Package: REDCapR (via r-universe)

October 24, 2024

Title Interaction Between R and REDCap

Description Encapsulates functions to streamline calls from R to the REDCap API. REDCap (Research Electronic Data CAPture) is a web application for building and managing online surveys and databases developed at Vanderbilt University. The Application Programming Interface (API) offers an avenue to access and modify data programmatically, improving the capacity for literate and reproducible programming.

Version 1.3.0

URL https://ouhscbbmc.github.io/REDCapR/, https://github.com/OuhscBbmc/REDCapR, https://www.ouhsc.edu/bbmc/, https://projectredcap.org

BugReports https://github.com/OuhscBbmc/REDCapR/issues

Depends R(>= 3.5.0)

Imports checkmate (>= 2.0), dplyr (>= 1.0), httr (>= 1.4.0), jsonlite, magrittr (>= 1.5), methods, readr (>= 2.0), rlang (>= 0.4), tibble (>= 2.0), tidyr (>= 1.0)

Suggests spelling, covr (>= 3.4), DBI (>= 1.1), kableExtra (>= 1.0), knitr (>= 1.18), odbc (>= 1.1.1), purrr (>= 0.3.4), rmarkdown (>= 2.0), sessioninfo (>= 1.1.1), testthat (>= 3.0), tidyselect, yaml

License MIT + file LICENSE

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.2

Config/testthat/edition 3

Language en-US

NeedsCompilation no

Author Will Beasley [aut, cre]
(<https://orcid.org/0000-0002-5613-5006>), David Bard [ctb]
(<https://orcid.org/0000-0002-3922-8489>), Thomas Wilson [ctb],
John J Aponte [ctb], Rollie Parrish [ctb]
(<https://orcid.org/0000-0001-8858-6381>), Benjamin Nutter
[ctb], Andrew Peters [ctb]
(<https://orcid.org/0000-0003-2487-1268>), Hao Zhu [ctb]
(<https://orcid.org/0000-0002-3386-6076>), Janosch
Linkersdörfer [ctb] (<https://orcid.org/0000-0002-1577-1233>),
Jonathan Mang [ctb] (<https://orcid.org/0000-0003-0518-4710>),
Felix Torres [ctb], Philip Chase [ctb]
(<https://orcid.org/0000-0002-5318-9420>), Victor Castro [ctb]
(<https://orcid.org/0000-0001-7390-6354>), Greg Botwin [ctb],
Stephan Kadauke [ctb]
(<https://orcid.org/0000-0003-2996-8034>), Ezra Porter [ctb]
(<https://orcid.org/0000-0002-4690-8343>), Matthew Schuelke
[ctb] (<https://orcid.org/0000-0001-5755-1725>)

Maintainer Will Beasley <wibeasley@hotmail.com>

Repository CRAN

Date/Publication 2024-10-23 09:10:02 UTC

Contents

`collapse_vector` *Collapse a vector of values into a single string when necessary*

Description

REDCap's API frequently specifies a series of values separated by commas. In the R world, it's easier to keep these values as separate elements in a vector. This functions squashes them together in a single character element (presumably right before the return value is passed to the API)

Usage

```
collapse_vector(elements)
```

Arguments

`elements`	An array of values. Can be `NULL`. Required.

Value

A single character element, where the values are separated by commas. Can be blank. (*i.e.*, `""`).

Author(s)

Will Beasley

Examples

```
library(REDCapR) # Load the package into the current R session.
REDCapR:::collapse_vector(elements = NULL   )
REDCapR:::collapse_vector(elements = letters)
```

`constant` *Collection of REDCap-specific constants*

Description

Collection of constants defined by the REDCap developers.

Usage

```
constant(name)
```

Arguments

`name`	Name of constant. Required character.

Details

Form Completeness

The current constants relate to the 'complete' variable at the end of each form.

- `form_incomplete`: 0L (*i.e.*, an integer)
- `form_unverified`: 1L
- `form_complete`: 2L

Export Rights

See https://your-server/redcap/api/help/?content=exp_users.

- `data_export_rights_no_access` : 0L
- `data_export_rights_deidentified` : 1L
- `data_export_rights_full` : 2L

Form Rights

See https://your-server/redcap/api/help/?content=exp_users. The order of these digits may be unexpected.

