Package 'boxr'

November 19, 2019

Type Package

Title Interface for the 'Box.com API'

Version 0.3.5

URL https://github.com/r-box/boxr/

BugReports https://github.com/r-box/boxr/issues

Description An R interface for the remote file hosting service 'Box' (<https://www.box.com/>). In addition to uploading and downloading files, this package includes functions which mirror base R operations for local files, (e.g. box_load(), box_save(), box_read(), box_setwd(), etc.), as well as 'git' style functions for entire directories (e.g. box_fetch(), box_push()).

License MIT + file LICENSE

Imports httr (>= 1.1.0), httpuv, assertthat, stringr, dplyr, digest, bit64, rio, mime, glue, fs, utils, stats, rlang

Suggests clipr (>= 0.3.0), testthat, knitr, rmarkdown, purrr, here, conflicted, usethis, jsonlite, openssl, jose, sodium, gargle (>= 0.3.0), png

VignetteBuilder knitr

RoxygenNote 6.1.1

Encoding UTF-8

NeedsCompilation no

Author Brendan Rocks [aut], Ian Lyttle [aut, cre] (<https://orcid.org/0000-0001-9962-4849>), Nate Day [aut] (<https://orcid.org/0000-0002-6714-8611>), Vincent Fulco [ctb], Alec Wong [ctb]

Maintainer Ian Lyttle <ian.lyttle@schneider-electric.com>

Repository CRAN

Date/Publication 2019-11-19 09:50:24 UTC

R topics documented:

`boxr-package` *boxr: access the Box API*

Description

A lightweight, *opinionated*, high-level R interface for the box.com API.

Details

This package has a documentation-website (created using pkgdown), containing:

- README.
- Get-started article, also accessible from R: `vignette("boxr")`.
- Function reference.

The boxr source-repository is at GitHub: `https://github.com/r-box/boxr`.

If you find anything you think might be a bug, please report it as a GitHub issue!

Happy hacking! :)

boxr_options *Get boxr options*

Description

This function gets the values of boxr's global options.

Usage

```
boxr_options()
```

Details

Options can be set in the usual way, using `options()`.

Value

`list`, current values of boxr options, with elements:

`boxr.interactive` `logical`, indicates if boxr is running in interactive mode.

`boxr.progress` `logical`, indicates to use progress-bars, if available.

`boxr.verbose` `logical`, indicates if boxr will use `cat()` to print to the console. Setting to `TRUE` may cause problems with `knitr`.

`boxr.wd` `list`, containing information on the Box working-directory: `id` (`numeric`), and `name` (`character`).

`boxr.wd.path` `character`, path to the Box working-directory.

`boxr.token` Object with S3 class `Token2.0` (`httr::Token2.0`).

boxr_S3_classes *boxr S3 Classes*

Description

boxr implements a series of S3 classes to manage the data returned by the Box API. These classes are built on `list`; if you wish to access the information directly, you can use `unclass(x)`.

Details

`boxr_file_reference`

- describes a file created, modified, or deleted at Box.
- returned by `box_ul()`, `box_save()`, `box_delete_file()`, etc.
- available methods: `print()`.

`boxr_folder_reference`

- describes a folder created or deleted at Box.
- returned by `box_dir_create()`, `box_delete_folder()`.
- available methods: `print()`.

`boxr_dir_wide_operation_result`

- describes the result of a directory-wide operation.
- returned by `box_fetch()` and `box_push()`.
- available methods: `print()`, `summary()`.

`boxr_object_list`

- describes a collection of files at Box.
- returned by `box_ls()`, `box_search()`, and related functions.
- available methods: `print()`, `as.data.frame()`.

`boxr_dir_comparison`

- describes the difference between directories.
- returned by the internal function `box_dir_diff()`.
- available methods: `print()`, `summary()`.

`box_add_description` *Add description to a Box file*

Description

This function will attach a description to a Box file; it will overwrite the Box file's existing description.

Usage

```
box_add_description(file_id, description)
```

Arguments

`file_id`	`numeric` or `character`, file ID at Box.
`description`	`character`, description caption for the file.

Details

Files hosted at Box can have small text-descriptions that you can be use to annotate files, or even to leave 'git commit' style messages.

Value

Object with S3 class `boxr_file_reference`.

box_auth *Authenticate to Box (interactive-app)*

Description

There are two common use-cases for `box_auth()`:

1. Connecting to box.com accounts from **boxr** for the first time.
2. Connecting to previously-connected box.com accounts.

