Package: REDCapSync (via r-universe)

June 9, 2026

Title Encapsulated 'REDCap' Projects for Synchronized Data Pipelines

Version 0.1.1

Description Wraps dozens of 'REDCap' API endpoints into a standardized R6 object. Research Electronic Data Capture ('REDCap') is a survey and database web application software maintained by Vanderbilt University. It has a robust application programming interface (API) utilized by several R packages. 'REDCapSync' uses 'redcapAPI' and 'REDCapR' behind-the-scenes to retrieve all metadata, data, and log details for a project. To minimize unnecessary server calls, the interim 'REDCap' log is analyzed and used to only update necessary records. Furthermore, the user can define custom datasets that save to a directory. Those datasets continue to refresh when projects are synced. Having a secure, standardized, API-efficient, project-agnostic R object for 'REDCap' projects, streamlines downstream use in scripts, functions, and shiny applications.

License GPL (>= 3)

URL https://github.com/thecodingdocs/REDCapSync, https://thecodingdocs.github.io/REDCapSync/

BugReports https://github.com/thecodingdocs/REDCapSync/issues

Depends R (>= 4.1)

Imports checkmate, cli, dplyr, hoardr, lubridate, openxlsx2, R6, readxl, redcapAPI, REDCapR, skimr, stats, stringr, tools, utils

Suggests keyring, knitr, listviewer, rmarkdown, spelling, testthat (>= 3.3.0), withr

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE, r6 = TRUE)

Config/pak/sysreqs libicu-dev libsodium-dev libssl-dev libx11-dev

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

Date/Publication 2026-06-09 01:00:07 UTC

RemoteUrl https://github.com/thecodingdocs/redcapsync

RemoteRef HEAD

RemoteSha ce48f5a536473ddef67784772e5d84d43bce86b7

Contents

`cache_clear` *Clear your cached projects*

Description

Finds the location of the cache established by `hoard` and deletes stored project information (not data)! If you provide `project_names`, it will remove only those projects from the cache. If you want to truly delete the project files, you must do so at the project directory you set up.

Usage

```
cache_clear(project_names = NULL)
```

Arguments

`project_names`	character vector of project project_names previously setup. If NULL, will get all from `get_projects()`

Details

The cache only stores information like project_name, token_name, directory location, and other details from `setup_project()`.The default location of the cache location is defined by using R_USER_CACHE_DIR if set. Otherwise, it follows platform conventions via hoardr, saving a file "R/REDCapSync/projects.rds". No direct project data is stored in the cache. Notably, tokens and data are not stored here. The key variables stored in the cache are...

- `project_name` - unique identifier for REDCapSync package
- `redcap_uri` - server location
- `token_name` - where to find token environment with `Sys.getenv()`
- `dir_path` - where to saved project and associated files locally
- `project_id` - obtained from API call and "locks-in" the connection
- `redcap_version` - obtained from API call and affects links
- `last_sync` and `sync_frequency` - informs REDCap sync of when to update
- other variables from project info and some internal package mechanics

Value

Message of outcome and invisible NULL.

See Also

`vignette("Cache", package = "REDCapSync")` setup_project for initializing projects

Examples

```
## Not run:
cache_clear("OLD_PROJECT")
cache_clear() # every project

## End(Not run)
```

---

config *Configuration*

Description

[Experimental] Internal configuration helpers used to retrieve package configuration values from options or environment variables.

Configuration is resolved in the following order:

1. `getOption("redcapsync.config.option.name")`
2. `Sys.getenv("REDCAPSYNC_CONFIG_OPTION_NAME")` # can use .Renviron file!
3. Default if unable to find and validate from above.

A vignette will be written for options/configuration in later versions when stable.

Usage

```
config
```

Details

The config function is operational but only some methods presently affect internal code. Most users will not ever need to modify default config. This is included to improve future versions of the package.

- Working configs: `allow.test.names`, `cache.dir`, `keyring`, `keyring.service`, `xlsx.header.color`, `xlsx.header.font.size`, `xlsx.header.font.color`, `xlsx.body.font.size`, `xlsx.body.font.color`, `xlsx.font.name`.
- Partial coverage: `offline`, `verbose`.
- Placeholder: `show.api.messages`

allow.test.names:
Logical for `setup_project()` allowing `project_name` starting with TEST_. Default is `FALSE`. This is provided for developers and testing environments.

```
# check current value package is using...
config$allow.test.names()
# set with options (which will be prioritized over envvar)
getOption("redcapsync.config.allow.test.names") # get
options(redcapsync.config.allow.test.names = FALSE) # set

# set with ennvar (which will be prioritized when options not defined)
Sys.getenv("REDCAPSYNC_CONFIG_ALLOW_TEST_NAMES")  # get
Sys.setenv(REDCAPSYNC_CONFIG_ALLOW_TEST_NAMES = FALSE) # or set in .Renviron
```

show.api.messages:
Logical for showing display API messages from REDCapR and redcapAPI. Default is `FALSE`.

```
# check current value package is using...
config$show.api.messages()
# set with options (which will be prioritized over envvar)
getOption("redcapsync.config.show.api.messages") # get
options(redcapsync.config.show.api.messages = FALSE) # set

# set with ennvar (which will be prioritized when options not defined)
Sys.getenv("REDCAPSYNC_CONFIG_SHOW_API_MESSAGES") # get
Sys.setenv(REDCAPSYNC_CONFIG_SHOW_API_MESSAGES = FALSE) # or set in .Renviron
```

verbose:
Logical for showing display API messages from REDCapR and redcapAPI. Default is `FALSE`.

```
# check current value package is using...
config$verbose()
# set with options (which will be prioritized over envvar)
getOption("redcapsync.config.verbose") # get
options(redcapsync.config.verbose = FALSE) # set

# set with ennvar (which will be prioritized when options not defined)
Sys.getenv("REDCAPSYNC_CONFIG_VERBOSE") # get
Sys.setenv(REDCAPSYNC_CONFIG_VERBOSE = FALSE) # or set in .Renviron
```

offline:

