Package: bowerbird (via r-universe)

May 27, 2026

Type Package

Title Keep a Collection of Sparkly Data Resources

Version 0.19.0

Description Tools to get and maintain a data repository from third-party data providers.

URL https://docs.ropensci.org/bowerbird, https://github.com/ropensci/bowerbird

BugReports https://github.com/ropensci/bowerbird/issues

License MIT + file LICENSE

Depends R (>= 3.3.0)

Imports archive, assertthat, aws.s3 (>= 0.3.22), CopernicusMarine (> 0.2.6.0001), curl, fs, httr, jsonlite, lubridate, magrittr, methods, openssl, R.utils, rmarkdown, rstac, rvest (>= 1.0.0), stringr, sys (>= 1.4.9000), tibble, xml2

LazyLoad yes

Encoding UTF-8

Suggests covr, knitr, testthat

VignetteBuilder knitr

Remotes cloudyr/aws.s3, pepijn-devries/CopernicusMarine

X-schema.org-applicationCategory Antarctic/Southern Ocean

X-schema.org-keywords ropensci, Antarctic, Southern Ocean, data, environmental, satellite, climate

X-schema.org-isPartOf https://ropensci.org, https://scar.org

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libabsl-dev libblosc-dev cmake libgdal-dev gdal-bin libgeos-dev make libarchive-dev libicu-dev libjpeg-dev libpng-dev libuv1-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

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

Date/Publication 2026-05-27 02:53:33 UTC

RemoteUrl https://github.com/ropensci/bowerbird

RemoteRef HEAD

RemoteSha 839c5d9b356e1882d8c7e4ddd6d6a05223157739

Contents

bb_aadc_source *Generate a bowerbird data source object for an Australian Antarctic Data Centre data set*

Description

Generate a bowerbird data source object for an Australian Antarctic Data Centre data set

Usage

```
bb_aadc_source(metadata_id, eds_id, id_is_metadata_id = FALSE, filter, ...)

bb_aadc_datasource_meta(metadata_id)
```

Arguments

metadata_id	string: the metadata ID of the data set. Browse the AADC's collection at https://data.aad.gov.au/metadata/records/ to find the relevant metadata_id
eds_id	integer: (optional) specify one or more eds_ids if the metadata record has multiple data assets attached to it and you don't want all of them
id_is_metadata_id	logical: if TRUE, use the metadata_id as the data source ID, otherwise use its DOI
filter	expression: (optional) an expression to use in a subset operation, which will be applied to the datasource metadata data frame. You can view this metadata using bb_aadc_datasource_meta()
...	: passed to bb_source

Value

A tibble containing the data source definition, as would be returned by bb_source

See Also

bb_source

Examples

```
## Not run:
  ## generate the source def for the "AADC-00009" dataset
  ##  (Antarctic Fur Seal Populations on Heard Island, Summer 1987-1988)
  src <- bb_aadc_source("AADC-00009")

  ## download it to a temporary directory
  data_dir <- tempfile()
  dir.create(data_dir)
  res <- bb_get(src, local_file_root = data_dir, verbose = TRUE)
```

```
  res$files

  ## generate the CPR data set source, which has two components
  ##  but we want to exclude the logbooks and only retrieve the actual data
  src <- bb_aadc_source("AADC-00099", filter = !grepl("logbooks", dataset_name))

## End(Not run)
```

---

bb_add *Add new data sources to a bowerbird configuration*

Description

Add new data sources to a bowerbird configuration

Usage

```
bb_add(config, source)
```

Arguments

config	bb_config: a bowerbird configuration (as returned by bb_config)
source	data.frame: one or more data source definitions, as returned by bb_source, to add to the configuration

Value

configuration object

See Also

bb_source, bb_config

Examples

```
## Not run:
  cf <- bb_config("/my/file/root") %>%
    bb_add(bb_example_sources())

## End(Not run)
```

bb_add_snapshot *Add a snapshot to a source*

Description

Experimental! Allows the results of a previous sync run to be added to a source, so that it can be used next time instead of re-indexing the remote site.

Usage

```
bb_add_snapshot(src, snapshot)
```

Arguments

src	data.frame or tibble: a single-row data source (as returned by bb_source)
snapshot	data.frame: as returned by bb_get

Value

A modified copy of src

Examples

```
## Not run:
  ## general usage: first define a source
  src <- bb_source(...)
  ## sync it once to index the remote site. This can be done as a dry run
  result <- bb_get(src, ..., dry_run = TRUE)
  ## add the snapshot to the source
  src <- bb_add_snapshot(src, result)
  ## now the sync can be run again, and the remote site does not need to be re-indexed
  result2 <- bb_get(src, ...)

## End(Not run)
```

bb_cleanup *Postprocessing: remove unwanted files*

Description

A function for removing unwanted files after downloading. This function is not intended to be called directly, but rather is specified as a postprocess option in bb_source.

Usage

```
bb_cleanup(
  pattern,
  recursive = FALSE,
  ignore_case = FALSE,
  all_files = FALSE,
  ...
)
```

Arguments

pattern	string: regular expression, passed to file.info
recursive	logical: should the cleanup recurse into subdirectories?
ignore_case	logical: should pattern matching be case-insensitive?
all_files	logical: should the cleanup include hidden files?
...	: extra parameters passed automatically by bb_sync

Details

This function can be used to remove unwanted files after a data source has been synchronized. The pattern specifies a regular expression that is passed to file.info to find matching files, which are then deleted. Note that only files in the data source's own directory (i.e. its subdirectory of the local_file_root specified in bb_config) are subject to deletion. But, beware! Some data sources may share directories, which can lead to unexpected file deletion. Be as specific as you can with the pattern parameter.

Value

a list, with components status (TRUE on success) and deleted_files (character vector of paths of files that were deleted)

See Also

bb_source, bb_config, bb_decompress

Examples

```
## Not run:
  ## remove .asc files after synchronization
  my_source <- bb_source(..., postprocess = list(list("bb_cleanup", pattern = "\\.asc$")))

## End(Not run)
```

bb_config *Initialize a bowerbird configuration*

Description

The configuration object controls the behaviour of the bowerbird synchronization process, run via `bb_sync(my_config)`. The configuration object defines the data sources that will be synchronized, where the data files from those sources will be stored, and a range of options controlling how the synchronization process is conducted. The parameters provided here are repository-wide settings, and will affect all data sources that are subsequently added to the configuration.

Usage

```
bb_config(
  local_file_root,
 wget_global_flags = list(restrict_file_names = "windows", progress = "dot:giga"),
  target_s3_args = list(),
  http_proxy = NULL,
  ftp_proxy = NULL,
  clobber = 1
)
```

Arguments

`local_file_root`	string: location of data repository on local file system
`wget_global_flags`	list: wget flags that will be applied to all data sources that call `bb_wget`. These will be appended to the data-source-specific wget flags provided via the source's method argument
`target_s3_args`	list: arguments to pass to `aws.s3` function calls. Will be used for all data sets that are uploading to s3 targets
`http_proxy`	string: URL of HTTP proxy to use e.g. 'http://your.proxy:8080' (NULL for no proxy)
`ftp_proxy`	string: URL of FTP proxy to use e.g. 'http://your.proxy:21' (NULL for no proxy)
`clobber`	numeric: 0=do not overwrite existing files, 1=overwrite if the remote file is newer than the local copy, 2=always overwrite existing files

Details

Note that the `local_file_root` directory need not actually exist when the configuration object is created, but when `bb_sync` is run, either the directory must exist or `create_root=TRUE` must be passed (i.e. `bb_sync(...,create_root=TRUE)`).

Value

configuration object

See Also

bb_source, bb_sync

Examples

```
## Not run:
  cf <- bb_config("/my/file/root") %>%
    bb_add(bb_example_sources())

  ## save to file
  saveRDS(cf,file="my_config.rds")
  ## load previously saved config
  cf <- readRDS(file="my_config.rds")

## End(Not run)
```

---

bb_copernicus_cleanup *Postprocessing: remove redundant Copernicus files*

Description

This function is not intended to be called directly, but rather is specified as a `postprocess` option in bb_source.

