based on configurable thresholds. Adds boolean risk flags and a composite risk score.

### Usage

```
bs_identify_at_risk(learner_usage, thresholds = list())
```

### Arguments

`learner_usage`     A tibble from the Learner Usage ADS.  
`thresholds`         A named list of thresholds to override defaults. Available thresholds: `progress` (default 25), `inactive_days` (default 14), `login_min` (default 2).

### Value

A tibble with all original columns plus risk flags (`never_accessed`, `low_progress`, `inactive`, `low_logins`), `risk_score` (0-4), and `risk_level` (ordered factor: Low, Medium, High, Critical), sorted by `risk_score` descending.

### Examples

```
if (bs_has_token()) {  
  usage <- bs_get_ads("Learner Usage")  
  at_risk <- bs_identify_at_risk(usage)  
  at_risk <- bs_identify_at_risk(usage, thresholds = list(progress = 30))  
}
```

---

bs\_join                     *Smart join two BDS tibbles*

---

### Description

Automatically detects shared key columns between two tibbles based on the schema registry and performs a join. Falls back to joining on common column names if schemas are not available.

### Usage

```
bs_join(df1, df2, type = c("left", "inner", "right", "full"))
```

**Arguments**

df1	First tibble.
df2	Second tibble.
type	Join type: "left" (default), "inner", "right", "full".

**Value**

A joined tibble.

**Examples**

```
if (bs_has_token()) {  
  users <- bs_get_dataset("Users")  
  enrollments <- bs_get_dataset("User Enrollments")  
  bs_join(users, enrollments)  
}
```

---

bs\_join\_content\_progress

*Join content objects with user progress*

---

**Description**

Left joins a content objects tibble with a content user progress tibble on content\_object\_id and org\_unit\_id.

**Usage**

```
bs_join_content_progress(content_objects, content_progress)
```

**Arguments**

content_objects	A tibble from the Content Objects dataset.
content_progress	A tibble from the Content User Progress dataset.

**Value**

A joined tibble.

**Examples**

```
if (bs_has_token()) {  
  content <- bs_get_dataset("Content Objects")  
  progress <- bs_get_dataset("Content User Progress")  
  bs_join_content_progress(content, progress)  
}
```

---

bs\_join\_enrollments\_grades

*Join enrollments with grade results*

---

**Description**

Left joins an enrollments tibble with a grade results tibble on `org_unit_id` and `user_id`.

**Usage**

```
bs_join_enrollments_grades(enrollments, grade_results)
```

**Arguments**

`enrollments` A tibble from the User Enrollments dataset.

`grade_results` A tibble from the Grade Results dataset.

**Value**

A joined tibble.

**Examples**

```
if (bs_has_token()) {  
  enrollments <- bs_get_dataset("User Enrollments")  
  grades <- bs_get_dataset("Grade Results")  
  bs_join_enrollments_grades(enrollments, grades)  
}
```

---

bs\_join\_enrollments\_orgunits  
*Join enrollments with org units*

---

**Description**

Left joins an enrollments tibble with an org units tibble on org\_unit\_id.

**Usage**

```
bs_join_enrollments_orgunits(enrollments, org_units)
```

**Arguments**

enrollments     A tibble from the User Enrollments dataset.  
org\_units        A tibble from the Org Units dataset.

**Value**

A joined tibble.

**Examples**

```
if (bs_has_token()) {  
  enrollments <- bs_get_dataset("User Enrollments")  
  org_units <- bs_get_dataset("Org Units")  
  bs_join_enrollments_orgunits(enrollments, org_units)  
}
```

---

bs\_join\_enrollments\_roles  
*Join enrollments with role details*

---

**Description**

Left joins an enrollments tibble with a role details tibble on role\_id.

**Usage**

```
bs_join_enrollments_roles(enrollments, role_details)
```

**Arguments**

enrollments     A tibble from the User Enrollments dataset.  
role\_details     A tibble from the Role Details dataset.

**Value**

A joined tibble.