Logical for offline, which if TRUE will block any API calls. Default is FALSE.

```
# check current value package is using...
config$offline()
# set with options (which will be prioritized over envvar)
getOption("redcapsync.config.offline") # get
options(redcapsync.config.offline = FALSE) # set

# set with ennvar (which will be prioritized when options not defined)
Sys.getenv("REDCAPSYNC_CONFIG_OFFLINE") # get
Sys.setenv(REDCAPSYNC_CONFIG_OFFLINE = FALSE) # or set in .Renviron
```

cache.dir:

Character file path overriding the default cache directory. Default follow system standards via tools::R_user_dir or R_USER_CACHE_DIR

```
# check current value package is using...
config$cache.dir()
# set with options (which will be prioritized over envvar)
getOption("redcapsync.config.cache.dir") # get
options(redcapsync.config.cache.dir = "file/path/to/keep/cache") # set

# set with ennvar (which will be prioritized when options not defined)
Sys.getenv("REDCAPSYNC_CONFIG_CACHE_DIR") # get
Sys.setenv(REDCAPSYNC_CONFIG_CACHE_DIR = "file/path/to/keep/cache") # set
```

keyring:

Character keyring name (parameter from keyring package). Default is NULL, which is at the system level. For locking use a keyring like "REDCapSync"

```
# check current value package is using...
config$keyring()
# set with options (which will be prioritized over envvar)
getOption("redcapsync.config.keyring") # get
options(redcapsync.config.keyring = "REDCapSync") # set

# set with ennvar (which will be prioritized when options not defined)
Sys.getenv("REDCAPSYNC_CONFIG_KEYRING") # get
Sys.setenv(REDCAPSYNC_CONFIG_KEYRING = "REDCapSync") # set
```

keyring.service:

Character keyring service name (parameter from keyring package). Default is "R-REDCapSync".

```
# check current value package is using...
config$keyring.service()
# set with options (which will be prioritized over envvar)
getOption("redcapsync.config.keyring.service") # get
options(redcapsync.config.keyring.service = "REDCapSync") # set
```

```
# set with ennvar (which will be prioritized when options not defined)
Sys.getenv("REDCAPSYNC_CONFIG_KEYRING_SERVICE") # get
Sys.setenv(REDCAPSYNC_CONFIG_KEYRING_SERVICE = "REDCapSync") # set
```

xlsx.header.color:

color name for hex color character for header color of exported excel sheets

```
# check current value package is using...
config$xlsx.header.color()
# set with options (which will be prioritized over envvar)
getOption("redcapsync.config.xlsx.header.color") # get
options(redcapsync.config.xlsx.header.color = "REDCapSync") # set

# set with ennvar (which will be prioritized when options not defined)
Sys.getenv("REDCAPSYNC_CONFIG_XLSX_HEADER_COLOR") # get
Sys.setenv(REDCAPSYNC_CONFIG_XLSX_HEADER_COLOR = "green") # set
```

xlsx.header.font.size:

color name for hex color character for header color of exported excel sheets

```
# check current value package is using...
config$xlsx.header.font.size()
# set with options (which will be prioritized over envvar)
getOption("redcapsync.config.xlsx.header.font.size") # get
options(redcapsync.config.xlsx.header.font.size = "REDCapSync") # set

# set with ennvar (which will be prioritized when options not defined)
Sys.getenv("REDCAPSYNC_CONFIG_XLSX_HEADER_FONT_SIZE") # get
Sys.setenv(REDCAPSYNC_CONFIG_XLSX_HEADER_FONT_SIZE = "12") # set
```

xlsx.header.font.color:

color name for hex color character for header color of exported excel sheets

```
# check current value package is using...
config$xlsx.header.font.color()
# set with options (which will be prioritized over envvar)
getOption("redcapsync.config.xlsx.header.font.color") # get
options(redcapsync.config.xlsx.header.font.color = "REDCapSync") # set

# set with ennvar (which will be prioritized when options not defined)
Sys.getenv("REDCAPSYNC_CONFIG_XLSX_HEADER_FONT_COLOR") # get
Sys.setenv(REDCAPSYNC_CONFIG_XLSX_HEADER_FONT_COLOR = "white") # set
```

xlsx.body.font.size:

color name for hex color character for header color of exported excel sheets

```
# check current value package is using...
config$xlsx.body.font.size()
# set with options (which will be prioritized over envvar)
getOption("redcapsync.config.xlsx.body.font.size") # get
```

```
options(redcapsync.config.xlsx.body.font.size = "REDCapSync") # set

# set with ennvar (which will be prioritized when options not defined)
Sys.getenv("REDCAPSYNC_CONFIG_XLSX_BODY_FONT_SIZE") # get
Sys.setenv(REDCAPSYNC_CONFIG_XLSX_BODY_FONT_SIZE = "8") # set
```

xlsx.body.font.color:

color name for hex color character for header color of exported excel sheets

```
# check current value package is using...
config$xlsx.body.font.color()
# set with options (which will be prioritized over envvar)
getOption("redcapsync.config.xlsx.body.font.color") # get
options(redcapsync.config.xlsx.body.font.color = "REDCapSync") # set

# set with ennvar (which will be prioritized when options not defined)
Sys.getenv("REDCAPSYNC_CONFIG_XLSX_BODY_FONT_COLOR") # get
Sys.setenv(REDCAPSYNC_CONFIG_XLSX_BODY_FONT_COLOR = "red") # set
```

xlsx.font.name:

color name for hex color character for header color of exported excel sheets

```
# check current value package is using...
config$xlsx.font.name()
# set with options (which will be prioritized over envvar)
getOption("redcapsync.config.xlsx.font.name") # get
options(redcapsync.config.xlsx.font.name = "REDCapSync") # set

# set with ennvar (which will be prioritized when options not defined)
Sys.getenv("REDCAPSYNC_CONFIG_XLSX_FONT_NAME") # get
Sys.setenv(REDCAPSYNC_CONFIG_XLSX_FONT_NAME = "Georgia") # set
```