In the first case, you will need to provide `box_auth()` with `client_id` and `client_secret`.

In the second case, you can call `box_auth()` with no arguments; the function will look for these in your R environment.

To run this function the first time, you will need access to the `client_id` and `client_secret` of a Box interactive-app. If you are using a work account, this information might be provided to you by your Box-admin team. If you are using a personal account, you will have to set up a Box interactive-app.

For both cases, these procedures are detailed in this boxr interactive-app article.

Usage

```
box_auth(client_id = NULL, client_secret = NULL, interactive = TRUE,
  cache = "~/.boxr-oauth", write.Renv, ...)
```

Arguments

`client_id`	`character`, the client id for the account to use.
`client_secret`	`character`, the client secret for the account to use.
`interactive`	`logical`, indicates that the authorization process will be interactive (requiring user input to the R console, and/or a visit to box.com).
`cache`	A logical value or a string. `TRUE` means to cache using the default cache file `.httr-oauth`, `FALSE` means don't cache, and `NA` means to guess using some sensible heuristics. A string means use the specified path as the cache file.
`write.Renv`	**deprecated**.
`...`	Other arguments passed to `httr::oauth2.0_token()`.

Value

`invisible(NULL)`, called for side-effects.

Side-effects

This function has some side effects which make subsequent calls to `box_auth()` easier:

- a browser window may be opened at box.com, for you to authorize to your Box app.
- a token file is written, according to the value of `cache`. The default behaviour is to write this file to `~/.boxr-oauth`.
- some global `options()` are set for your session to manage the token.
- environment variables `BOX_USER_ID`, `BOX_CLIENT_ID`, and `BOX_CLIENT_SECRET` are set.
- if these environment variables have changed, and you have the usethis package installed, it will copy some text to your clipboard that you can paste into your `.Renviron` file.
- a message is printed to the console.

See Also

`box_auth_service()` for authenticating to service-apps.

`httr::oauth2.0_token()` for details on how tokens are handled.

Box Developers: Setup with OAuth 2.0 documentation for setting up Box (interactive) apps with OAuth 2.0.

---

box_auth_on_attach *Authenticate to Box (interactive) automatically*

Description

This function is deprecated, and will be removed at the next release.

This function saves you the effort of typing `box_auth()` after the package loads. Executing `box_auth_on_attach(TRUE)` will mean that `boxr` will automatically attempt to authorize itself when 'attached' (e.g. `library(boxr)`), using the credentials from the current session.

Usage

```
box_auth_on_attach(auth_on_attach = FALSE)
```

Arguments

`auth_on_attach` `logical`, indicates if boxr should authenticate as soon as it's loaded.

Value

`invisible(NULL)`, called for side-effects.

Note

This is provided for convenience, but it's a bad idea to use, if:

- You'd like your code to be reproducible. Even if your collaborators have access to the same files on box.com, as the default behaviour is to require using `box_auth()`, code is likely to become irreproducible.
- You use more than one box.com account. Things could get rather confusing.

See Also

`box_auth()`

---

box_auth_service *Authenticate to Box (service-app)*

Description

How you authenticate to Box depends the Box-app through which you connect. A Box service-app can be useful for unattended jobs that need access to only a limited part of Box, e.g. one folder.

Use this function to access Box using a service-app.

To access a service-app, you will need a JSON web-token (JWT), generated by your Box-admin team. If you have a personal Box account, *you* are your Box-admin team. You specify the JWT either as `token_file`, the path to the JWT file, or as `token_text`, the text of the JWT.

Using JWT-authentication is more convenient than using standard OAuth2 authentication, as you do not have to go through the "OAuth Dance". This convenience brings additional considerations because the JWT file gives its bearer uninhibited access to anything the Box service-app can access. Accordingly, you are recommended to:

- give the service-account access to as little information as you need it to have, e.g. a single folder.
- keep the JWT file secure.

Usage

```
box_auth_service(token_file = NULL, token_text = NULL)
```

Arguments

`token_file`	`character`, path to JSON token-file. If not provided, the function will look for an environment variable `BOX_TOKEN_FILE`. If that is not there, it will try `~/.boxr-auth/token.json`.
`token_text`	`character`, JSON text. If this is provided, `token_file` is ignored.

Details

The default behavior of a service-app is to act on behalf of the service-account associated with the service-app. This is different from an interactive-app, which acts on behalf of the Box user who authenticates to it.

To use a service-app on a folder belonging to a Box user, either the Box user has to invite the service-account to collaborate on a folder belonging to the user, or the service-account has to invite the Box user to collaborate on a folder belonging to the service-account.