Usage

```
bb_copernicus_cleanup(...)
```

Arguments

`...`	: extra parameters passed automatically by `bb_sync`

Details

This function will remove files from a Copernicus collection that have been superseded by a more recently-processed version. You might see, for example, the file `nrt_global_allsat_phy_l4_20250910_20250910.nc` which is then superseded by `nrt_global_allsat_phy_l4_20250910_20250913.nc` at a later date.

Value

a list, with components `status` (TRUE on success) and `deleted_files` (character vector of paths of files that were deleted)

bb_data_source_dir *Return the local directory of each data source in a configuration*

Description

Return the local directory of each data source in a configuration. Files from each data source are stored locally in the associated directory. Note that if a data source has multiple `source_url` values, this function might return multiple directory names (depending on whether those `source_url`s map to the same directory or not).

Usage

```
bb_data_source_dir(config)
```

Arguments

`config`	bb_config: configuration as returned by `bb_config`

Value

character vector of directories

Examples

```
cf <- bb_config("/my/file/root") %>%
  bb_add(bb_example_sources())
bb_data_source_dir(cf)
```

bb_data_sources *Gets or sets a bowerbird configuration object's data sources*

Description

Gets or sets the data sources contained in a bowerbird configuration object.

Usage

```
bb_data_sources(config)

bb_data_sources(config) <- value
```

Arguments

`config`	bb_config: a bowerbird configuration (as returned by `bb_config`)
`value`	data.frame: new data sources to set (e.g. as returned by `bb_example_sources`)

Details

Note that an assignment along the lines of `bb_data_sources(cf) <- new_sources` replaces all of the sources in the configuration with the `new_sources`. If you wish to modify the existing sources then read them, modify as needed, and then rewrite the whole lot back into the configuration object.

Value

a tibble with columns as specified by `bb_source`

See Also

`bb_config`, `bb_source`, `bb_example_sources`

Examples

```
## create a configuration and add data sources
cf <- bb_config(local_file_root="/your/data/directory")
cf <- bb_add(cf,bb_example_sources())

## examine the sources contained in cf
bb_data_sources(cf)

## replace the sources with different ones
## Not run:
bb_data_sources(cf) <- new_sources

## End(Not run)
```

---

bb_decompress — *Postprocessing: decompress zip, gz, bz2, tar, Z files and optionally delete the compressed copy*

Description

Functions for decompressing files after downloading. These functions are not intended to be called directly, but rather are specified as a `postprocess` option in `bb_source`. `bb_unzip`, `bb_untar`, `bb_gunzip`, `bb_bunzip2`, and `bb_uncompress` are convenience wrappers around `bb_decompress` that specify the method.

Usage

```
bb_decompress(method, delete = FALSE, ...)

bb_unzip(...)

bb_gunzip(...)
```

```
bb_bunzip2(...)

bb_uncompress(...)

bb_inflate(...)

bb_untar(...)
```

Arguments

method	string: one of "unzip", "gunzip", "bunzip2", "decompress", "untar"
delete	logical: delete the zip files after extracting their contents?
...	: extra parameters passed automatically by bb_sync

Details

Tar files can be compressed (i.e. file extensions .tar, .tgz, .tar.gz, .tar.bz2, or .tar.xz). Support for tar files may depend on your platform (see untar).

If the data source delivers compressed files, you will most likely want to decompress them after downloading. These functions will do this for you. By default, these do not delete the compressed files after decompressing. The reason for this is so that on the next synchronization run, the local (compressed) copy can be compared to the remote compressed copy, and the download can be skipped if nothing has changed. Deleting local compressed files will save space on your file system, but may result in every file being re-downloaded on every synchronization run.

Value

list with components status (TRUE on success), files (character vector of paths to extracted files), and deleted_files (character vector of paths of files that were deleted)

See Also

bb_source, bb_config, bb_cleanup

Examples

```
## Not run:
  ## decompress .zip files after synchronization but keep zip files intact
  my_source <- bb_source(..., postprocess = list("bb_unzip"))

  ## decompress .zip files after synchronization and delete zip files
  my_source <- bb_source(..., postprocess = list(list("bb_unzip", delete = TRUE)))

## End(Not run)
```

`bb_example_sources` *Example bowerbird data sources*

Description

These example sources are useful as data sources in their own right, but are primarily provided as demonstrations of how to define data sources. See also `vignette("bowerbird")` for further examples and discussion.

Usage

```
bb_example_sources(sources)
```

Arguments

`sources`	character: names or identifiers of one or more sources to return. See Details for the list of example sources and a brief explanation of each

Details

Example data sources:

- "NOAA OI SST V2" - a straightforward data source that requires a simple one-level recursive download
- "Australian Election 2016 House of Representatives data" - an example of a recursive download that uses additional criteria to restrict what is downloaded
- "CMEMS global gridded SSH reprocessed (1993-ongoing)" - a data source that requires a username and password
- "Oceandata SeaWiFS Level-3 mapped monthly 9km chl-a" - an example data source that uses the `bb_handler_oceandata` method
- "Sea Ice Trends and Climatologies from SMMR and SSM/I-SSMIS, Version 3" - an example data source that uses the `bb_handler_earthdata` method
- "Bathymetry of Lake Superior" - another example that passes extra flags to the `bb_handler_rget` call in order to restrict what is downloaded

Value

a tibble with columns as specified by `bb_source`

References

See the `doc_url` and `citation` field in each row of the returned tibble for references associated with these particular data sources

See Also

`bb_config`, `bb_handler_rget`, `bb_handler_oceandata`, `bb_handler_earthdata`, `bb_source_us_buildings`

Examples

```
## define a configuration and add the 2016 election data source to it
cf <- bb_config("/my/file/root") %>% bb_add(
   bb_example_sources("Australian Election 2016 House of Representatives data"))

## Not run:
  ## synchronize (download) the data
  bb_sync(cf)

## End(Not run)
```

`bb_find_wget` *Find the wget executable*

Description

This function will return the path to the wget executable if it can be found on the local system, and optionally install it if it is not found. Installation (if required) currently only works on Windows platforms. The wget.exe executable will be downloaded from https://eternallybored.org/misc/wget/ installed into your appdata directory (typically something like C:/Users/username/AppData/Roaming/)

Usage

```
bb_find_wget(install = FALSE, error = TRUE)
```

Arguments

`install`	logical: attempt to install the executable if it is not found? (Windows only)
`error`	logical: if wget is not found, raise an error. If `FALSE`, do not raise an error but return NULL

Value

the path to the wget executable, or (if error is `FALSE`) NULL if it was not found

References

https://eternallybored.org/misc/wget/

See Also

`bb_install_wget`

Examples

```
## Not run:
  wget_path <- bb_find_wget()
  wget_path <- bb_find_wget(install=TRUE) ## install (on windows) if needed

## End(Not run)
```

`bb_fingerprint` *Fingerprint the files associated with a data source*

Description

The `bb_fingerprint` function, given a data repository configuration, will return the timestamp of download and hashes of all files associated with its data sources. This is intended as a general helper for tracking data provenance: for all of these files, we have information on where they came from (the data source ID), when they were downloaded, and a hash so that later versions of those files can be compared to detect changes. See also `vignette("data_provenance")`.

Usage

```
bb_fingerprint(config, hash = "sha1")
```

Arguments

`config`	bb_config: configuration as returned by `bb_config`
`hash`	string: algorithm to use to calculate file hashes: "md5", "sha1", or "none". Note that file hashing can be slow for large file collections

Value

a tibble with columns:

- filename - the full path and filename of the file
- data_source_id - the identifier of the associated data source (as per the `id` argument to `bb_source`)
- size - the file size
- last_modified - last modified date of the file
- hash - the hash of the file (unless `hash="none"` was specified)

See Also

`vignette("data_provenance")`

Examples

```
## Not run:
  cf <- bb_config("/my/file/root") %>%
    bb_add(bb_example_sources())
  bb_fingerprint(cf)

## End(Not run)
```

---

bb_get *Convenience function to define and synchronize a bowerbird data collection*

Description

This is a convenience function that provides a shorthand method for synchronizing a small number of data sources. The call `bb_get(...)` is roughly equivalent to `bb_sync(bb_add(bb_config(...), ...), ...)` (don't take the dots literally here, they are just indicating argument placeholders).