**Examples**

```
if (bs_has_token()) {  
  enrollments <- bs_get_dataset("User Enrollments")  
  roles <- bs_get_dataset("Role Details")  
  bs_join_enrollments_roles(enrollments, roles)  
}
```

---

bs\_join\_grades\_objects

*Join grade results with grade objects*

---

**Description**

Left joins a grade results tibble with a grade objects tibble on grade\_object\_id and org\_unit\_id.

**Usage**

```
bs_join_grades_objects(grade_results, grade_objects)
```

**Arguments**

grade\_results    A tibble from the Grade Results dataset.  
grade\_objects    A tibble from the Grade Objects dataset.

**Value**

A joined tibble.

**Examples**

```
if (bs_has_token()) {  
  grades <- bs_get_dataset("Grade Results")  
  objects <- bs_get_dataset("Grade Objects")  
  bs_join_grades_objects(grades, objects)  
}
```

---

bs\_join\_users\_enrollments  
*Join users with enrollments*

---

**Description**

Left joins a users tibble with an enrollments tibble on user\_id.

**Usage**

```
bs_join_users_enrollments(users, enrollments)
```

**Arguments**

users            A tibble from the Users dataset.  
enrollments    A tibble from the User Enrollments dataset.

**Value**

A joined tibble.

**Examples**

```
if (bs_has_token()) {  
  users <- bs_get_dataset("Users")  
  enrollments <- bs_get_dataset("User Enrollments")  
  bs_join_users_enrollments(users, enrollments)  
}
```

---

bs\_list\_ads            *List available Advanced Data Sets*

---

**Description**

Retrieves all available ADS datasets from the Brightspace instance.

**Usage**

```
bs_list_ads()
```

**Value**

A tibble with columns: dataset\_id, name, description, category.

**Examples**

```
if (bs_has_token()) {  
  ads <- bs_list_ads()  
  ads  
}
```

---

bs\_list\_ads\_jobs      *List all submitted ADS export jobs*

---

**Description**

List all submitted ADS export jobs

**Usage**

```
bs_list_ads_jobs()
```

**Value**

A tibble of all submitted export jobs with columns: export\_job\_id, name, dataset\_id, status, status\_text, submit\_date.

**Examples**

```
if (bs_has_token()) {  
  bs_list_ads_jobs()  
}
```

---

bs\_list\_ads\_schemas      *List all registered ADS dataset schemas*

---

**Description**

List all registered ADS dataset schemas

**Usage**

```
bs_list_ads_schemas()
```

**Value**

A character vector of registered dataset names (snake\_case).

**Examples**

```
bs_list_ads_schemas()
```

---

bs\_list\_datasets      *List available Brightspace Data Sets*

---

**Description**

Retrieves all available BDS datasets from the Brightspace instance.

**Usage**

```
bs_list_datasets()
```

**Value**

A tibble with columns: schema\_id, plugin\_id, name, description, full\_download\_link, diff\_download\_link, created\_date.

**Examples**

```
if (bs_has_token()) {  
  datasets <- bs_list_datasets()  
  datasets  
}
```

---

bs\_list\_extracts      *List available extracts for a dataset*

---

**Description**

Retrieves available full and differential extracts for a specific dataset.

**Usage**

```
bs_list_extracts(schema_id, plugin_id)
```

**Arguments**

schema_id	Schema ID of the dataset.
plugin_id	Plugin ID of the dataset.

**Value**

A tibble with columns: extract\_id, extract\_type, bds\_type, created\_date, download\_link, download\_size.

**Examples**

```
if (bs_has_token()) {
  datasets <- bs_list_datasets()
  extracts <- bs_list_extracts(datasets$schema_id[1], datasets$plugin_id[1])
}
```

---

bs_list_schemas	<i>List all registered dataset schemas</i>
-----------------	--

---

**Description**

List all registered dataset schemas

**Usage**

```
bs_list_schemas()
```

**Value**

A character vector of registered dataset names (snake\_case).

**Examples**

```
bs_list_schemas()
```

---

bs_org_id	<i>Get the root organisation ID</i>
-----------	-------------------------------------

---

**Description**

Calls /d21/api/lp/(version)/organization/info and returns the org identifier.

**Usage**

```
bs_org_id()
```

**Value**

Character string of the root org unit ID.