In either case, you can use `box_dir_invite()`.

For more details on Box service-apps, including how to create them, and service-app-based workflows, please read this boxr service-app article.

Value

Invisible `NULL`, called for side-effects.

Side-effects

This function has some side effects:

- some global `options()` are set for your session to manage the token.
- a message is printed to the console.

See Also

`box_auth()` for authenticating to interactive-apps.

`box_dir_invite()` for inviting a different account to collaborate on a Box folder.

Box Developers: Setup with JWT documentation for setting up Box (service) apps with JWT.

---

box_delete_file *Move files within Box, from/to trash directory*

Description

In the Box context, deleting a file moves it to a special folder within your Box account: 'Trash'. As of mid-2019, Box' default policy is to retain files in Trash for 30 days.

Usage

```
box_delete_file(file_id)

box_restore_file(file_id)

box_delete_folder(dir_id)

box_restore_folder(dir_id)
```

Arguments

file_id	numeric or character, file ID at Box.
dir_id	numeric or character, folder ID at Box.

Details

box_delete_file() Move a file to Trash.

box_restore_file() Restore a file from Trash.

box_delete_folder() Move a folder, including contents, to Trash.

box_restore_folder() Restore a folder, including contents, from Trash.

Value

box_delete_file() Object with S3 class boxr_file_reference.

box_restore_file() Object with S3 class boxr_file_reference.

box_delete_folder() Object with S3 class boxr_folder_reference.

box_restore_folder() Object with S3 class boxr_folder_reference.

---

box_dir_create *Create a Box directory*

Description

This will create a new folder at Box, with name dir_name, in the Box folder with ID parent_dir_id.

Usage

```
box_dir_create(dir_name, parent_dir_id = box_getwd())
```

Arguments

dir_name	character, name for new folder at Box.
parent_dir_id	character or numeric, ID for the parent folder at Box.

Value

Object with S3 class boxr_folder_reference.

See Also

box_delete_folder() to move Box folders to trash, box_ls() to list files in a Box folder.

box_dir_invite *Invite collaboration on a Box folder*

Description

Although this function can be used in all sorts of situations, it can be particularly useful in setting up a workflow with a service-account:

- If you are authenticated as a user, using box_auth(), you can invite the service account to collaborate on a folder in your *user* filespace. In this case, the shared folder will appear in the service-account file-space.
- If you are authenticated as as the service-account using box_auth_service(), you can invite your *user-account* to collaborate. In this case, the shared folder will appear in your user file-space.

Once you issue an invitation to create a collaboration, you cannot change it, e.g. you cannot change the role from "viewer" to "co-owner". However, you can delete the collaboration, then issue a *new* invitation. To delete a collaboration, you can use the Box web-portal.

The default role, i.e. permission level, for an invitation is "viewer". Legal values for role are "editor", "viewer", "previewer", "uploader", "previewer uploader", "viewer uploader", "co-owner", "owner".

Usage

```
box_dir_invite(dir_id, user_id, login = NULL, role = "viewer",
  can_view_path = FALSE)
```

Arguments

dir_id	numeric or character, folder ID at Box.
user_id	character ID for Box account to invite, e-mail address (login) will also work.
login	character email address of account to invite, can be used instead of user_id.
role	character role of the collaborator; default is "viewer".
can_view_path	logical indicates to allow the collaborator to navigate parent-folders at Box.

Details

Regardless of the scenario, to use this function, you need the dir_id of folder you want to share and the user_id of the account with which you want to share it.

While authenticated from the host account, the one that will issue the invitation, you can use box_ls() and box_setwd() to get the dir_id for the folder you want to share. If the host-account is the user-account, you can also use the web-portal to find the dir_id. If the host account is the service-account, you can use the Box content-portal to find the dir_id.

A user can find their user_id using the Box web-portal. As well, when you authenticate using boxr, the user_id is included in the login message. Thus, you can use box_auth_service() to find out the user_id for a given service-account.

Value

Invisible `list()` containing collaboration-information.