- `form_rights_no_access` : 0L
- `form_rights_readonly` : 2L
- `form_rights_edit_form` : 1L
- `form_rights_edit_survey` : 3L

Access Rights

See https://your-server/redcap/api/help/?content=exp_users.

- `access_no` : 0L
- `access_yes` : 1L

To add more, please for and edit constant.R on GitHub and submit a pull request. For instructions, please see Editing files in another user's repository # nolint in the GitHub documentation.

Value

The constant's value. Currently all are single integers, but that could be expanded in the future.

Author(s)

Will Beasley

Examples

```
REDCapR::constant("form_incomplete")  # Returns 0L
REDCapR::constant("form_unverified")  # Returns 1L
REDCapR::constant("form_complete"  )  # Returns 2L

REDCapR::constant("data_export_rights_no_access"   )  # Returns 0L
REDCapR::constant("data_export_rights_deidentified")  # Returns 1L
REDCapR::constant("data_export_rights_full"        )  # Returns 2L

REDCapR::constant("form_rights_no_access")   # Returns 0L
REDCapR::constant("form_rights_readonly" )   # Returns 2L --Notice the order
REDCapR::constant("form_rights_edit_form")   # Returns 1L
REDCapR::constant("form_rights_edit_survey") # Returns 3L

REDCapR::constant("access_no" )  # Returns 0L
REDCapR::constant("access_yes")  # Returns 1L

REDCapR::constant(c(
  "form_complete",
  "form_complete",
  "form_incomplete"
)) # Returns c(2L, 2L, 0L)
REDCapR::constant(c(
  "form_rights_no_access",
  "form_rights_readonly",
  "form_rights_edit_form",
  "form_rights_edit_survey"
)) # Returns c(0L, 2L, 1L, 3L)


constant_to_form_completion( c(0, 2, 1, 2, NA))
constant_to_form_rights(     c(0, 2, 1, 2, NA))
constant_to_export_rights(   c(0, 2, 1, 3, NA))
constant_to_access(          c(0, 1, 1, 0, NA))

## Not run:
# The following line returns an error:
#     Assertion on 'name' failed: Must be a subset of
#     {'form_complete','form_incomplete','form_unverified'},
#     but is {'bad-name'}.

REDCapR::constant("bad-name")    # Returns an error

REDCapR::constant(c("form_complete", "bad-name")) # Returns an error
```

```
## End(Not run)
```

`create_batch_glossary` *Creates a dataset that help batching long-running read and writes*

Description

The function returns a `base::data.frame()` that other functions use to separate long-running read and write REDCap calls into multiple, smaller REDCap calls. The goal is to (1) reduce the chance of time-outs, and (2) introduce little breaks between batches so that the server isn't continually tied up.

Usage

```
create_batch_glossary(row_count, batch_size)
```

Arguments

`row_count`	The number records in the large dataset, before it's split.
`batch_size`	The maximum number of subject records a single batch should contain.

Details

This function can also assist splitting and saving a large data frame to disk as smaller files (such as a .csv). The padded columns allow the OS to sort the batches/files in sequential order.

Value

Currently, a `base::data.frame()` is returned with the following columns,

- `id`: an `integer` that uniquely identifies the batch, starting at `1`.
- `start_index`: the index of the first row in the batch. `integer`.
- `stop_index`: the index of the last row in the batch. `integer`.
- `id_pretty`: a `character` representation of `id`, but padded with zeros.
- `start_index`: a `character` representation of `start_index`, but padded with zeros.
- `stop_index`: a `character` representation of `stop_index`, but padded with zeros.
- `label`: a `character` concatenation of `id_pretty`, `start_index`, and `stop_index_pretty`.

Author(s)

Will Beasley

See Also

See `redcap_read()` for a function that uses `create_batch_glossary`.

Examples

```
REDCapR::create_batch_glossary(100, 50)
REDCapR::create_batch_glossary(100, 25)
REDCapR::create_batch_glossary(100,  3)
REDCapR::create_batch_glossary(  0,  3)
d <- data.frame(
  record_id = 1:100,
  iv        = sample(x=4, size=100, replace=TRUE),
  dv        = rnorm(n=100)
)
REDCapR::create_batch_glossary(nrow(d), batch_size=40)
```

---

kernel_api *REDCapR internal function for calling the REDCap API*

Description

This function is used by other functions to read and write values.