**Examples**

```
if (bs_has_token()) {
  bs_org_id()
}
```

---

bs\_retention\_summary *Summarize retention and dropout rates*

---

**Description**

Calculates retention metrics by course or department, including start rates, completion rates, and dropout rates.

**Usage**

```
bs_retention_summary(learner_usage, by = c("course", "department"))
```

**Arguments**

learner\_usage A tibble from the Learner Usage ADS.  
by Grouping dimension: "course" or "department".

**Value**

A summarised tibble sorted by completion\_rate.

**Examples**

```
if (bs_has_token()) {  
  usage <- bs_get_ads("Learner Usage")  
  bs_retention_summary(usage, by = "course")  
}
```

---

bs\_set\_timezone *Set the timezone for Brightspace analytics*

---

**Description**

All analytics functions use this timezone for converting date columns.

**Usage**

```
bs_set_timezone(tz)
```

**Arguments**

tz A valid timezone string from [OlsonNames\(\)](#).

**Value**

Invisibly returns the timezone string.

**Examples**

```
bs_set_timezone("Pacific/Auckland")
bs_set_timezone("America/New_York")
```

---

```
bs_summarize_enrollments
```

*Summarize enrollments to one row per user per course*

---

**Description**

Collapses enrollment records to keep only the latest enrollment date for each user-course combination.

**Usage**

```
bs_summarize_enrollments(enriched_enrollments, event_type = "Enroll")
```

**Arguments**

`enriched_enrollments` An enriched enrollment tibble (from [bs\\_enrich\\_enrollments\(\)](#)).

`event_type` The event type to filter to (default "Enroll").

**Value**

A tibble with one row per user per course.

**Examples**

```
if (bs_has_token()) {
  enriched <- bs_enrich_enrollments(enroll, org_units, users)
  summary <- bs_summarize_enrollments(enriched)
}
```

# Index

`bs_ads_filter`, 3  
`bs_ads_filter()`, 13, 20  
`bs_ads_job_status`, 4  
`bs_ads_job_status()`, 13  
`bs_api_version`, 4  
`bs_assessment_performance`, 5  
`bs_assignment_completion`, 5  
`bs_auth`, 6  
`bs_auth_refresh`, 7  
`bs_auth_token`, 8  
`bs_check_scopes`, 9  
`bs_clean_names`, 9  
`bs_config`, 10  
`bs_config_set`, 11  
`bs_course_engagement`, 12  
`bs_course_summary`, 12  
`bs_create_ads_job`, 13  
`bs_create_ads_job()`, 3, 4  
`bs_deauth`, 14  
`bs_download_ads`, 14  
`bs_download_ads()`, 13  
`bs_download_all`, 15  
`bs_download_dataset`, 15  
`bs_engagement_score`, 16  
`bs_engagement_score()`, 12  
`bs_engagement_summary`, 17  
`bs_enrich_enrollments`, 18  
`bs_enrich_enrollments()`, 34  
`bs_filter_test_users`, 19  
`bs_get_ads`, 20  
`bs_get_ads()`, 3  
`bs_get_ads_schema`, 20  
`bs_get_dataset`, 21  
`bs_get_schema`, 22  
`bs_get_timezone`, 22  
`bs_get_timezone()`, 12  
`bs_grade_summary`, 23  
`bs_has_token`, 23  
`bs_identify_at_risk`, 24  
`bs_join`, 24  
`bs_join_content_progress`, 25  
`bs_join_enrollments_grades`, 26  
`bs_join_enrollments_orgunits`, 27  
`bs_join_enrollments_roles`, 27  
`bs_join_grades_objects`, 28  
`bs_join_users_enrollments`, 29  
`bs_list_ads`, 29  
`bs_list_ads_jobs`, 30  
`bs_list_ads_schemas`, 30  
`bs_list_datasets`, 31  
`bs_list_extracts`, 31  
`bs_list_schemas`, 32  
`bs_org_id`, 32  
`bs_retention_summary`, 33  
`bs_set_timezone`, 33  
`bs_set_timezone()`, 22  
`bs_summarize_enrollments`, 34  
  
`OlsonNames()`, 33