See Also

`box_auth()`, `box_auth_service()`

---

box_dl *Download/upload files from/to Box*

Description

`box_dl()` download a file from Box to a local directory

`box_ul()` upload a local file to a Box folder

Usage

```
box_dl(file_id, local_dir = getwd(), overwrite = FALSE,
  file_name = NULL, version_id = NULL, version_no = NULL,
  pb = options()$boxr.progress, filename)

box_ul(dir_id = box_getwd(), file, pb = options()$boxr.progress,
  description = NULL)
```

Arguments

`file_id`	`numeric` or `character`, file ID at Box.
`local_dir`	`character`, path to local directory.
`overwrite`	`logical`, indicates that newer files at origin will overwrite older files at destination.
`file_name`	`character`, if supplied, an alternate filename for the local version of the Box file.
`version_id`	`character` or `numeric`, the `version_id` of the file.
`version_no`	`numeric`, version of the file you'd like to download (starting at 1).
`pb`	`logical`, indicates to show progress bar (via `setTxtProgressBar()`).
`filename`	`character`, **deprecated**: use `file_name` instead.
`dir_id`	`numeric` or `character`, folder ID at Box.
`file`	`character`, local path to the file.
`description`	`character`, description caption for the file.

Value

`box_dl()` `character`, local path to the downloaded file.

`box_ul()` Object with S3 class `boxr_file_reference`.

Versions

box_dl() can accept one of two parameters to specify file versions: version_id or version_no.

The box.com API refers to file versions using 11 digit ids (which can be accessed via box_previous_versions()) - you can specify these using the version_id parameter.

However, this isn't terribly intuitive. As a result, box_dl() provides the version_no parameter, which accepts a whole number, and corresponds to the versions that you'll see via the web UI. For example to download the version marked 'V2' on box.com, specify version_no = 2. This works by making an internal call to box_previous_versions() to retrieve the version_id, which makes it slightly slower.

See Also

- box_fetch() and box_push() for directory-wide equivalents.
- box_delete_file() for removing uploaded files.
- box_source() for R code.
- box_save()/box_load() for remote R objects.

---

box_fetch *Download/upload directories from/to Box*

Description

box_fetch() download the contents of a Box folder to a local directory

box_push() upload the contents of a local directory to a Box folder

Files present in the origin but not the destination will be copied over.

Behaviour when a file exists in both depends on the arguments supplied.

Usage

```
box_fetch(dir_id = box_getwd(), local_dir = getwd(),
  recursive = TRUE, overwrite = FALSE, delete = FALSE)

box_push(dir_id = box_getwd(), local_dir = getwd(),
  ignore_dots = TRUE, overwrite = FALSE, delete = FALSE)
```

Arguments

dir_id	numeric or character, folder ID at Box.
local_dir	character, path to local directory.
recursive	logical, indicates to include subdirectories.
overwrite	logical, indicates that newer files at origin will overwrite older files at destination.
delete	logical, indicates to delete files that exist at destination, but not at origin.
ignore_dots	logical, indicates to ignore directories with names that begin with dots, e.g. .git and .Rproj.user.

Value

Object with S3 class boxr_dir_wide_operation_result.

Overwrite/Update

In the interests of preventing mishaps, overwrite is by default set to FALSE, which means that files which exist in the destination, but which are out of date, are not modified.

Setting overwrite to TRUE is likely to produce expected behavior for most users.

This is a conservative precaution to prevent users unexpectedly overwriting their files, and may change as a default in later releases.

However, files at Box are versioned, and most operating systems have file recovery features (e.g. 'Trash' (Ubuntu/Debian/OSX), or 'Recycle Bin' (Windows)), so unintended modification of files will be revertible for most users.

Implementation

At the time of writing, the Box API allows for only one file at a time to be uploaded/downloaded. As a result, boxr recursively scans the directory tree, uploading/downloading files in loops. Because the Box API can send, but not accept, gzipped files, downloading tends to be faster than uploading.

box_fetch()/box_push() rely on the internal function box_dir_diff() to determine how to process individual files (i.e. which to update, which to leave as is, etc.). See its help page for details.

See Also

box_dl()/box_ul() for single file operations, box_dir_diff() determines how files should be processed

---

box_fresh_auth *Re-authenticate to Box (interactive-app)*

Description

Deletes the cached token-file before trying to re-authenticate. This is often the solution to authentication problems.

Usage

```
box_fresh_auth(cache = "~/.boxr-oauth", ...)
```

Arguments

cache	A logical value or a string. TRUE means to cache using the default cache file .httr-oauth, FALSE means don't cache, and NA means to guess using some sensible heuristics. A string means use the specified path as the cache file.
...	Other arguments passed to box_auth().

See Also

box_auth() for the usual method of authentication.