Value

list of functions that returns config values

Option Names (searched first)

```
option_list <- list(
  redcapsync.config.allow.test.names = NULL,
  redcapsync.config.show.api.messages = NULL,
  redcapsync.config.verbose = NULL,
  redcapsync.config.offline = NULL,
  redcapsync.config.cache.dir = NULL,
  redcapsync.config.keyring = NULL,
  redcapsync.config.keyring.service = NULL,
  redcapsync.config.xlsx.header.color = NULL,
  redcapsync.config.xlsx.header.font.size = NULL,
  redcapsync.config.xlsx.header.font.color = NULL,
  redcapsync.config.xlsx.body.font.size = NULL,
```

```
  redcapsync.config.xlsx.body.font.color = NULL,
  redcapsync.config.xlsx.font.name = NULL
)
```

Environment Variable Names (searched second)

```
envvar_list <- list(
  REDCAPSYNC_CONFIG_ALLOW_TEST_NAMES = NA,
  REDCAPSYNC_CONFIG_SHOW_API_MESSAGES = NA,
  REDCAPSYNC_CONFIG_VERBOSE = NA,
  REDCAPSYNC_CONFIG_OFFLINE = NA,
  R_USER_CACHE_DIR = NA, # affects your entire cache for any package
  REDCAPSYNC_CONFIG_CACHE_DIR = NA, # affects only REDCapSync Cache
  REDCAPSYNC_CONFIG_KEYRING = NA,
  REDCAPSYNC_CONFIG_KEYRING_SERVICE = NA,
  REDCAPSYNC_CONFIG_XLSX_HEADER_COLOR = NA,
  REDCAPSYNC_CONFIG_XLSX_HEADER_FONT_SIZE = NA,
  REDCAPSYNC_CONFIG_XLSX_HEADER_FONT_COLOR = NA,
  REDCAPSYNC_CONFIG_XLSX_BODY_FONT_SIZE = NA,
  REDCAPSYNC_CONFIG_XLSX_BODY_FONT_COLOR = NA,
  REDCAPSYNC_CONFIG_XLSX_FONT_NAME = NA
)
```

See Also

`vignette("Projects", package = "REDCapSync")` setup_project for initializing projects

Examples

```
# disable with environment variable
Sys.setenv(REDCAPSYNC_CONFIG_OFFLINE = FALSE)

config$offline()

# change to offline
options(redcapsync.config.offline = TRUE)

config$offline()
```

---

dataset *REDCapSync Dataset Object*

Description

[Experimental] R6 object representing a standardized dataset generated from a REDCapSync project. Use this object to inspect transformed REDCap data, export datasets, and assign prepared analysis tables into the calling environment.

Details

A REDCapSyncDataset can be created ad-hoc from a project with dataset <- project$generate_dataset(). For reusability, you can also define with project$add_dataset() and then load with dataset <- project$load_dataset().

You can add variables manually if you want them to be passed to Excel. But add_field is a feature in development.

Typical workflow:

```
setup project from test object
project <- setup_project(
  project_name = "TEST_CLASSIC",
  dir_path = tempdir()
)
# Create and save a filtered dataset
project$add_dataset(
  dataset_name = "analysis_set",
  filter_field = "var_yesno",
  filter_choices = "Yes",
  field_names = c("ecog_at_diagnosis", "stage_at_diagnosis")
)
# generate dataset for R environment
dataset <- project$load_dataset("analysis_set")
# optional send to global environment
dataset$to_envir(globalenv()) # keep in mind potential name conflicts
# Optional modify (final save depends on what is in the dataset object)
dataset$data$merged$stage_2 <- dataset$data$merged$stage_at_diagnosis == "II"
# save to directory
dataset$save() # can specify `dir_other`, by default saves to output folder
```

This object is designed for users who want a stable dataset output from REDCap without modifying the underlying project. This is also used behind-the-scenes in the RosyREDCap shiny app.

Key features:

- Stores a project-specific dataset definition and resulting data
- Keeps metadata, record details, user information, and comments in sync
- Saves datasets in Excel or CSV formats with optional hyperlinks
- Exports dataset components to a user-specified environment

Value

An R6 REDCapSyncDataset object containing dataset output, metadata, records, user information, and optional REDCap log data.

Public fields

data list of data where names are forms

metadata list of metadata

`records` data.frame of records with timestamps

`users` data.frame of users with timestamps

`log` data.frame of log

`comments` data.frame of comments

Active bindings

`project_details` Read-only list of dataset details from `setup_project()`.

`dataset_details` Read-only list of dataset details

`redcap` Read-only list of dataset details from project

`links` Read-only list of dataset details from project

Methods

Public methods:

- `REDCapSyncDataset$new()`
- `REDCapSyncDataset$print()`
- `REDCapSyncDataset$preview()`
- `REDCapSyncDataset$save()`
- `REDCapSyncDataset$to_envir()`

`REDCapSyncDataset$new()`**:** The end user will not see `dataset$new()`. This is handled internally. Users should construct objects using REDCapSyncProject.

Usage:

```
REDCapSyncDataset$new(
  project,
  dataset_name,
  transformation_type = "default",
  merge_form_name = "merged",
  filter_field = NULL,
  filter_choices = NULL,
  filter_list = NULL,
  filter_strict = TRUE,
  field_names = NULL,
  form_names = NULL,
  exclude_identifiers = FALSE,
  exclude_free_text = FALSE,
  date_handling = "none",
  labelled = TRUE,
  clean = TRUE,
  drop_blanks = FALSE,
  drop_missing_codes = FALSE,
  drop_others = NULL,
  include_metadata = TRUE,
  include_users = TRUE,
  include_records = TRUE,
```

```
  include_log = FALSE,
  annotate_from_log = TRUE,
  include_comments = FALSE
)
```

Arguments:

`project` Project object from `setup_project()` or `load_project()`.