Usage

```
kernel_api(
  redcap_uri,
  post_body,
  config_options,
  encoding = "UTF-8",
  content_type = "text/csv",
  handle_httr = NULL,
  encode_httr = "form"
)
```

Arguments

redcap_uri	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
post_body	List of contents expected by the REDCap API. Required.
config_options	A list of options passed to `httr::POST()`. See details at `httr::httr_options()`. Optional.
encoding	The encoding value passed to `httr::content()`. Defaults to 'UTF-8'.
content_type	The MIME value passed to `httr::content()`. Defaults to 'text/csv'.
handle_httr	The value passed to the `handle` parameter of `httr::POST()`. This is useful for only unconventional authentication approaches. It should be `NULL` for most institutions.
encode_httr	The value passed to the `encode` parameter of `httr::POST()`. Defaults to `"form"`, which is appropriate for most actions. (Currently, the only exception is importing a file, which uses "multipart".)

Details

If the API call is unsuccessful, a value of `base::package_version("0.0.0")` will be returned. This ensures that a the function will always return an object of class base::package_version. It guarantees the value can always be used in `utils::compareVersion()`.

Value

A utils::packageVersion.

Examples

```
config_options <- NULL
uri            <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token          <- "9A068C425B1341D69E83064A2D273A70"
post_body      <- list(
  token    = token,
  content  = "project",
  format   = "csv"
)
## Not run:
kernel <- REDCapR:::kernel_api(uri, post_body, config_options)

# Consume the results in a few different ways.
kernel$result
read.csv(text = kernel$raw_text)
as.list(read.csv(text = kernel$raw_text))

## End(Not run)
```

`metadata_utilities` *Manipulate and interpret the metadata of a REDCap project*

Description

A collection of functions that assists handling REDCap project metadata.

Usage

```
regex_named_captures(pattern, text, perl = TRUE)

checkbox_choices(select_choices)
```

Arguments

`pattern`	The regular expression pattern. Required.
`text`	The text to apply the regex against. Required.
`perl`	Indicates if perl-compatible regexps should be used. Default is `TRUE`. Optional.
`select_choices`	The text containing the choices that should be parsed to determine the `id` and `label` values. Required.

Details

The `regex_named_captures()` function is general, and not specific to REDCap; it accepts any arbitrary regular expression. It returns a `tibble::tibble()` with as many columns as named matches.

The `checkbox_choices()` function is specialized, and accommodates the "select choices" for a *single* REDCap checkbox group (where multiple boxes can be selected). It returns a `tibble::tibble()` with two columns, one for the numeric id and one for the text label.

The parse will probably fail if a label contains a pipe (*i.e.*, `|`), since that the delimiter REDCap uses to separate choices presented to the user.

Value

Currently, a `tibble::tibble()` is returned a row for each match, and a column for each *named* group within a match. For the `retrieve_checkbox_choices()` function, the columns will be.

- `id`: The numeric value assigned to each choice (in the data dictionary).
- `label`: The label assigned to each choice (in the data dictionary).

Author(s)

Will Beasley

References

See the official documentation for permissible characters in a checkbox label. *I'm bluffing here, because I don't know where this is located. If you know, please tell me.*

Examples

```
# The weird ranges are to avoid the pipe character;
#   PCRE doesn't support character negation.
pattern_boxes <- "(?<=\\A| \\| )(?<id>\\d{1,}), (?<label>[\x20-\x7B\x7D-\x7E]{1,})(?= \\| |\\Z)"

choices_1 <- paste0(
  "1, American Indian/Alaska Native | ",
  "2, Asian | ",
  "3, Native Hawaiian or Other Pacific Islander | ",
  "4, Black or African American | ",
  "5, White | ",
  "6, Unknown / Not Reported"
)

# This calls the general function, and requires the correct regex pattern.
REDCapR::regex_named_captures(pattern=pattern_boxes, text=choices_1)

# This function is designed specifically for the checkbox values.
REDCapR::checkbox_choices(select_choices=choices_1)

## Not run:
uri         <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token       <- "9A068C425B1341D69E83064A2D273A70"
```

```
ds_metadata <- redcap_metadata_read(uri, token)$data
choices_2  <- ds_metadata[ds_metadata$field_name == "race", ]$select_choices_or_calculations

REDCapR::regex_named_captures(pattern = pattern_boxes, text = choices_2)

## End(Not run)

path_3     <- system.file(package = "REDCapR", "test-data/projects/simple/metadata.csv")
ds_metadata_3  <- read.csv(path_3)
choices_3 <- ds_metadata_3[ds_metadata_3$field_name=="race", "select_choices_or_calculations"]
REDCapR::regex_named_captures(pattern = pattern_boxes, text = choices_3)
```