Usage

```
bb_get(
  data_sources,
  local_file_root,
  clobber = 1,
  http_proxy = NULL,
  ftp_proxy = NULL,
  create_root = FALSE,
  verbose = FALSE,
  confirm_downloads_larger_than = 0.1,
  dry_run = FALSE,
  ...
)
```

Arguments

`data_sources`	tibble: one or more data sources to download, as returned by e.g. `bb_example_sources`
`local_file_root`	string: location of data repository on local file system
`clobber`	numeric: 0=do not overwrite existing files, 1=overwrite if the remote file is newer than the local copy, 2=always overwrite existing files
`http_proxy`	string: URL of HTTP proxy to use e.g. 'http://your.proxy:8080' (NULL for no proxy)
`ftp_proxy`	string: URL of FTP proxy to use e.g. 'http://your.proxy:21' (NULL for no proxy)

create_root	logical: should the data root directory be created if it does not exist? If this is FALSE (default) and the data root directory does not exist, an error will be generated
verbose	logical: if TRUE, provide additional progress output
confirm_downloads_larger_than	numeric or NULL: if non-negative, bb_sync will ask the user for confirmation to download any data source of size greater than this number (in GB). A value of zero will trigger confirmation on every data source. A negative or NULL value will not prompt for confirmation. Note that this only applies when R is being used interactively. The expected download size is taken from the collection_size parameter of the data source, and so its accuracy is dependent on the accuracy of the data source definition
dry_run	logical: if TRUE, bb_sync will do a dry run of the synchronization process without actually downloading files
...	: additional parameters passed through to bb_config or bb_sync

Details

Note that the local_file_root directory must exist or create_root=TRUE must be passed.

Value

a tibble, as for bb_sync

See Also

bb_config, bb_example_sources, bb_source, bb_sync

Examples

```
## Not run:
 my_source <- bb_example_sources("Australian Election 2016 House of Representatives data")
 status <- bb_get(local_file_root = tempdir(), data_sources = my_source, verbose = TRUE)

 ## the files that have been downloaded:
 status$files[[1]]

 ## Define a new source: Geelong bicycle paths from data.gov.au
 my_source <- bb_source(
   name = "Bike Paths - Greater Geelong",
   id = "http://data.gov.au/dataset/7af9cf59-a4ea-47b2-8652-5e5eeed19611",
   doc_url = "https://data.gov.au/dataset/geelong-bike-paths",
   citation = "See https://data.gov.au/dataset/geelong-bike-paths",
   source_url = "https://data.gov.au/dataset/7af9cf59-a4ea-47b2-8652-5e5eeed19611",
   license = "CC-BY",
   method = list("bb_handler_rget", accept_download = "\\.zip$", level = 1),
   postprocess = list("bb_unzip"))

 ## get the data
 status <- bb_get(data_sources = my_source, local_file_root = tempdir(), verbose = TRUE)
```

```
  ## find the .shp file amongst the files, and plot it
  shpfile <- status$files[[1]]$file[grepl("shp$", status$files[[1]]$file)]
  library(sf)
  bx <- read_st(shpfile)
  plot(bx)

## End(Not run)
```

---

bb_handler_aws_s3 *Handler for public AWS S3 data sources*

Description

This is a handler function to be used with AWS S3 data providers. This function is not intended to be called directly, but rather is specified as a `method` option in `bb_source`. Note that this currently only works with public data sources that are accessible without an S3 key. The method arguments accepted by `bb_handler_aws_s3` are currently:

- "bucket" string: name of the bucket (defaults to "")
- "base_url" string: as for `s3HTTP`
- "region" string: as for `s3HTTP`
- "use_https" logical: as for `s3HTTP`
- "prefix" string: as for `get_bucket`; only keys in the bucket that begin with the specified prefix will be processed
- and other parameters passed to the `bb_rget` function, including "accept_download", "accept_download_extra", "reject_download"

Note that the "prefix", "accept_download", "accept_download_extra", "reject_download" parameters can be used to restrict which files are downloaded from the bucket.

Usage

```
bb_handler_aws_s3(...)
```

Arguments

...	: parameters, see Description

Value

A tibble with columns `ok`, `files`, `message`

Examples

```
## Not run:
  ## an example AWS S3 data source
  src <- bb_source(
           name = "SILO climate data",
           id = "silo-open-data",
           description = "Australian climate data from 1889 to yesterday.
                       This source includes a single example monthly rainfall data file.
                         Adjust the 'accept_download' parameter to change this.",
           doc_url = "https://www.longpaddock.qld.gov.au/silo/gridded-data/",
           citation = "SILO datasets are constructed by the Queensland Government using
                     observational data provided by the Australian Bureau of Meteorology
                    and are available under the Creative Commons Attribution 4.0 license",
           license = "CC-BY 4.0",
           method = list("bb_handler_aws_s3", region = "silo-open-data.s3",
                    base_url = "amazonaws.com", prefix = "Official/annual/monthly_rain/",
                         accept_download = "2005\\.monthly_rain\\.nc$"),
          comment = "The unusual specification of region and base_url is a workaround for
                     an aws.s3 issue, see https://github.com/cloudyr/aws.s3/issues/318",
           postprocess = NULL,
           collection_size = 0.02,
           data_group = "Climate")
   temp_root <- tempdir()
   status <- bb_get(src, local_file_root = temp_root, verbose = TRUE)

## End(Not run)
```

---

bb_handler_copernicus *Handler for Copernicus Marine datasets*

Description

This is a handler function to be used with data sets from Copernicus Marine. This function is not intended to be called directly, but rather is specified as a `method` option in `bb_source`.

Usage

```
bb_handler_copernicus(product, layer, prefix = NULL, pattern = "", ctype, ...)
```

Arguments

`product`	string: the desired Copernicus marine product. See `cms_products_list`
`layer`	string: (required if the product contains more than one layer) the layer within a product. See `cms_product_details`
`prefix`	string: (optional) as for `cms_list_native_files`
`pattern`	string: (optional) as for `cms_list_native_files`
`ctype`	legacy parameter, ignored
`...`	: parameters passed to `bb_rget`

Details

Note that users will need a Copernicus login.

Value

TRUE on success

References

https://help.marine.copernicus.eu/en/collections/4060068-copernicus-marine-toolbox

---

bb_handler_earthdata *Handler for data sets from Earthdata providers*

Description

This is a handler function to be used with data sets from NASA's Earthdata system. This function is not intended to be called directly, but rather is specified as a `method` option in `bb_source`.

Usage

```
bb_handler_earthdata(...)
```

Arguments

...	: parameters passed to `bb_rget`

Details

This function uses `bb_rget`, and so data sources using this function will need to provide appropriate `bb_rget` parameters. Note that curl v5.2.1 introduced a breaking change to the default value of the 'unrestricted_auth' option: see <https://github.com/jeroen/curl/issues/260>. Your Earthdata source definition might require 'allow_unrestricted_auth = TRUE' as part of the method parameters.