`dataset_name` Character. Name of the dataset to generate or load. If the dataset already exists in the project, the existing definition is reused.

`transformation_type` Character. Data transformation strategy: "default" (preferred merged output), "none" (raw data structure), or "merge_non_repeating" (merge only non-repeating forms). Default is "default".

`merge_form_name` Character. Name used for merged non-repeating records. Default is "merged".

`filter_field` Character. Field used for filtering the dataset.

`filter_choices` Vector. Allowed values for `filter_field`.

`filter_list` List. Named list mapping field names to allowed values. Use instead of `filter_field`/`filter_choices` for more complex filters.

`filter_strict` Logical. If `TRUE`, filters are applied to every form. If `FALSE`, filters apply only to the record identifier. Default is `TRUE`.

`field_names` Character vector. Variables to include in the dataset. Default is `NULL` (all fields).

`form_names` Character vector. Forms to include in the dataset. Default is `NULL` (all forms).

`exclude_identifiers` Logical. Remove identifier fields. Default is `TRUE`.

`exclude_free_text` Logical. Remove free text fields. Default is `FALSE`.

`date_handling` Character. Date handling method: "none", "exclude_dates", "random_shift_by_record", "random_shift_by_project", "zero_by_record", or "zero_by_project". Default is "none".

`labelled` Logical. Convert values to labelled vectors if `TRUE`. Default is `TRUE`.

`clean` Logical. Clean the dataset by standardizing missing values and blanks. Default is `TRUE`.

`drop_blanks` Logical. Drop records with blank fields. Default is `FALSE`.

`drop_missing_codes` Logical. Convert REDCap missing codes to `NA`. Default is `FALSE`.

`drop_others` Character vector of additional values to remove.

`include_metadata` Logical. Include field metadata in the dataset. Default is `TRUE`.

`include_users` Logical. Include user information in the dataset. Default is `TRUE`.

`include_records` Logical. Include record-level details. Default is `TRUE`.

`include_log` Logical. Include REDCap activity log details. Default is `FALSE`.

`annotate_from_log` Logical. Annotate metadata and records using the change log. Default is `TRUE`.

`include_comments` Logical. Include REDCap comments. Default is `FALSE`.

`REDCapSyncDataset$print()`**:** Print some key dataset information

Usage:

```
REDCapSyncDataset$print()
```

`REDCapSyncDataset$preview()`**:** Saves temporary excel and opens as a preview

Usage:

```
REDCapSyncDataset$preview(with_links = TRUE)
```

Arguments:

`with_links` Logical. Include hyperlinks in Excel exports. Default is `TRUE`.

`REDCapSyncDataset$save()`: Return flat list

Usage:

```
REDCapSyncDataset$save(
  with_links = TRUE,
  separate = FALSE,
  use_csv = FALSE,
  dir_other = NULL,
  file_name = NULL
)
```

Arguments:

`with_links` Logical. Include hyperlinks in Excel exports. Default is `TRUE`.

`separate` Logical. Save each form as a separate file instead of a multi-sheet workbook. Default is `FALSE`.

`use_csv` Logical. Write CSV files instead of Excel. Default is `FALSE`.

`dir_other` Character. Directory where the dataset file should be saved. Defaults to the project's output folder.

`file_name` Character. Base file name for saved datasets. Defaults to `<project_name>_<dataset_name>`.

`REDCapSyncDataset$to_envir()`: export dataset to envir of your choosing. Keep in mind potential name conflicts

Usage:

```
REDCapSyncDataset$to_envir(envir = NULL)
```

Arguments:

`envir` Environment to assign exported dataset objects. Default is `NULL`.

See Also

project for using the project objects `vignette("Datasets", package = "REDCapSync")` `vignette("RosyREDCap", package = "REDCapSync")`

Examples

```
project <- load_project("TEST_CLASSIC")

dataset <- project$generate_dataset(
  dataset_name = "stage_2_patients",
  filter_field = "stage_at_diagnosis",
  filter_choices = "II",
  field_names = c("ecog_at_diagnosis", "stage_at_diagnosis")
 )

dataset$save(dir_other = tempdir())
```

project *REDCapSync Project Object*

Description

R6 project object for managing REDCap data access and synchronization. The project object is your main interface to REDCap data through the REDCapSync package. It stores your REDCap configuration, data, metadata, and provides methods to sync, transform, and export your data.

Details

Workflow:

Initialize a project with `setup_project()`:

```
project <- setup_project(
  project_name = "FIRST_PROJECT",
  redcap_uri = "https://redcap.yourinstitution.edu/api/",
  dir_path = "~/redcap_projects"
)
```

Synchronize REDCap data into your project object using the REDCap API and log-based change detection:

```
project$sync()
```

Load previously saved REDCap project object using cache:

```
project <- load_project("FIRST_PROJECT)
projects$names() # should see "FIRST_PROJECT"
```

Access REDCap data, metadata, and users via read-only fields:

```
project$data           # Named list of REDCap forms/instruments
project$metadata       # REDCap field definitions, forms, and choices
project$redcap$users   # REDCap user information
project$redcap$log     # REDCap log information

# you can bypass read-only if needed
users <- project$redcap$users
```

Transform and export your data into clean, analysis-ready datasets to be used in R and/or Excel:

```
project$add_dataset("analysis_data", ...) # adds to project for re-use
project$save_datasets() # can save to Excel but is also part of future syncs
```

Upload corrected or new data back to REDCap:

```
project$upload(updated_data)
```

Design Features:

The project object uses **log-based sync**: Since REDCap maintains a detailed change log, `project$sync()` only retrieves and updates records that have changed since the last sync. This dramatically reduces API calls and improves performance for large projects.