---

box_ls *List files in a Box directory*

Description

Non-recursive

Usage

```
box_ls(dir_id = box_getwd(), limit = 100, max = Inf, fields = NULL)
```

Arguments

dir_id	numeric or character, folder ID at Box.
limit	integer, maximum number of entries to retrieve per query-page.
max	integer, maximum number of entries to retrieve in total.
fields	character, fields to return; the default value, NULL, will return all possible columns: modified_at, content_modified_at, name, id, type, sha1 ,size, owned_by, path_collection, description.

Value

Object with S3 class boxr_object_list.

See Also

box_fetch() and box_push() for synchronizing the contents of local and remote directories.

---

box_previous_versions *Get details of previous versions of a Box file*

Description

Box explicitly versions files; this function returns a data.frame containing information on a file's previous versions on Box. No information about the current version of the file is returned.

Usage

```
box_previous_versions(file_id)
```

Arguments

file_id	numeric or character, file ID at Box.

Details

The returned data.frame contains a variable, file_version_id, which you can use with box_dl().

Value

data.frame containing information about previous versions of the file (if available).

References

This function is a light wrapper of the box.com API versions method.

https://developers.box.com/docs/#files-view-versions-of-a-file

See Also

box_dl()

---

box_read *Read an R object from a Box file*

Description

These functions are used to download a Box file, specified by file_id, then attempt to parse its contents into memory as an R object. For example, you may wish to read a Box CSV file as a data.frame.

Usage

```
box_read(file_id, type = NULL, version_id = NULL, version_no = NULL,
  read_fun = rio::import, ...)

box_read_csv(file_id, ...)

box_read_tsv(file_id, ...)

box_read_json(file_id, ...)

box_read_excel(file_id, ...)
```

Arguments

file_id	numeric or character, file ID at Box.
type	character, MIME type used to override the content type returned by the server.
version_id	character or numeric, the version_id of the file.
version_no	numeric, version of the file you'd like to download (starting at 1).
read_fun	function, used to read (parse) the content into R; for box_read() the default function is rio::import(); the specific helpers each use a different function directly.
...	Other arguments passed to read_fun.

Details

This is a two-step process. The first is to download the contents of the file, the second is to parse those contents into an R object. The default parsing-function is rio::import().

In addition to box_read(), some specific helpers are provided:

box_read_csv() parse a remote CSV file into a data.frame. Default read-function is rio::import() with format = "csv", which uses data.table::fread() if available, and utils::read.csv() if not. Pass the argument fread = FALSE to ... to always use utils::read.csv().

box_read_tsv() parse a remote TSV file into a data.frame. Default read-function is rio::import() with format = "tsv", which uses data.table::fread() if available, and utils::read.delim() if not. Pass the argument fread = FALSE to ... to always use utils::read.delim().

box_read_json() parse a remote JSON file into a R object. Default read-function is jsonlite::fromJSON().

box_read_excel() parse a remote Microsoft Excel file into a data.frame. Default read-function is rio::import() with format = "excel", which uses readxl::read_excel() by default. Pass the argument readxl = FALSE to ... to use openxlsx::read.xlsx() instead.

Value

Object returned by function read_fun.

rio's import() and JSON files

In rio (0.5.18) there was a change in how JSON files are processed by rio::import(), a non-data.frame object stored in JSON is no longer coerced into a data.frame. The old behavior would produce unexpected results or fatal errors if the stored object was not a data.frame. The new behavior is closer to that of the underlying function jsonlite::fromJSON() and similar to the behavior for RDS files.

In keeping with the spirit of jsonlite, box_read_json() has been modified to call jsonlite::fromJSON() directly, which by-passes the old "undesirable" behavior of rio (< 0.5.18). If you are using the current CRAN release of rio (0.5.16) you should use box_read_json() to avoid these issues.

See Also

box_dl(), box_save(), box_source()

box_save *Download/upload an R workspace from/to a Box file*

Description

Similar to save(), save.image(), and load(); these functions operate on files at Box instead of on local files.

Usage

```
box_save(..., dir_id = box_getwd(), file_name = ".RData",
  description = NULL)

box_save_image(dir_id = box_getwd(), file_name = ".RData",
  description = NULL, filename)

box_load(file_id)
```

Arguments

...	Objects to be saved, quoted or unquoted; passed to save().
dir_id	numeric or character, folder ID at Box.
file_name	character, if supplied, an alternate filename for the local version of the Box file.
description	character, description caption for the file.
filename	character, **deprecated**: use file_name instead.
file_id	numeric or character, file ID at Box.

Details

box_save() Save object(s) using save(), write to Box file.

box_save_image() Save image using save.image(), write to Box file.

box_load() Read from Box file, load using load().