Value

TRUE on success

References

https://wiki.earthdata.nasa.gov/display/EL/How+To+Register+With+Earthdata+Login

Examples

```
## Not run:

## note that the full version of this data source is provided as part of bb_example_data_sources()

my_source <- bb_source(
  name = "Sea Ice Trends and Climatologies from SMMR and SSM/I-SSMIS, Version 3",
  id = "10.5067/IJ0T7HFHB9Y6",
 description = "NSIDC provides this data set ... [truncated; see bb_example_data_sources()]",
  doc_url = "https://nsidc.org/data/NSIDC-0192/versions/3",
  citation = "Stroeve J, Meier WN (2018) ... [truncated; see bb_example_data_sources()]",
 source_url = "https://daacdata.apps.nsidc.org/pub/DATASETS/nsidc0192_seaice_trends_climo_v3/",
  license = "Please cite, see http://nsidc.org/about/use_copyright.html",
  authentication_note = "Requires Earthdata login, see https://urs.earthdata.nasa.gov/.
    Note that you will also need to authorize the application 'nsidc-daacdata'
    (see 'My Applications' at https://urs.earthdata.nasa.gov/profile)",
  method = list("bb_handler_earthdata", level = 4, relative = TRUE,
                accept_download = "\\.(s|n|png|txt)$", allow_unrestricted_auth = TRUE),
  user = "your_earthdata_username",
  password = "your_earthdata_password",
  collection_size = 0.02)

## End(Not run)
```

---

bb_handler_earthdata_stac

Handler for data sets from Earthdata providers, using the STAC catalogue to do the initial search

---

Description

This is a handler function to be used with data sets from NASA's Earthdata system. This function is not intended to be called directly, but rather is specified as a `method` option in `bb_source`. This function finds the items to download via the STAC catalog, and then uses `bb_rget` to download them. Data sources using this function can provide appropriate `bb_rget` parameters if required. The method arguments accepted by `bb_handler_aws_s3` are currently:

- "stac_id" string: the STAC identifier
- "collection_id" string: the collection identifier

Usage

```
bb_handler_earthdata_stac(...)
```

Arguments

...	: parameters, see Description

Value

A tibble with columns ok, files, message

References

https://wiki.earthdata.nasa.gov/display/EL/How+To+Register+With+Earthdata+Login

Examples

```
## Not run:

my_source <- bb_source(
  name = "Nimbus Ice Edge Points from Nimbus Visible Imagery",
  id = "10.5067/NIMBUS/NmIcEdg2",
  description = "This data set (NmIcEdg2) ... [truncated; see sources_seaice()]",
  doc_url = "http://nsidc.org/data/nmicedg2/",
  citation = "Gallaher D and Campbell G ... [truncated; see sources_seaice()]",
  license = "Please cite, see http://nsidc.org/about/use_copyright.html",
  authentication_note = "Requires Earthdata login, see https://urs.earthdata.nasa.gov/.
    Note that you will also need to authorize the application 'nsidc-daacdata'
    (see 'My Applications' at https://urs.earthdata.nasa.gov/profile)",
  method = list("bb_handler_earthdata_stac", stac_id = "NSIDC_CPRD",
    collection_id = "NmIcEdg2_1"),
  user = "your_earthdata_username",
  password = "your_earthdata_password",
  collection_size = 0.02)

## End(Not run)
```

---

bb_handler_oceandata *Handler for Oceandata data sets*

Description

This is a handler function to be used with data sets from NASA's Oceandata system. This function is not intended to be called directly, but rather is specified as a method option in bb_source.

Usage

```
bb_handler_oceandata(search, dtype, sensor, ...)
```

Arguments

search	string: (required) the search string to pass to the oceancolor file searcher (https://oceandata.sci.gsfc.nasa.g
dtype	string: (optional) the data type (e.g. "L3m") to pass to the oceancolor file searcher. Valid options at the time of writing are aquarius, seawifs, aqua, terra, meris, octs, czcs, hico, viirs (for snpp), viirsj1, s3olci (for sentinel-3a), s3bolci (see https://oceancolor.gsfc.nasa.gov/data/download_methods/)

sensor	string: (optional) the sensor (e.g. "seawifs") to pass to the oceancolor file searcher. Valid options at the time of writing are L0, L1, L2, L3b (for binned data), L3m (for mapped data), MET (for ancillary data), misc (for sundry products)
...	: extra parameters passed automatically by bb_sync

Details

Note that users will need an Earthdata login, see https://urs.earthdata.nasa.gov/. Users will also need to authorize the application 'OB.DAAC Data Access' (see 'My Applications' at https://urs.earthdata.nasa.gov/profile)

Oceandata uses standardized file naming conventions (see https://oceancolor.gsfc.nasa.gov/docs/format/), so once you know which products you want you can construct a suitable file name pattern to search for. For example, "S*L3m_MO_CHL_chlor_a_9km.nc" would match monthly level-3 mapped chlorophyll data from the SeaWiFS satellite at 9km resolution, in netcdf format. This pattern is passed as the search argument. Note that the bb_handler_oceandata does not take need 'source_url' to be specified in the bb_source call.

Value

TRUE on success

References

https://oceandata.sci.gsfc.nasa.gov/

Examples

```
my_source <- bb_source(
  name="Oceandata SeaWiFS Level-3 mapped monthly 9km chl-a",
  id="SeaWiFS_L3m_MO_CHL_chlor_a_9km",
  description="Monthly remote-sensing chlorophyll-a from the SeaWiFS satellite at
    9km spatial resolution",
  doc_url="https://oceancolor.gsfc.nasa.gov/",
  citation="See https://oceancolor.gsfc.nasa.gov/citations",
  license="Please cite",
  method=list("bb_handler_oceandata",search="S*L3m_MO_CHL_chlor_a_9km.nc"),
  postprocess=NULL,
  collection_size=7.2,
  data_group="Ocean colour")
```

---

bb_handler_rget *Mirror an external data source using bowerbird's bb_rget utility*

Description

This is a general handler function that is suitable for a range of data sets. This function is not intended to be called directly, but rather is specified as a method option in bb_source.

Usage

```
bb_handler_rget(...)
```

Arguments

...	: parameters passed to bb_rget

Details

This handler function makes calls to the bb_rget function. Arguments provided to bb_handler_rget are passed through to bb_rget.

Value

A tibble with columns ok, files, message

See Also

bb_rget, bb_source, bb_sync

Examples

```
my_source <- bb_source(
   name = "Australian Election 2016 House of Representatives data",
   id = "aus-election-house-2016",
   description = "House of Representatives results from the 2016 Australian election.",
   doc_url = "http://results.aec.gov.au/",
  citation = "Copyright Commonwealth of Australia 2017. As far as practicable, material for
              which the copyright is owned by a third party will be clearly labelled. The
               AEC has made all reasonable efforts to ensure that this material has been
              reproduced on this website with the full consent of the copyright owners.",
  source_url = "http://results.aec.gov.au/20499/Website/HouseDownloadsMenu-20499-Csv.htm",
   license = "CC-BY",
   method = list("bb_handler_rget", level = 1, accept_download = "csv$"),
   collection_size = 0.01)

my_data_dir <- tempdir()
cf <- bb_config(my_data_dir)
cf <- bb_add(cf, my_source)

## Not run:
bb_sync(cf, verbose = TRUE)

## End(Not run)
```

bb_handler_thredds *Mirror an external thredds data source using bowerbird's bb_rget utility*

Description

This function is not intended to be called directly, but rather is specified as a `method` option in `bb_source`.

Usage

```
bb_handler_thredds(...)
```

Arguments

`...`	: parameters passed to `bb_rget`

Value

TRUE on success

See Also

`bb_rget`, `bb_source`, `bb_sync`

Examples

```
my_source <- bb_source(
    name = "OSI SAF Global Low Resolution Sea Ice Drift",
    id = "10.15770/EUM_SAF_OSI_NRT_2007",
    description = "Example dataset.",
    doc_url = "https://osi-saf.eumetsat.int/products/osi-405-c",
    ## just the 2009 subset for demo purposes
    source_url =
   "https://thredds.met.no/thredds/catalog/osisaf/met.no/ice/drift_lr/merged/2009/catalog.html",
    citation = "See https://doi.org/10.15770/EUM_SAF_OSI_NRT_2007",
    license = "Please cite",
    method = list("bb_handler_thredds", level = 2),
    access_function = "",
    data_group = "Sea ice")

## Not run:
  result <- bb_get(my_source, local_file_root = tempdir(), verbose = TRUE)

## End(Not run)
```

bb_handler_wget *Mirror an external data source using the wget utility*

Description

This is a general handler function that is suitable for a range of data sets. This function is not intended to be called directly, but rather is specified as a `method` option in `bb_source`.

Usage

```
bb_handler_wget(...)
```

Arguments

`...`	: parameters passed to `bb_wget`

Details

This handler function makes calls to the `wget` utility via the `bb_wget` function. Arguments provided to `bb_handler_wget` are passed through to `bb_wget`.