All methods use **method chaining**: Methods invisibly return the project object (self), allowing fluent code:

```
dataset <- load_project("TEST_CLASSIC")$sync()$load_dataset("REDCapSync")
```

Read-Only Fields:

- `project$project_name`: Project identifier (set with setup_project)
- `project$dir_path`: Persistent storage directory (set with setup_project)
- `project$data`: Named list of synchronized REDCap forms
- `project$metadata`: REDCap field metadata (forms, fields, choices)
- `project$redcap`: REDCap project info, users, and activity log
- `project$.internal`: Internal project object (for advanced use)

TEST Projects:

All TEST projects start with "TEST_" and by default the average user cannot create a project that starts with "TEST_". They are produced from actual server REDCap projects but never contained real data and were scrubbed of any "real" user/log data. They are subject to change in future versions as the package matures. Keep in mind there are many combinations of REDCap structures to account for.

TEST projects are meant to be used for demonstration and testing purposes. In general, try to use actual REDCap project(s) to explore the package.

This is how to load a project with or without specifying a directory.

```
# return the list of available test projects
REDCapSync::projects$test_names()

project <- load_project("TEST_CLASSIC")

YOUR_DIRECTORY <- getwd()
project <- setup_project("TEST_CLASSIC", dir_path = YOUR_DIRECTORY)
# now can test object functions (does not use API)
project$sync() # will also save datasets
```

The currently available TEST projects are as follows:

- `TEST_CLASSIC`: Classic project (nothing repeats). Has every data type. Contains data for survival analysis testing. Contains comments.
- `TEST_REPEATING`: Non-longitudinal project with repeating instruments.
- `TEST_LONGITUDINAL`: Longitudinal project (with events).
- `TEST_MULTIARM`: Multi-arm and longitudinal project.
- `TEST_EDGE`: Edge-case project. Meant to test things that may be uncommon but should be accounted for as package matures.
- `TEST_DATA`: Placeholder project meant to contain more rich test dataset.
- `TEST_CANCER`: Placeholder project meant to contain more rich test dataset.

- TEST_REDCAPR_SIMPLE: Borrowed with permission from REDCapR.
- TEST_REDCAPR_LONGITUDINAL: Borrowed with permission from REDCapR.
- TEST_REDCAPR_CLIN_TRIAL: Borrowed with permission from REDCapR.

Value

An R6 REDCapSyncProject class generator for internal use. Users interact with instances created by setup_project or load_project.

Active bindings

`project_name` Read-only character string of project_name as assigned with setup_project.

`dir_path` Read-only directory path assigned with setup_project.

`data` Read-only named list where each name is an instrument name. See public methods for REDCapSyncProject.

`metadata` Read-only named list with REDCap metadata. See public methods for REDCapSyncProject.

`redcap` Read-only named list with REDCap information including users and log.

`.internal` Read-only internal project object for custom workflows

Methods

Public methods:

- `REDCapSyncProject$new()`
- `REDCapSyncProject$print()`
- `REDCapSyncProject$sync()`
- `REDCapSyncProject$add_dataset()`
- `REDCapSyncProject$load_dataset()`
- `REDCapSyncProject$remove_datasets()`
- `REDCapSyncProject$generate_dataset()`
- `REDCapSyncProject$save_datasets()`
- `REDCapSyncProject$save_dataset()`
- `REDCapSyncProject$save()`
- `REDCapSyncProject$set_keyring_token()`
- `REDCapSyncProject$test_token()`
- `REDCapSyncProject$url_launch()`
- `REDCapSyncProject$url_record_launch()`
- `REDCapSyncProject$upload()`

`REDCapSyncProject$new()`**:** Active binding are read-only

The end user will not see `project$new()`. This is handled internally. Users should construct objects using `setup_project()`. The remain methods will be accessible to any user.

Usage:

```
REDCapSyncProject$new(project)
```

Arguments:

`project` a list object meant to be stored internally within R6

`REDCapSyncProject$print()`: Print some key project information

Usage:

```
REDCapSyncProject$print()
```

`REDCapSyncProject$sync()`: Updates the REDCap data for (`project` object) by checking REDCap log for changes. Sync is performed according to the `sync_frequency` set in `setup_project()` by default. Use `hard_check` to force a check, or `hard_reset` to force a complete refresh. As a default, this object will be saved to your directory when necessary.

Usage:

```
REDCapSyncProject$sync(
  save_datasets = TRUE,
  save_to_dir = TRUE,
  hard_check = FALSE,
  hard_reset = FALSE
)
```

Arguments:

`save_datasets` Logical. Save dataset outputs during sync. Default is `TRUE`.

`save_to_dir` Logical. Save updated project object to directory. Default is `TRUE`.

`hard_check` Logical. Force REDCap API check regardless of `sync_frequency`. Default is `FALSE`.

`hard_reset` Logical. Overwrite existing dataset files. Default is `FALSE`.

`REDCapSyncProject$add_dataset()`: Add a new dataset entry

Usage:

```
REDCapSyncProject$add_dataset(
  dataset_name,
  transformation_type = "default",
  merge_form_name = "merged",
  filter_field = NULL,
  filter_choices = NULL,
  filter_list = NULL,
  filter_strict = TRUE,
  field_names = NULL,
  form_names = NULL,
  exclude_identifiers = FALSE,
  exclude_free_text = FALSE,
  date_handling = "none",
  labelled = TRUE,
  clean = TRUE,
  drop_blanks = FALSE,
  drop_missing_codes = FALSE,
  drop_others = NULL,
  include_metadata = TRUE,
```

```
  include_records = TRUE,
  include_users = TRUE,
  include_log = FALSE,
  annotate_from_log = TRUE,
  include_comments = TRUE,
  with_links = TRUE,
  separate = FALSE,
  use_csv = FALSE,
  dir_other = NULL,
  file_name = NULL,
  hard_reset = FALSE
)
```

Arguments:

`dataset_name` Character. Name of the dataset configuration to create, load, or reference.

`transformation_type` Character. How to transform data: "default" (merge non-repeating then add to repeating), "none" (no transformation), or "merge_non_repeating" (merge non-repeating only). Default is "default".

`merge_form_name` Character. Name for the merged non-repeating form. Default is "merged".