`redcap_arm_export` *Export Arms*

Description

Export Arms of a REDCap project

Usage

```
redcap_arm_export(
  redcap_uri,
  token,
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

`redcap_uri`	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
`token`	The user-specific string that serves as the password for a project. Required.
`verbose`	A boolean value indicating if `messages` should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
`config_options`	A list of options passed to `httr::POST()`. See details at `httr::httr_options()`. Optional.
`handle_httr`	The value passed to the `handle` parameter of `httr::POST()`. This is useful for only unconventional authentication approaches. It should be `NULL` for most institutions. Optional.

Value

Currently, a list is returned with the following elements:

- `has_arms`: a `logical` value indicating if the REDCap project has arms (*i.e.*, "TRUE") or is a classic non-longitudinal project (*i.e.*, "FALSE").
- `data`: a `tibble::tibble()` with one row per arm. The columns are `arm_number` (an integer) and `arm_name` (a human-friendly string).
- `success`: A boolean value indicating if the operation was apparently successful.
- `status_code`: The http status code of the operation.
- `outcome_message`: A human readable string indicating the operation's outcome.
- `elapsed_seconds`: The duration of the function.
- `raw_text`: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the `raw_text` is returned as an empty string to save RAM.

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
uri            <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"

# Query a classic project with 3 arms
token_1  <- "14817611F9EA1A6E149BBDC37134E8EA" # arm-multiple-delete
result_1 <- REDCapR::redcap_arm_export(redcap_uri=uri, token=token_1)
result_1$has_arms
result_1$data

# Query a classic project without arms
token_2  <- "F9CBFFF78C3D78F641BAE9623F6B7E6A" # simple-write
result_2 <- REDCapR::redcap_arm_export(redcap_uri=uri, token=token_2)
result_2$has_arms
result_2$data

## End(Not run)
```

`redcap_column_sanitize` *Sanitize to adhere to REDCap character encoding requirements*

Description

Replace non-ASCII characters with legal characters that won't cause problems when writing to a REDCap project.

Usage

```
redcap_column_sanitize(
  d,
  column_names = colnames(d),
  encoding_initial = "latin1",
  substitution_character = "?"
)
```

Arguments

`d`	The `base::data.frame()` or `tibble::tibble()` containing the dataset used to update the REDCap project. Required.
`column_names`	An array of `character` values indicating the names of the variables to sanitize. Optional.
`encoding_initial`	An array of `character` values indicating the names of the variables to sanitize. Optional.
`substitution_character`	The `character` value that replaces characters that were unable to be appropriately matched.

Details

Letters like an accented 'A' are replaced with a plain 'A'.

This is a thin wrapper around `base::iconv()`. The `ASCII//TRANSLIT` option does the actual transliteration work. As of `R 3.1.0`, the OSes use similar, but different, versions to convert the characters. Be aware of this in case you notice OS-dependent differences.

Value

A data frame with same columns, but whose character values have been sanitized.

Author(s)

Will Beasley

Examples

```
# Typical examples are not shown because they require non-ASCII encoding,
#   which makes the package documentation less portable.

dirty <- data.frame(
  id     = 1:3,
  names  = c("Ekstr\xf8m", "J\xf6reskog", "bi\xdfchen Z\xfcrcher")
)

REDCapR::redcap_column_sanitize(dirty)
```

---

redcap_dag_read *Read data access groups from a REDCap project*

Description

This function reads all available data access groups from REDCap an returns them as a `tibble::tibble()`.