Value

TRUE on success

See Also

`bb_wget`, `bb_source`

Examples

```
my_source <- bb_source(
   id="gshhg_coastline",
   name="GSHHG coastline data",
  description="A Global Self-consistent, Hierarchical, High-resolution Geography Database",
   doc_url= "http://www.soest.hawaii.edu/pwessel/gshhg",
   citation="Wessel, P., and W. H. F. Smith, A Global Self-consistent, Hierarchical,
     High-resolution Shoreline Database, J. Geophys. Res., 101, 8741-8743, 1996",
   source_url="ftp://ftp.soest.hawaii.edu/gshhg/*",
   license="LGPL",
   method=list("bb_handler_wget",recursive=TRUE,level=1,accept="*bin*.zip,README.TXT"),
   postprocess=list("bb_unzip"),
   collection_size=0.6)
```

`bb_install_wget` *Install wget*

Description

This is a helper function to install wget. Currently it only works on Windows platforms. The wget.exe executable will be downloaded from https://eternallybored.org/misc/wget/ and saved to either a temporary directory or your user appdata directory (see the `use_appdata_dir` parameter).

Usage

```
bb_install_wget(force = FALSE, use_appdata_dir = FALSE)
```

Arguments

`force`	logical: force reinstallation if wget already exists
`use_appdata_dir`	logical: by default, `bb_install_wget` will install wget into a temporary directory, which does not persist between R sessions. If you want a persistent installation, specify `use_appdata_dir=TRUE` to install wget into your appdata directory (on Windows, typically something like C:/Users/username/AppData/Roaming/)

Value

the path to the installed executable

References

https://eternallybored.org/misc/wget/

See Also

`bb_find_wget`

Examples

```
## Not run:
  bb_install_wget()

  ## confirm that it worked:
  bb_wget("help")

## End(Not run)
```

bb_modify_source *Modify a data source*

Description

This is a helper function designed to make it easier to modify an already-defined data source. Generally, parameters passed here will replace existing entries in src if they exist, or will be added if not. The method and postprocess parameters are slightly different: see Details, below.

Usage

```
bb_modify_source(src, ...)
```

Arguments

src	data.frame or tibble: a single-row data source (as returned by bb_source)
...	: parameters as for bb_source

Details

With the exception of the method and postprocess parameters, any parameter provided here will entirely replace its equivalent in the src object. Pass a new value of NULL to remove an existing parameter.

The method and postprocess parameters are lists, and modification for these takes place at the list-element level: any element of the new list will replace its equivalent element in the list in src. If the src list does not contain that element, it will be added. To illustrate, say that we have created a data source with:

```
src <- bb_source(method=list("bb_handler_rget", parm1 = value1, parm2 = value2), ...)
```

Calling

```
bb_modify_source(src, method = list(parm1 = newvalue1))
```

will result in a new method value of `list("bb_handler_rget", parm1 = newvalue1, parm2 = value2)`

Modifying postprocess elements is similar. Note that it is not currently possible to entirely remove a postprocess component using this function. If you need to do so, you'll need to do it manually.

Value

as for bb_source: a tibble with columns as per the bb_source function arguments (excluding warn_empty_auth)

See Also

bb_source

Examples

```
## this pre-defined source requires a username and password
src <- bb_example_sources(
          "Sea Ice Trends and Climatologies from SMMR and SSM/I-SSMIS, Version 3")

## add username and password
src <- bb_modify_source(src,user="myusername",password="mypassword")

## or using the pipe operator
src <- bb_example_sources(
          "Sea Ice Trends and Climatologies from SMMR and SSM/I-SSMIS, Version 3") %>%
  bb_modify_source(user="myusername",password="mypassword")

## remove the existing "data_group" component
src %>% bb_modify_source(data_group=NULL)

## change just the 'level' setting of an existing method definition
src %>% bb_modify_source(method=list(level=3))

## remove the 'level' component of an existing method definition
src %>% bb_modify_source(method=list(level=NULL))
```

---

bb_oceandata_cleanup *Postprocessing: remove redundant NRT oceandata files*

Description

This function is not intended to be called directly, but rather is specified as a `postprocess` option in `bb_source`.

Usage

```
bb_oceandata_cleanup(...)
```

Arguments

`...`	: extra parameters passed automatically by `bb_sync`

Details

This function will remove near-real-time (NRT) files from an oceandata collection that have been superseded by their non-NRT versions.

Value

a list, with components `status` (TRUE on success) and `deleted_files` (character vector of paths of files that were deleted)

bb_rget *A recursive download utility*

Description

This function provides similar functionality to the the command-line `wget` utility.

Usage

```
bb_rget(
  url,
  level = 0,
  wait = 0,
  accept_follow = c("(/|\\.html?)$"),
  reject_follow = character(),
  accept_download = bb_rget_default_downloads(),
  accept_download_extra = character(),
  reject_download = character(),
  user,
  password,
  clobber = 1,
  no_parent = TRUE,
  no_parent_download = no_parent,
  no_check_certificate = FALSE,
  relative = FALSE,
  remote_time = TRUE,
  verbose = FALSE,
  show_progress = verbose,
  debug = FALSE,
  dry_run = FALSE,
  stop_on_download_error = FALSE,
  retries = 0,
  force_local_filename,
  use_url_directory = TRUE,
  no_host = FALSE,
  cut_dirs = 0L,
  link_css = "a",
  link_href = "href",
  curl_opts,
  target_s3_args,
  download_link_rewrite
)

bb_rget_default_downloads()
```

Arguments

`url`	string: the URL to retrieve

`level` integer >=0: recursively download to this maximum depth level. Specify 0 for no recursion

`wait` numeric >=0: wait this number of seconds between successive retrievals. This option may help with servers that block users making too many requests in a short period of time

`accept_follow` character: character vector with one or more entries. Each entry specifies a regular expression that is applied to the complete URL. URLs matching all entries will be followed during the spidering process. Note that the first URL (provided via the `url` parameter) will always be visited, unless it matches the download criteria

`reject_follow` character: as for `accept_follow`, but specifying URL regular expressions to reject

`accept_download`
character or function: if a character vector with one or more entries, each entry specifies a regular expression that is applied to the complete URL. URLs that match all entries will be accepted for download. By default the `accept_download` parameter is that returned by `bb_rget_default_downloads`: use `bb_rget_default_downloads()` to see what that is. Otherwise `accept_download` can be provided as a function that accepts a character vector of URLs and returns a logical vector of the same length. Elements of the returned vector should be `TRUE` (the associated URL is a download link and should be downloaded), `FALSE` (a download link that should not be downloaded), or `NA` (not a download link). Note that if `accept_download` is a function, then `accept_download_extra` and `reject_download` are ignored

`accept_download_extra`
character: character vector with one or more entries. If provided, URLs will be accepted for download if they match all entries in `accept_download` OR all entries in `accept_download_extra`. This is a convenient method to add one or more extra download types, without needing to re-specify the defaults in `accept_download`. Ignored if `accept_download` is a function

`reject_download`
character: as for `accept_download`, but specifying URL regular expressions to reject. Ignored if `accept_download` is a function

`user` string: username used to authenticate to the remote server

`password` string: password used to authenticate to the remote server

`clobber` numeric: 0=do not overwrite existing files, 1=overwrite if the remote file is newer than the local copy, 2=always overwrite existing files

`no_parent` logical: if `TRUE`, do not ever ascend to the parent directory when retrieving recursively. This is `TRUE` by default, bacause it guarantees that only the files below a certain hierarchy will be downloaded. Note that this check only applies to links on the same host as the starting `url`. If that URL links to files on another host, those links will be followed (unless `relative = TRUE`)

`no_parent_download`
logical: similar to `no_parent`, but applies only to download links. A typical use case is to set `no_parent` to `TRUE` and `no_parent_download` to `FALSE`, in which case the spidering process (following links to find downloadable files) will not ascend to the parent directory, but files can be downloaded from a directory that is not within the parent

`no_check_certificate`
: logical: if `TRUE`, don't check the server certificate against the available certificate authorities. Also don't require the URL host name to match the common name presented by the certificate. This option might be useful if trying to download files from a server with an expired certificate, but it is clearly a security risk and so should be used with caution

`relative`
: logical: if `TRUE`, only follow relative links. This can be useful for restricting what is downloaded in recursive mode

`remote_time`
: logical: if `TRUE`, attempt to set the local file's time to that of the remote file

`verbose`
: logical: print trace output?