`filter_field` Character. Field name to filter dataset on.

`filter_choices` Vector. Allowed values for `filter_field`.

`filter_list` List. Named list where names are field names and values are allowed value sets. Alternative to `filter_field`/`filter_choices`.

`filter_strict` Logical. If `TRUE`, filter all forms. If `FALSE`, apply filter only to records (ID column). Default is `TRUE`.

`field_names` Character vector. Fields to include. Default `NULL` includes all fields.

`form_names` Character vector. Forms to include. Default `NULL` includes all forms.

`exclude_identifiers` Logical. Exclude identifier fields. Default is `TRUE`.

`exclude_free_text` Logical. Exclude free-text fields (for deidentification). Default is `FALSE`.

`date_handling` Character. Date handling strategy: "none", "exclude_dates", "random_shift_by_record", "random_shift_by_project", "zero_by_record", or "zero_by_project". Default is "none".

`labelled` Logical. Convert to labelled data if `TRUE`. Default is `TRUE`.

`clean` Logical. Clean data (standardize missing values). Default is `TRUE`.

`drop_blanks` Logical. Drop records with blank fields. Default is `TRUE`.

`drop_missing_codes` Logical. Convert REDCap missing codes to `NA`. Default is `FALSE`.

`drop_others` Character vector of additional values to drop.

`include_metadata` Logical. Include field metadata in dataset. Default is `TRUE`.

`include_records` Logical. Include record-level information. Default is `TRUE`.

`include_users` Logical. Include user information. Default is `TRUE`.

`include_log` Logical. Include REDCap activity log. Default is `FALSE`.

`annotate_from_log` Logical. Annotate data using the REDCap log. Default is `TRUE`.

`include_comments` Logical. Include field comments. Default is `TRUE`.

`with_links` Logical. Include hyperlinks in Excel exports. Default is `FALSE`.

`separate` Logical. Separate each form into distinct files (vs. multi-tab Excel). Default is `FALSE`.

`use_csv` Logical. Use CSV format instead of Excel. Default is `FALSE`.

`dir_other` Character. Output directory (default is project's `output` folder).

`file_name` Character. Dataset file name (default is `<project_name>_<dataset_name>`).

`hard_reset` Logical. Overwrite existing dataset files. Default is `FALSE`.

`REDCapSyncProject$load_dataset()`: Load dataset if previously defined with `add_dataset`.

Usage:

```
REDCapSyncProject$load_dataset(dataset_name, envir = NULL)
```

Arguments:

`dataset_name` Character. Name of the dataset configuration to create, load, or reference.

`envir` Environment to assign dataset objects. Default is `NULL`.

`REDCapSyncProject$remove_datasets()`: Clear all or specified datasets from the `project` object.

Usage:

```
REDCapSyncProject$remove_datasets(dataset_names = NULL)
```

Arguments:

`dataset_names` Character vector. Dataset names to remove/operate on. Default is `NULL`.

`REDCapSyncProject$generate_dataset()`: Generate dataset object. This is usually handled internally for some default behavior but is provided here for ad-hoc custom datasets.

Usage:

```
REDCapSyncProject$generate_dataset(
  dataset_name,
  envir = NULL,
  transformation_type = "default",
  merge_form_name = "merged",
  filter_field = NULL,
  filter_choices = NULL,
  filter_list = NULL,
  filter_strict = TRUE,
  field_names = NULL,
  form_names = NULL,
  exclude_identifiers = FALSE,
  exclude_free_text = FALSE,
  date_handling = "none",
  labelled = TRUE,
  clean = TRUE,
  drop_blanks = FALSE,
  drop_missing_codes = FALSE,
  drop_others = NULL,
  include_metadata = TRUE,
  include_records = TRUE,
  include_users = TRUE,
  include_log = FALSE,
  annotate_from_log = TRUE,
```

```
  include_comments = FALSE
)
```

Arguments:

`dataset_name` Character. Name of the dataset configuration to create, load, or reference.

`envir` Environment to assign dataset objects. Default is `NULL`.

`transformation_type` Character. How to transform data: "default" (merge non-repeating then add to repeating), "none" (no transformation), or "merge_non_repeating" (merge non-repeating only). Default is "default".

`merge_form_name` Character. Name for the merged non-repeating form. Default is "merged".

`filter_field` Character. Field name to filter dataset on.

`filter_choices` Vector. Allowed values for `filter_field`.

`filter_list` List. Named list where names are field names and values are allowed value sets. Alternative to `filter_field`/`filter_choices`.

`filter_strict` Logical. If `TRUE`, filter all forms. If `FALSE`, apply filter only to records (ID column). Default is `TRUE`.

`field_names` Character vector. Fields to include. Default `NULL` includes all fields.

`form_names` Character vector. Forms to include. Default `NULL` includes all forms.

`exclude_identifiers` Logical. Exclude identifier fields. Default is `TRUE`.

`exclude_free_text` Logical. Exclude free-text fields (for deidentification). Default is `FALSE`.

`date_handling` Character. Date handling strategy: "none", "exclude_dates", "random_shift_by_record", "random_shift_by_project", "zero_by_record", or "zero_by_project". Default is "none".

`labelled` Logical. Convert to labelled data if `TRUE`. Default is `TRUE`.

`clean` Logical. Clean data (standardize missing values). Default is `TRUE`.

`drop_blanks` Logical. Drop records with blank fields. Default is `TRUE`.

`drop_missing_codes` Logical. Convert REDCap missing codes to `NA`. Default is `FALSE`.

`drop_others` Character vector of additional values to drop.

`include_metadata` Logical. Include field metadata in dataset. Default is `TRUE`.

`include_records` Logical. Include record-level information. Default is `TRUE`.

`include_users` Logical. Include user information. Default is `TRUE`.

`include_log` Logical. Include REDCap activity log. Default is `FALSE`.

`annotate_from_log` Logical. Annotate data using the REDCap log. Default is `TRUE`.