Value

box_save() Object with S3 class boxr_file_reference.

box_save_image() Object with S3 class boxr_file_reference.

box_load() From load(), a character vector of the names of objects created, invisibly.

See Also

save(), save.image(), load()

box_search *Search Box files*

Description

Search Box files

Usage

```
box_search(query = "", content_types = c("name", "description",
  "file_content", "comments", "tags"), type = NULL,
  file_extensions = NULL, ancestor_folder_ids = NULL,
  created_at_range = NULL, updated_at_range = NULL,
  size_range = NULL, trash = FALSE, owner_user_ids = NULL,
  max = 200)

box_search_files(query, ...)

box_search_folders(query, ...)

box_search_trash(query, ...)
```

Arguments

query	character, search term.
content_types	character, content to search; more than one can be supplied with a vector.
type	character, type of object to return; the default, NULL, returns all possible types ("file", "folder", or "weblink").
file_extensions	character, vector of strings containing the file extensions (without dots) by which to narrow your search.
ancestor_folder_ids	numeric or character, if supplied, results are limited to one or more parent (ancestor) folders.
created_at_range	POSIXct (vector, length 2), range of created-at times.
updated_at_range	POSIXct (vector, length 2), range of updated-at times.
size_range	numeric (vector, length 2), range of file sizes (bytes).
trash	logical, indicates to search only the trash folder.
owner_user_ids	numeric or character, limits search to files owned by users with these IDs.
max	numeric, upper limit on the number of search results.
...	Other arguments passed to box_search().

Details

The Box API supports a maximum of 200 results per request. If `max > 200`, then multiple requests will be sent to retrieve and combine 'paginated' results for you, behind the scenes.

See the box.com search description for details of the features of the service. Some notable details:

- Full-text searching
  - is available for many source code file types, though not including R at the time of writing.
- Boolean operators are supported
  - such as `and`, `or`, and `not` (upper or lower case).
- Phrases can be searched
  - by putting them in "quotation marks".
- Search availability
  - it takes around 10 minutes for a newly uploaded file to enter the search index.

Value

Object with S3 class `boxr_object_list`.

---

box_setwd *Get/set Box default working-directory*

Description

Similar to `getwd()` and `setwd()`, these functions get and set the folder ID of the working directory at box.com.

This folder ID is also stored in `boxr_options()`.

Usage

```
box_setwd(dir_id)

box_getwd()
```

Arguments

`dir_id`	`numeric` or `character`, folder ID at Box.

Value

`box_getwd()` `numeric`, ID for working folder at Box.

`box_setwd()` `invisible(NULL)`, called for side-effects.

See Also

`box_ls()` to list files in a Box directory, `box_fetch()`/`box_push()` to download/upload directories from/to Box

box_source *Source R code from a Box file*

Description

This function downloads a file from Box, then runs its contents, as R code, using `source()`.

Usage

```
box_source(file_id, local = globalenv(), ...)
```

Arguments

`file_id`	`numeric` or `character`, file ID at Box.
`local`	`TRUE`, `FALSE` or an environment, determining where the parsed expressions are evaluated. `FALSE` (the default) corresponds to the user's workspace (the global environment) and `TRUE` to the environment from which `source` is called.
`...`	Other arguments passed to `source()`.

Value

Object returned by `source()`, called for side-effect of modifying an environment.

See Also

`box_dl()`, `box_save()`, `box_read()`

box_write *Write an R object to a Box file*

Description

This function is used to serialize an R object and write it to a Box file. For example, you may wish to write a `data.frame` to Box as a CSV file.

Usage

```
box_write(x, file_name, dir_id = box_getwd(), description = NULL,
  write_fun = rio::export, filename, ...)
```

Arguments

x	Object to be written.
file_name	character, name of the new Box file.
dir_id	numeric or character, folder ID at Box.
description	character, description caption for the file.
write_fun	function, used to write (serialize) the content from R; default function is rio::export().
filename	character, **deprecated**: use file_name instead.
...	Other arguments passed to write_fun.

Details

This is a two-step process. The first is to serialize the contents of the R object, the second is to upload that serialization to a Box file. The default serialization-function is rio::export().

The rio::export() function currently only supports data.frame; to serialize lists, you may wish to use jsonlite::toJSON().

Please note that box_write() is used to write R objects to Box files using standard formats. To write R objects as .RData files, you can use box_save().

Value

Object with S3 class boxr_file_reference.

Index