`show_progress`
: logical: if `TRUE`, show download progress

`debug`
: logical: if `TRUE`, will print additional debugging information. If bb_rget is not behaving as expected, try setting this to `TRUE`

`dry_run`
: logical: if `TRUE`, spider the remote site and work out which files would be downloaded, but don't download them

`stop_on_download_error`
: logical: if `TRUE`, the download process will stop if any file download fails. If `FALSE`, the process will issue a warning and continue to the next file to download

`retries`
: integer: number of times to retry a request if it fails with a transient error (similar to curl, a transient error means a timeout, an FTP 4xx response code, or an HTTP 5xx response code

`force_local_filename`
: character: if provided, then each `url` will be treated as a single URL (no recursion will be conducted). It will be downloaded to a file with name given `force_local_filename`, in a local directory determined by the `url`. `force_local_filename` should be a character vector of the same length as the `url` vector

`use_url_directory`
: logical: if `TRUE`, files will be saved into a local directory that follows the URL structure (e.g. files from `http://some.where/place` will be saved into directory `some.where/place`). If `FALSE`, files will be saved into the current directory

`no_host`
: logical: if `use_url_directory = TRUE`, specifying `no_host = TRUE` will remove the host name from the directory (e.g. files from files from `http://some.where/place` will be saved into directory `place`)

`cut_dirs`
: integer: if `use_url_directory = TRUE`, specifying `cut_dirs` will remove this many directory levels from the path of the local directory where files will be saved (e.g. if `cut_dirs = 2`, files from `http://some.where/place/baa/haa` will be saved into directory `some.where/haa`. if `cut_dirs = 1` and `no_host = TRUE`, files from `http://some.where/place/baa/haa` will be saved into directory `baa/haa`)

`link_css`
: string: css selector that identifies links (passed as the `css` parameter to `html_elements`). Note that link elements must have an `link_href` attribute

`link_href`
: string: the attribute of a link that gives the destination (i.e. the URL to follow)

`curl_opts`
: named list: options to use with `curl` downloads, passed to the `.list` parameter of `curl::new_handle`

target_s3_args list: named list or arguments to provide to get_bucket_df and put_object. Files will be uploaded into that bucket instead of the local filesystem

download_link_rewrite
function: if supplied, this function will be applied to each download link after it is scraped from the source page and expanded to an absolute URL but before it is checked against accept_download. This function should take three parameters:

- x is a character vector of download link URLs
- url is the starting URL (from which those download URLs were scraped)
- content is the content of the starting URL, as an XML document as returned by [xml2::read_html()]

and it should return a copy of x, with entries appropriately modified.

Value

a list with components 'ok' (TRUE/FALSE), 'files', and 'message' (error or other messages)

---

bb_settings *Gets or sets a bowerbird configuration object's settings*

Description

Gets or sets a bowerbird configuration object's settings. These are repository-wide settings that are applied to all data sources added to the configuration. Use this function to alter the settings of a configuration previously created using bb_config.

Usage

```
bb_settings(config)

bb_settings(config) <- value
```

Arguments

config	bb_config: a bowerbird configuration (as returned by bb_config)
value	list: new values to set

Details

Note that an assignment along the lines of bb_settings(cf) <- new_settings replaces all of the settings in the configuration with the new_settings. The most common usage pattern is to read the existing settings, modify them as needed, and then rewrite the whole lot back into the configuration object (as per the examples here).

Value

named list

See Also

bb_config

Examples

```
cf <- bb_config(local_file_root="/your/data/directory")

## see current settings
bb_settings(cf)

## add an http proxy
sets <- bb_settings(cf)
sets$http_proxy <- "http://my.proxy"
bb_settings(cf) <- sets

## change the current local_file_root setting
sets <- bb_settings(cf)
sets$local_file_root <- "/new/location"
bb_settings(cf) <- sets
```

---

bb_source *Define a data source*

Description

This function is used to define a data source, which can then be added to a bowerbird data repository configuration. Passing the configuration object to bb_sync will trigger a download of all of the data sources in that configuration.

Usage

```
bb_source(
  id,
  name,
  description = NA_character_,
  doc_url,
  source_url,
  citation,
  license,
  comment = NA_character_,
  method,
  postprocess,
  authentication_note = NA_character_,
  user = NA_character_,
  password = NA_character_,
  access_function = NA_character_,
  data_group = NA_character_,
```

```
  collection_size = NA,
  warn_empty_auth = TRUE
)
```

Arguments

id	string: (required) a unique identifier of the data source. If the data source has a DOI, use that. Otherwise, if the original data provider has an identifier for this dataset, that is probably a good choice here (include the data version number if there is one). The ID should be something that changes when the data set changes (is updated). A DOI is ideal for this
name	string: (required) a unique name for the data source. This should be a human-readable but still concise name
description	string: a plain-language description of the data source, provided so that users can get an idea of what the data source contains (for full details they can consult the doc_url link)
doc_url	string: (required) URL to the metadata record or other documentation of the data source
source_url	character vector: one or more source URLs. Required for bb_handler_rget, although some method functions might not require one
citation	string: (required) details of the citation for the data source
license	string: (required) description of the license. For standard licenses (e.g. creative commons) include the license descriptor ("CC-BY", etc)
comment	string: comments about the data source. If only part of the original data collection is mirrored, mention that here
method	list (required): a list object that defines the function used to synchronize this data source. The first element of the list is the function name (as a string or function). Additional list elements can be used to specify additional parameters to pass to that function. Note that bb_sync automatically passes the data repository configuration object as the first parameter to the method handler function. If the handler function uses bb_rget (e.g. bb_handler_rget), these extra parameters are passed through to the bb_rget function
postprocess	list: each element of postprocess defines a postprocessing step to be run after the main synchronization has happened. Each element of this list can be a function or string function name, or a list in the style of list(fun,arg1=val1,arg2=val2) where fun is the function to be called and arg1 and arg2 are additional parameters to pass to that function
authentication_note	string: if authentication is required in order to access this data source, make a note of the process (include a URL to the registration page, if possible)
user	string: username, if required
password	string: password, if required
access_function	string: can be used to suggest to users an appropriate function to read these data files. Provide the name of an R function or even a code snippet

`data_group` string: the name of the group to which this data source belongs. Useful for arranging sources in terms of thematic areas

`collection_size` numeric: approximate disk space (in GB) used by the data collection, if known. If the data are supplied as compressed files, this size should reflect the disk space used after decompression. If the data_source definition contains multiple source_url entries, this size should reflect the overall disk space used by all combined

`warn_empty_auth` logical: if `TRUE`, issue a warning if the data source requires authentication (authentication_note is not NA) but user and password have not been provided. Set this to `FALSE` if you are defining a data source for others to use with their own credentials: they will typically call your data source constructor and then modify the `user` and `password` components

Details

The `method` parameter defines the handler function used to synchronize this data source, and any extra parameters that need to be passed to it.

Parameters marked as "required" are the minimal set needed to define a data source. Other parameters are either not relevant to all data sources (e.g. `postprocess`, `user`, `password`) or provide metadata to users that is not strictly necessary to allow the data source to be synchronized (e.g. `description`, `access_function`, `data_group`). Note that three of the "required" parameters (namely `citation`, `license`, and `doc_url`) are not strictly needed by the synchronization code, but are treated as "required" because of their fundamental importance to reproducible science.

See `vignette("bowerbird")` for more examples and discussion of defining data sources.