`include_comments` Logical. Include field comments. Default is `TRUE`.

`REDCapSyncProject$save_datasets()`: saves datasets to Excel via setting provided to `add_dataset`.

Usage:

```
REDCapSyncProject$save_datasets(hard_reset = FALSE)
```

Arguments:

`hard_reset` Logical. Overwrite existing dataset files. Default is `FALSE`.

`REDCapSyncProject$save_dataset()`: saves dataset to Excel via setting provided to `add_dataset`.

Usage:

```
REDCapSyncProject$save_dataset(dataset_name)
```

Arguments:

`dataset_name` Character. Name of the dataset configuration to create, load, or reference.

`REDCapSyncProject$save()`: Save project object to directory chosen with setup_project.

Usage:

```
REDCapSyncProject$save()
```

`REDCapSyncProject$set_keyring_token()`: Set keyring token. See vignette and config for detail.

Usage:

```
REDCapSyncProject$set_keyring_token()
```

`REDCapSyncProject$test_token()`: test connection via communication with API.

Usage:

```
REDCapSyncProject$test_token(keyring_if_error = TRUE)
```

Arguments:

`keyring_if_error` Logical. Launch pop-up for keyring if token is not valid or fails to call REDCap. Max 3 attempts. Default is `TRUE`.

`REDCapSyncProject$url_launch()`: opens links in browser.

Usage:

```
REDCapSyncProject$url_launch(link_type = "home", open_browser = TRUE)
```

Arguments:

`link_type` Character. REDCap link type: "base", "home", "record_home", "records_dashboard", "api", "api_playground", "codebook", "user_rights", "setup", "logging", "designer", "dictionary", "data_quality", or "identifiers".

`open_browser` Logical. Open link in browser. Default is `TRUE`.

`REDCapSyncProject$url_record_launch()`: opens record links in browser.

Usage:

```
REDCapSyncProject$url_record_launch(
  record = NULL,
  page = NULL,
  instance = NULL,
  open_browser = TRUE
)
```

Arguments:

`record` Character. Record ID.

`page` Character. REDCap form/instrument name.

`instance` Character. Repeating instance number.

`open_browser` Logical. Open link in browser. Default is `TRUE`.

`REDCapSyncProject$upload()`: This will only overwrite and new data. It will not directly delete any data. Because this is a function that can mess up your data, use it very carefully. Remember all changes are saved in the REDCap log if there's an issue. Missing rows and columns are allowed!

Usage:

```
REDCapSyncProject$upload(to_be_uploaded)
```

Arguments:

`to_be_uploaded` Data frame in raw coded format ready to upload to REDCap.

See Also

`vignette("REDCapSync", package = "REDCapSync")` projects for shortcuts of cached setup projects `vignette("Tokens", package = "REDCapSync")` setup_project for initializing projects dataset for using the dataset objects `vignette("Datasets", package = "REDCapSync")` `vignette("Uploads", package = "REDCapSync")`

Examples

```
# Load a test project
projects$test_names() # available test projects
project <- setup_project("TEST_CLASSIC", dir_path = tempdir())

# Sync data from REDCap
project$sync()

# Access data and metadata
head(project$data$text)
project$metadata$fields[1:5, ]

forms <- project$metadata$forms # unchanged metadata
fields <- project$metadata$fields # unchanged metadata
choices <- project$metadata$choices # unchanged metadata

users <- project$redcap$users # unchanged users
log <- project$redcap$log # unchanged log

project$test_token()

dataset <- project$load_dataset("REDCapSync")
```

`projects` *Manage REDCapSync projects*

Description

`projects` is the central interface for managing REDCap projects in **REDCapSync**. It provides a convenient, persistent registry of projects that can be accessed across R sessions.

Each time a project is set up or synced, basic metadata is stored locally so that projects can be reloaded and updated without reconfiguration.

Usage

```
projects
```

Format

A named list of functions for managing REDCapSync projects.

Details

`projects` is implemented as a singleton list object and serves as a single entry point to the REDCapSync workflow. All project-level operations—such as setup, loading, syncing, and removal—are accessed through this object.

A key advantage of this design is support for **method chaining**, even without loading the namespace. Because many project methods return project objects, you can write concise, readable workflows that operate in sequence:

```
# load, sync, and generate dataset to global
REDCapSync::projects$load("TEST_CLASSIC")$
  sync()$
  generate_dataset(envir = globalenv(),
                   filter_field = "var_branching",
                   filter_choices = "Yes")
```

Methods

`print()` Display a summary of all registered projects.

- Shows total number of projects
- Lists project names
- Indicates how many are due for sync

Returns the project data frame invisibly.

`any()` Check whether any projects are registered.

Returns a logical scalar.

`n()` Return the number of registered projects.

Returns an integer.

`names()` Return the names of registered projects.

Returns a character vector.

`test_names()` Return the names of test projects.

Returns a character vector.

`df()` Retrieve the project registry as a data frame.

Returns a data frame containing stored project metadata.

`load(project_name)` Load a project by name. This is a wrapper for load_project.

Arguments

- `project_name`: Character. Name of the project.

Returns a `REDCapSyncProject` object.

See Also

`vignette("REDCapSync", package = "REDCapSync")` `vignette("Cache", package = "REDCapSync")` `vignette("RosyREDCap", package = "REDCapSync")` setup_project for initializing projects project for using the project objects

Examples

```
project_df <- projects$df()
projects$print()
projects$any()
projects$n()
projects$names()
project <- projects$load("TEST_CLASSIC")
```

`setup_project` *Setup or Load REDCapSync Project*

Description

Setup or load a REDCapSync project object. Prepares a new REDCapSync project by recording the REDCap URI, token name, sync settings, and optional data selection preferences.