Usage

```
redcap_dag_read(
  redcap_uri,
  token,
  http_response_encoding = "UTF-8",
  locale = readr::default_locale(),
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

redcap_uri	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
token	The user-specific string that serves as the password for a project. Required.
http_response_encoding	The encoding value passed to `httr::content()`. Defaults to 'UTF-8'.
locale	a `readr::locale()` object to specify preferences like number, date, and time formats. This object is passed to `readr::read_csv()`. Defaults to `readr::default_locale()`.
verbose	A boolean value indicating if `messages` should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
config_options	A list of options passed to `httr::POST()`. See details at `httr::httr_options()`. Optional.
handle_httr	The value passed to the `handle` parameter of `httr::POST()`. This is useful for only unconventional authentication approaches. It should be `NULL` for most institutions. Optional.

Value

Currently, a list is returned with the following elements:

- `data`: A `tibble::tibble()` of all data access groups of the project.
- `success`: A boolean value indicating if the operation was apparently successful.
- `status_codes`: A collection of http status codes, separated by semicolons. There is one code for each batch attempted.
- `outcome_messages`: A collection of human readable strings indicating the operations' semicolons. There is one code for each batch attempted. In an unsuccessful operation, it should contain diagnostic information.
- `elapsed_seconds`: The duration of the function.

Author(s)

Jonathan M. Mang

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
uri     <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token   <- "9A068C425B1341D69E83064A2D273A70"
REDCapR::redcap_dag_read(redcap_uri=uri, token=token)$data

## End(Not run)
```

---

redcap_delete *Delete records in a REDCap project*

Description

Delete existing records by their ID from REDCap.

Usage

```
redcap_delete(
  redcap_uri,
  token,
  records_to_delete,
  arm_of_records_to_delete = NULL,
  verbose = TRUE,
```

```
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

redcap_uri	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
token	The user-specific string that serves as the password for a project. Required.
records_to_delete	A character vector of the project's record_id values to delete. Required.
arm_of_records_to_delete	A single integer reflecting the arm containing the records to be deleted. Leave it as NULL if the project has no arms and is not longitudinal. Required if the REDCap project has arms. See Details below.
verbose	A boolean value indicating if messages should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
config_options	A list of options passed to httr::POST(). See details at httr::httr_options(). Optional.
handle_httr	The value passed to the handle parameter of httr::POST(). This is useful for only unconventional authentication approaches. It should be NULL for most institutions. Optional.

Details

REDCap requires that at least one record_id value be passed to the delete call.

When the project has arms, REDCapR has stricter requirements than REDCap. If the REDCap project has arms, a value must be passed to arm_of_records_to_delete. This is a different behavior than calling the server through cURL –which if no arm number is specified, then all arms are cleared of the specified record_ids.

Note that all longitudinal projects technically have arms, even if only one arm is defined. Therefore a value of arm_number must be specified for all longitudinal projects.

Value

Currently, a list is returned with the following elements:

- success: A boolean value indicating if the operation was apparently successful.
- status_code: The http status code of the operation.
- outcome_message: A human readable string indicating the operation's outcome.
- records_affected_count: The number of records inserted or updated.
- elapsed_seconds: The duration of the function.
- raw_text: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the raw_text is returned as an empty string to save RAM.

Author(s)

Jonathan Mang, Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
if (FALSE) {
records_to_delete <- c(102, 103, 105, 120)

# Deleting from a non-longitudinal project with no defined arms:
REDCapR::redcap_delete(
  redcap_uri               = uri,
  token                    = token,
  records_to_delete        = records_to_delete,
)

# Deleting from a project that has arms or is longitudinal:
arm_number <- 2L # Not the arm name
REDCapR::redcap_delete(
  redcap_uri               = uri,
  token                    = token,
  records_to_delete        = records_to_delete,
  arm_of_records_to_delete = arm_number
)
}
```

---

redcap_event_instruments

Enumerate the instruments to event mappings

Description

Export the instrument-event mappings for a project (i.e., how the data collection instruments are designated for certain events in a longitudinal project). (Copied from "Export Instrument-Event Mappings" method of REDCap API documentation, v.10.5.1)