Value

a tibble with columns as per the function arguments (excluding `warn_empty_auth`)

See Also

bb_config, bb_sync, `vignette("bowerbird")`

Examples

```
## a minimal definition for the GSHHG coastline data set:

my_source <- bb_source(
   id = "gshhg_coastline",
   name = "GSHHG coastline data",
   doc_url = "http://www.soest.hawaii.edu/pwessel/gshhg",
   citation = "Wessel, P., and W. H. F. Smith, A Global Self-consistent, Hierarchical,
     High-resolution Shoreline Database, J. Geophys. Res., 101, 8741-8743, 1996",
   source_url = "ftp://ftp.soest.hawaii.edu/gshhg/",
   license = "LGPL",
   method = list("bb_handler_rget",level = 1, accept_download = "README|bin.*\\.zip$"))

## a more complete definition, which unzips the files after downloading and also
```

```
##  provides an indication of the size of the dataset

my_source <- bb_source(
   id = "gshhg_coastline",
   name = "GSHHG coastline data",
  description = "A Global Self-consistent, Hierarchical, High-resolution Geography Database",
   doc_url = "http://www.soest.hawaii.edu/pwessel/gshhg",
   citation = "Wessel, P., and W. H. F. Smith, A Global Self-consistent, Hierarchical,
     High-resolution Shoreline Database, J. Geophys. Res., 101, 8741-8743, 1996",
   source_url = "ftp://ftp.soest.hawaii.edu/gshhg/*",
   license = "LGPL",
   method = list("bb_handler_rget", level = 1, accept_download = "README|bin.*\\.zip$"),
   postprocess = list("bb_unzip"),
   collection_size = 0.6)

## define a data repository configuration
cf <- bb_config("/my/repo/root")

## add this source to the repository
cf <- bb_add(cf, my_source)

## Not run:
## sync the repo
bb_sync(cf)

## End(Not run)
```

`bb_source_us_buildings` *Example bowerbird data source: Microsoft US Buildings*

Description

This function constructs a data source definition for the Microsoft US Buildings data set. This data set contains 124,885,597 computer generated building footprints in all 50 US states. NOTE: currently, the downloaded zip files will not be unzipped automatically. Work in progress.

Usage

```
bb_source_us_buildings(states)
```

Arguments

`states`	character: (optional) one or more US state names for which to download data. If missing, data from all states will be downloaded. See the reference page for valid state names

Value

a tibble with columns as specified by bb_source

References

https://github.com/Microsoft/USBuildingFootprints

See Also

bb_example_sources, bb_config, bb_handler_rget

Examples

```
## Not run:
## define a configuration and add this buildings data source to it
##  only including data for the District of Columbia and Hawaii
cf <- bb_config(tempdir()) %>%
  bb_add(bb_source_us_buildings(states = c("District of Columbia", "Hawaii")))

## synchronize (download) the data
bb_sync(cf)

## End(Not run)
```

---

bb_subset *Keep only selected data_sources in a bowerbird configuration*

Description

Keep only selected data_sources in a bowerbird configuration

Usage

```
bb_subset(config, idx)
```

Arguments

config	bb_config: a bowerbird configuration (as returned by bb_config)
idx	logical or numeric: index vector of data_source rows to retain

Value

configuration object

See Also

bb_source, bb_config

Examples

```
## Not run:
  cf <- bb_config("/my/file/root") %>%
    bb_add(bb_example_sources()) %>%
    bb_subset(1:2)

## End(Not run)
```

`bb_summary` *Produce a summary of a bowerbird configuration*

Description

This function produces a summary of a bowerbird configuation in HTML or Rmarkdown format. If you are maintaining a data collection on behalf of other users, or even just for yourself, it may be useful to keep an up-to-date HTML summary of your repository in an accessible location. Users can refer to this summary to see which data are in the repository and some details about them.

Usage

```
bb_summary(
  config,
  file = tempfile(fileext = ".html"),
  format = "html",
  inc_license = TRUE,
  inc_auth = TRUE,
  inc_size = TRUE,
  inc_access_function = TRUE,
  inc_path = TRUE
)
```

Arguments

`config`	bb_config: a bowerbird configuration (as returned by `bb_config`)
`file`	string: path to file to write summary to. A temporary file is used by default
`format`	string: produce HTML ("html") or Rmarkdown ("Rmd") file?
`inc_license`	logical: include each source's license and citation details?
`inc_auth`	logical: include information about authentication for each data source (if applicable)?
`inc_size`	logical: include each source's size (disk space) information?
`inc_access_function`	logical: include each source's access function?
`inc_path`	logical: include each source's local file path?

Value

path to the summary file in HTML or Rmarkdown format

Examples

```
## Not run:
  cf <- bb_config("/my/file/root") %>%
    bb_add(bb_example_sources())
  browseURL(bb_summary(cf))

## End(Not run)
```

---

bb_sync *Run a bowerbird data repository synchronization*

Description

This function takes a bowerbird configuration object and synchronizes each of the data sources defined within it. Data files will be downloaded if they are not present on the local machine, or if the configuration has been set to update local files.

Usage

```
bb_sync(
  config,
  create_root = FALSE,
  verbose = FALSE,
  catch_errors = TRUE,
  confirm_downloads_larger_than = 0.1,
  dry_run = FALSE
)
```

Arguments

config	bb_config: configuration as returned by bb_config
create_root	logical: should the data root directory be created if it does not exist? If this is FALSE (default) and the data root directory does not exist, an error will be generated
verbose	logical: if TRUE, provide additional progress output
catch_errors	logical: if TRUE, catch errors and continue the synchronization process. The sync process works through data sources sequentially, and so if catch_errors is FALSE, then an error during the synchronization of one data source will prevent all subsequent data sources from synchronizing

confirm_downloads_larger_than
: numeric or NULL: if non-negative, bb_sync will ask the user for confirmation to download any data source of size greater than this number (in GB). A value of zero will trigger confirmation on every data source. A negative or NULL value will not prompt for confirmation. Note that this only applies when R is being used interactively. The expected download size is taken from the collection_size parameter of the data source, and so its accuracy is dependent on the accuracy of the data source definition

dry_run
: logical: if TRUE, bb_sync will do a dry run of the synchronization process without actually downloading files

Details

Note that when bb_sync is run, the local_file_root directory must exist or create_root=TRUE must be specified (i.e. bb_sync(...,create_root=TRUE)). If create_root=FALSE and the directory does not exist, bb_sync will fail with an error.

Value

a tibble with the name, id, source_url, sync success status, and files of each data source. Data sources that contain multiple source URLs will appear as multiple rows in the returned tibble, one per source_url. files is a tibble with columns url (the URL the file was downloaded from), file (the path to the file), and note (either "downloaded" for a file that was downloaded, "local copy" for a file that was not downloaded because there was already a local copy, or "decompressed" for files that were extracted from a downloaded (or already-locally-present) compressed file. url will be NA for "decompressed" files

See Also

bb_config, bb_source

Examples

```
## Not run:
  ## Choose a location to store files on the local file system.
  ## Normally this would be an explicit choice by the user, but here
  ## we just use a temporary directory for example purposes.

  td <- tempdir()
  cf <- bb_config(local_file_root = td)

  ## Bowerbird must then be told which data sources to synchronize.
  ## Let's use data from the Australian 2016 federal election, which is provided as one
  ## of the example data sources:

 my_source <- bb_example_sources("Australian Election 2016 House of Representatives data")

  ## Add this data source to the configuration:

  cf <- bb_add(cf, my_source)
```

```
## Once the configuration has been defined and the data source added to it,
## we can run the sync process.
## We set \code{verbose=TRUE} so that we see additional progress output:

status <- bb_sync(cf, verbose = TRUE)

## The files in this data set have been stored in a data-source specific
## subdirectory of our local file root:

status$files[[1]]

## We can run this at any later time and our repository will update if the source has changed:

status2 <- bb_sync(cf, verbose = TRUE)

## End(Not run)
```

---

bb_wget *Make a wget call*

Description

This function is an R wrapper to the command-line `wget` utility, which is called using either the `exec_wait` or the `exec_internal` function from the sys package. Almost all of the parameters to `bb_wget` are translated into command-line flags to `wget`. Call `bb_wget("help")` to get more information about wget's command line flags. If required, command-line flags without equivalent `bb_wget` function parameters can be passed via the `extra_flags` parameter.