Usage

```
setup_project(
  project_name,
  dir_path,
  redcap_uri,
  token_name = paste0("REDCAPSYNC_", project_name),
  sync_frequency = "daily",
  labelled = TRUE,
  get_type = "identified",
  records = NA,
  fields = NA,
  forms = NA,
  events = NA,
  filter_logic = NA,
  id_position = 1L,
  get_users = TRUE,
  get_data = TRUE,
  batch_size_download = 1000L,
  batch_size_upload = 500L,
  get_entire_log = FALSE,
  log_days = 10L,
  log_drop_details = FALSE,
  log_drop_exports = FALSE,
  get_files = FALSE,
```

```
  get_file_repository = FALSE,
  original_file_names = FALSE,
  add_default_datasets = TRUE,
  timezone = Sys.timezone(),
  hard_reset = FALSE
)

load_project(project_name)
```

Arguments

project_name	Character scalar. Unique uppercase project name with no spaces or symbols.
dir_path	Optional character scalar. Directory where REDCap project files are stored. If omitted, the project is only available in the current R session and cannot be persisted to disk.
redcap_uri	Character scalar. Base URL of the REDCap API endpoint.
token_name	Optional character scalar. Environment variable name used to store the REDCap token. Defaults to REDCAPSYNC_<project_name>.
sync_frequency	Character scalar. Allowed values are "always", "hourly", "daily", "weekly", "monthly", "once", and "never". This setting controls how often future syncs are considered due.
labelled	Logical scalar. If TRUE, data will be preserved as labelled values. Default is TRUE.
get_type	Character scalar. REDCap API export type. One of "identified", "deidentified", "deidentified_strict", or "deidentified_super_strict". Default is "identified".
records	Optional character vector of record IDs to request.
fields	Optional character vector of field names to request.
forms	Optional character vector of form names to request.
events	Optional character vector of event names to request.
filter_logic	Optional character scalar. REDCap filter logic used to limit returned records or record-events.
id_position	Integer scalar of the variable that uniquely identifies the subject (typically record_id). This defaults to the first variable in the data dictionary.
get_users	Logical scalar. If TRUE, the project will be configured to retrieve REDCap users during sync.
get_data	Logical scalar. If TRUE, the project will be configured to retrieve REDCap data during sync.
batch_size_download	Integer scalar. Number of records to request per download batch. Default is 1000.
batch_size_upload	Integer scalar. Number of records to process per upload batch. Default is 500.
get_entire_log	Logical scalar. If TRUE, REDCap activity logs are retrieved in full. Default is FALSE.

`log_days` Integer scalar. Number of days of log history to consider when a new project is being set up or a hard reset occurs. Default is `10`.

`log_drop_details` Logical scalar. If `TRUE`, the log details are excluded.

`log_drop_exports` Logical scalar. If `TRUE`, the log export events are excluded.

`get_files` Logical scalar. If `TRUE`, file attachments are configured to be retrieved from REDCap. Default is `FALSE`.

`get_file_repository` Logical scalar. If `TRUE`, the file repository is configured to be retrieved from REDCap. Default is `FALSE`.

`original_file_names` Logical scalar. If `TRUE`, retrieved files keep their original names. Default is `FALSE`.

`add_default_datasets` Logical scalar. If `TRUE`, dataset templates are added to a new project. Default is `TRUE`.

`timezone` Optional character scalar. Time zone used for REDCap data timestamps. Defaults to `Sys.timezone()`.

`hard_reset` Logical scalar. If `TRUE`, any existing project with the same `project_name` is ignored and a fresh project object is initialized. Default is `FALSE`.

Details

Unless `hard_reset = TRUE`, it will first attempt to load an existing project from cache or the supplied `dir_path` before creating a new project object. If settings differ from the loaded project, they will be used or it will trigger a hard reset with a warning. `setup_project()` itself does not perform a full REDCap sync. It configures the project so that subsequent sync actions may retrieve data, users, files, and logs according to the project settings.

Value

R6 project object with REDCapSyncProject class.

See Also

`vignette("REDCapSync", package = "REDCapSync")` `vignette("Tokens", package = "REDCapSync")` projects for shortcuts of cached setup projects project for using the project objects `vignette("Cache", package = "REDCapSync")`

Examples

```
# Initialize the project object with the REDCap API token and URL
save_folder <- tempdir() # replace with real folder
project <- setup_project(
  project_name = "FIRST_PROJECT",
  dir_path = save_folder,
  redcap_uri = "https://redcap.yourinstitution.edu/api/"
```

```
)

# object can be named whatever you choose to assign
# TEST projects can be loaded in addition to real projects
project_test <- load_project("TEST_CLASSIC")
```

---

sync *Synchronize REDCap Data*

Description

Syncs with REDCap via `project` object that user defined with setup_project

Usage

```
sync(
  project_names = NULL,
  save_datasets = TRUE,
  hard_check = FALSE,
  hard_reset = FALSE
)
```

Arguments

`project_names`	character vector of project project_names previously setup. If NULL, will get all from `get_projects()`
`save_datasets`	Logical (TRUE/FALSE). If TRUE, saves datasets to directory.
`hard_check`	Will check REDCap even if not due (see `sync_frequency` parameter from `setup_project()`)
`hard_reset`	Logical that forces a fresh update if TRUE. Default is `FALSE`.

Details

Syncs all projects by default but can be used to hands-free sync one or defined set projects. This is not intended to return project object. User should use `load_project("project_name")`. However, by default will invisibly return the last project in the set of project_names.

Value

Invisibly returns last project object (R6 REDCapSyncProject class)

See Also

`vignette("REDCapSync", package = "REDCapSync")` `vignette("Cache", package = "REDCapSync")` setup_project for initializing projects project for using the project objects

Examples

```
## Not run:
# if you setup 3 projects "ONE", "TWO", "THREE" using [setup_project()]...
sync() # will check all three projects for updates if due for sync
sync("TWO") # will only check and sync project TWO
project <- sync("TWO") # can also be used for invisible return ...
sync(c("ONE","THREE")) # will only check and sync projects ONE and THREE
project <- sync(c("ONE","THREE")) # invisible return of THREE ...

## End(Not run)
```

Index