Usage

```
redcap_event_instruments(
  redcap_uri,
  token,
  arms = NULL,
```

```
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

redcap_uri	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
token	The user-specific string that serves as the password for a project. Required.
arms	A character string of arms to retrieve. Defaults to all arms of the project.
verbose	A boolean value indicating if messages should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
config_options	A list of options passed to httr::POST(). See details at httr::httr_options(). Optional.
handle_httr	The value passed to the handle parameter of httr::POST(). This is useful for only unconventional authentication approaches. It should be NULL for most institutions. Optional.

Value

Currently, a list is returned with the following elements,

- data: A tibble::tibble() where each row represents one column in the REDCap dataset.
- success: A boolean value indicating if the operation was apparently successful.
- status_code: The http status code of the operation.
- outcome_message: A human readable string indicating the operation's outcome.
- elapsed_seconds: The duration of the function.
- raw_text: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the raw_text is returned as an empty string to save RAM.

Author(s)

Victor Castro, Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

See Also

redcap_instruments()

Examples

```
## Not run:
uri                <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"

# Longitudinal project with one arm
token_1  <- "76B4A71A0158BD34C98F10DA72D5F27C" # "arm-single-longitudinal" test project
REDCapR::redcap_arm_export(redcap_uri=uri, token=token_1)$data
REDCapR::redcap_event_instruments(redcap_uri=uri, token=token_1)$data

# Project with two arms
token_2  <- "DA6F2BB23146BD5A7EA3408C1A44A556" # "longitudinal" test project
REDCapR::redcap_arm_export(redcap_uri=uri, token=token_2)$data
REDCapR::redcap_event_instruments(redcap_uri=uri, token=token_2)$data
REDCapR::redcap_event_instruments(redcap_uri=uri, token=token_2, arms = c("1", "2"))$data
REDCapR::redcap_event_instruments(redcap_uri=uri, token=token_2, arms = "2")$data

# Classic project (without arms) throws an error
token_3  <- "9A068C425B1341D69E83064A2D273A70" # "simple" test project
REDCapR::redcap_arm_export(redcap_uri=uri, token=token_3)$data
# REDCapR::redcap_event_instruments(redcap_uri=uri, token=token_3)$data

## End(Not run)
```

`redcap_event_read` *Export Events*

Description

Export Events of a REDCap project

Usage

```
redcap_event_read(
  redcap_uri,
  token,
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

`redcap_uri`	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
`token`	The user-specific string that serves as the password for a project. Required.
`verbose`	A boolean value indicating if `messages` should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.

config_options A list of options passed to httr::POST(). See details at httr::httr_options(). Optional.

handle_httr The value passed to the handle parameter of httr::POST(). This is useful for only unconventional authentication approaches. It should be NULL for most institutions. Optional.

Value

Currently, a list is returned with the following elements:

- data: a tibble::tibble() with one row per arm-event combination. The columns are event_name (a human-friendly string), arm_num (an integer), unique_event_name (a string), custom_event_label (a string), and event_id (an integer).
- success: A boolean value indicating if the operation was apparently successful.
- status_code: The http status code of the operation.
- outcome_message: A human readable string indicating the operation's outcome.
- elapsed_seconds: The duration of the function.
- raw_text: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the raw_text is returned as an empty string to save RAM.

Author(s)

Ezra Porter, Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
uri            <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"

# Query a longitudinal project with a single arm and 3 events
token_1  <- "76B4A71A0158BD34C98F10DA72D5F27C" # arm-single-longitudinal
result_1 <- REDCapR::redcap_event_read(redcap_uri=uri, token=token_1)
result_1$data

# Query a longitudinal project with 2 arms and complex arm-event mappings
token_2  <- "DA6F2BB23146BD5A7EA3408C1A44A556" # longitudinal
result_2 <- REDCapR::redcap_event_read(redcap_uri=uri, token=token_2)
result_2$data

# Query a classic project without events
token_3  <- "F9CBFFF78C3D78F641BAE9623F6B7E6A" # simple-write
result_3 <- REDCapR::redcap_event_read(redcap_uri=uri, token=token_3)
```

```
result_3$data

## End(Not run)
```

`redcap_file_download_oneshot` *Download a file from a REDCap project record*

Description

This function uses REDCap's API to download a file.

Usage