Usage

```
bb_wget(
  url,
  recursive = TRUE,
  level = 1,
  wait = 0,
  accept,
  reject,
  accept_regex,
  reject_regex,
  exclude_directories,
  restrict_file_names,
  progress,
  user,
  password,
  output_file,
  robots_off = FALSE,
  timestamping = FALSE,
```

```
  no_if_modified_since = FALSE,
  no_clobber = FALSE,
  no_parent = TRUE,
  no_check_certificate = FALSE,
  relative = FALSE,
  adjust_extension = FALSE,
  retr_symlinks = FALSE,
  extra_flags = character(),
  verbose = FALSE,
  capture_stdout = FALSE,
  quiet = FALSE,
  debug = FALSE
)
```

Arguments

`url`	string: the URL to retrieve
`recursive`	logical: if true, turn on recursive retrieving
`level`	integer >=0: recursively download to this maximum depth level. Only applicable if `recursive=TRUE`. Specify 0 for infinite recursion. See https://www.gnu.org/software/wget/manual/wget.html#Recursive-Download for more information about wget's recursive downloading
`wait`	numeric >=0: wait this number of seconds between successive retrievals. This option may help with servers that block multiple successive requests, by introducing a delay between requests
`accept`	character: character vector with one or more entries. Each entry specifies a comma-separated list of filename suffixes or patterns to accept. Note that if any of the wildcard characters '*', '?', '[', or ']' appear in an element of accept, it will be treated as a filename pattern, rather than a filename suffix. In this case, you have to enclose the pattern in quotes, for example `accept="\"*.csv\""`
`reject`	character: as for `accept`, but specifying filename suffixes or patterns to reject
`accept_regex`	character: character vector with one or more entries. Each entry provides a regular expression that is applied to the complete URL. Matching URLs will be accepted for download
`reject_regex`	character: as for `accept_regex`, but specifying regular expressions to reject
`exclude_directories`	character: character vector with one or more entries. Each entry specifies a comma-separated list of directories you wish to exclude from download. Elements may contain wildcards
`restrict_file_names`	character: vector of one of more strings from the set "unix", "windows", "nocontrol", "ascii", "lowercase", and "uppercase". See https://www.gnu.org/software/wget/manual/wget.html#index-Windows-file-names for more information on this parameter. `bb_config` sets this to "windows" by default: if you are downloading files from a server with a port (http://somewhere.org:1234/) Unix will allow the ":" as part of directory/file names, but Windows will not (the

":" will be replaced by "+"). Specifying `restrict_file_names="windows"` causes Windows-style file naming to be used

`progress` string: the type of progress indicator you wish to use. Legal indicators are "dot" and "bar". "dot" prints progress with dots, with each dot representing a fixed amount of downloaded data. The style can be adjusted: "dot:mega" will show 64K per dot and 3M per line; "dot:giga" shows 1M per dot and 32M per line. See https://www.gnu.org/software/wget/manual/wget.html#index-dot-style for more information

`user` string: username used to authenticate to the remote server

`password` string: password used to authenticate to the remote server

`output_file` string: save wget's output messages to this file

`robots_off` logical: by default wget considers itself to be a robot, and therefore won't recurse into areas of a site that are excluded to robots. This can cause problems with servers that exclude robots (accidentally or deliberately) from parts of their sites containing data that we want to retrieve. Setting `robots_off=TRUE` will add a "-e robots=off" flag, which instructs wget to behave as a human user, not a robot. See https://www.gnu.org/software/wget/manual/wget.html#Robot-Exclusion for more information about robot exclusion

`timestamping` logical: if `TRUE`, don't re-retrieve a remote file unless it is newer than the local copy (or there is no local copy)

`no_if_modified_since`
logical: applies when retrieving recursively with timestamping (i.e. only downloading files that have changed since last download, which is achieved using `bb_config(...,clobber=1)`). The default method for timestamping is to issue an "If-Modified-Since" header on the request, which instructs the remote server not to return the file if it has not changed since the specified date. Some servers do not support this header. In these cases, trying using `no_if_modified_since=TRUE`, which will instead send a preliminary HEAD request to ascertain the date of the remote file

`no_clobber` logical: if `TRUE`, skip downloads that would overwrite existing local files

`no_parent` logical: if `TRUE`, do not ever ascend to the parent directory when retrieving recursively. This is `TRUE` by default, bacause it guarantees that only the files below a certain hierarchy will be downloaded

`no_check_certificate`
logical: if `TRUE`, don't check the server certificate against the available certificate authorities. Also don't require the URL host name to match the common name presented by the certificate. This option might be useful if trying to download files from a server with an expired certificate, but it is clearly a security risk and so should be used with caution

`relative` logical: if `TRUE`, only follow relative links. This can sometimes be useful for restricting what is downloaded in recursive mode

`adjust_extension`
logical: if a file of type 'application/xhtml+xml' or 'text/html' is downloaded and the URL does not end with .htm or .html, this option will cause the suffix '.html' to be appended to the local filename. This can be useful when

	mirroring a remote site that has file URLs that conflict with directories (e.g. http://somewhere.org/this/page which has further content below it, say at http://somewhere.org/this/page/n If "somewhere.org/this/page" is saved as a file with that name, that name can't also be used as the local directory name in which to store the lower-level content. Setting `adjust_extension=TRUE` will cause the page to be saved as "somewhere.org/this/page.html", thus resolving the conflict
`retr_symlinks`	logical: if `TRUE`, follow symbolic links during recursive download. Note that this will only follow symlinks to files, NOT to directories
`extra_flags`	character: character vector of additional command-line flags to pass to wget
`verbose`	logical: print trace output?
`capture_stdout`	logical: if `TRUE`, return 'stdout' and 'stderr' output in the returned object (see exec_internal from the sys package). Otherwise send these outputs to the console
`quiet`	logical: if `TRUE`, suppress wget's output
`debug`	logical: if `TRUE`, wget will print lots of debugging information. If wget is not behaving as expected, try setting this to `TRUE`

Value

the result of the system call (or if `bb_wget("--help")` was called, a message will be issued). The returned object will have components 'status' and (if `capture_stdout` was `TRUE`) 'stdout' and 'stderr'

See Also

`bb_install_wget`, `bb_find_wget`

Examples

```
## Not run:
  ## get help about wget command line parameters
  bb_wget("help")

## End(Not run)
```

---

bb_zenodo_source *Generate a bowerbird data source object for a Zenodo data set*

Description

Generate a bowerbird data source object for a Zenodo data set

Usage

```
bb_zenodo_source(id, use_latest = FALSE)
```

Arguments

id	: the ID of the data set
use_latest	logical: if TRUE, use the most recent version of the data set (if there is one). The most recent version might have a different data set ID to the one provided here

Value

A tibble containing the data source definition, as would be returned by bb_source

See Also

bb_source

Examples

```
## Not run:
  ## generate the source object for the dataset
  ##   'Ichtyological data of Station de biologie des Laurentides 2019'
  src <- bb_zenodo_source(3533328)

  ## download it to a temporary directory
  data_dir <- tempfile()
  dir.create(data_dir)
  res <- bb_get(src, local_file_root = data_dir, verbose = TRUE)
  res$files

## End(Not run)
```

---

bowerbird **bowerbird**

Description

Often it's desirable to have local copies of third-party data sets. Fetching data on the fly from remote sources can be a great strategy, but for speed or other reasons it may be better to have local copies. This is particularly common in environmental and other sciences that deal with large data sets (e.g. satellite or global climate model products). Bowerbird is an R package for maintaining a local collection of data sets from a range of data providers.

Author(s)

Maintainer: Ben Raymond <ben.raymond@aad.gov.au>

Authors:

- Ben Raymond <ben.raymond@aad.gov.au>
- Michael Sumner

Other contributors:

- Miles McBain <miles.mcbain@gmail.com> [reviewer, contributor]
- Leah Wasser [reviewer, contributor]

References

https://github.com/AustralianAntarcticDivision/bowerbird

See Also

Useful links:

- https://docs.ropensci.org/bowerbird
- https://github.com/ropensci/bowerbird
- Report bugs at https://github.com/ropensci/bowerbird/issues

Index