```
redcap_file_download_oneshot(
  file_name = NULL,
  directory = NULL,
  overwrite = FALSE,
  redcap_uri,
  token,
  record,
  field,
  event = "",
  repeat_instance = NULL,
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

`file_name`	The name of the file where the downloaded file is saved. If empty the original name of the file will be used and saved in the default directory. Optional.
`directory`	The directory where the file is saved. By default current directory. Optional
`overwrite`	Boolean value indicating if existing files should be overwritten. Optional
`redcap_uri`	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
`token`	The user-specific string that serves as the password for a project. Required.
`record`	The record ID where the file is to be imported. Required
`field`	The name of the field where the file is saved in REDCap. Required
`event`	The name of the event where the file is saved in REDCap. Optional
`repeat_instance`	(only for projects with repeating instruments/events) The repeat instance number of the repeating event (if longitudinal) or the repeating instrument (if classic or longitudinal). Default value is '1'. Optional

verbose	A boolean value indicating if messages should be printed to the R console during the operation. Optional.
config_options	A list of options passed to httr::POST(). See details at httr::httr_options(). Optional.
handle_httr	The value passed to the handle parameter of httr::POST(). This is useful for only unconventional authentication approaches. It should be NULL for most institutions. Optional.

Details

For files in a repeating instrument, don't specify repeating_instrument. The server only needs field (name) and repeating_instance.

The function redcap_download_file_oneshot() is soft-deprecated as of REDCapR 1.2.0. Please rename to redcap_file_download_oneshot().

Value

Currently, a list is returned with the following elements,

- success: A boolean value indicating if the operation was apparently successful.
- status_code: The http status code of the operation.
- outcome_message: A human readable string indicating the operation's outcome.
- records_affected_count: The number of records inserted or updated.
- affected_ids: The subject IDs of the inserted or updated records.
- elapsed_seconds: The duration of the function.
- raw_text: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the raw_text is returned as an empty string to save RAM.
- file_name: The name of the file persisted to disk. This is useful if the name stored in REDCap is used (which is the default).

Author(s)

Will Beasley, John J. Aponte

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
uri     <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token   <- "F9CBFFF78C3D78F641BAE9623F6B7E6A" # simple-write
record  <- 1
```

```
field   <- "mugshot"
# event <- "" # only for longitudinal projects

result_1 <- REDCapR::redcap_file_download_oneshot(
  record        = record,
  field         = field,
  redcap_uri    = uri,
  token         = token
)
base::unlink("mugshot-1.jpg")

(full_name <- base::tempfile(pattern="mugshot", fileext = ".jpg"))
result_2   <- REDCapR::redcap_file_download_oneshot(
  file_name     = full_name,
  record        = record,
  field         = field,
  redcap_uri    = uri,
  token         = token
)
base::unlink(full_name)

(relative_name <- "ssss.jpg")
result_3 <- REDCapR::redcap_file_download_oneshot(
  file_name    = relative_name,
  record       = record,
  field        = field,
  redcap_uri   = uri,
  token        = token
)
base::unlink(relative_name)

## End(Not run)
```

`redcap_file_upload_oneshot` *Upload a file into to a REDCap project record*

Description

This function uses REDCap's API to upload a file.

Usage

```
redcap_file_upload_oneshot(
  file_name,
  record,
  redcap_uri,
  token,
  field,
```

```
  event = "",
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

file_name	The name of the relative or full file to be uploaded into the REDCap project. Required.
record	The record ID where the file is to be imported. Required
redcap_uri	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
token	The user-specific string that serves as the password for a project. Required.
field	The name of the field where the file is saved in REDCap. Required
event	The name of the event where the file is saved in REDCap. Optional
verbose	A boolean value indicating if messages should be printed to the R console during the operation. Optional.
config_options	A list of options passed to httr::POST(). See details at httr::httr_options(). Optional.
handle_httr	The value passed to the handle parameter of httr::POST(). This is useful for only unconventional authentication approaches. It should be NULL for most institutions. Optional.

Details

Currently, the function doesn't modify any variable types to conform to REDCap's supported variables. See validate_for_write() for a helper function that checks for some common important conflicts.

The function redcap_upload_file_oneshot() is soft-deprecated as of REDCapR 1.2.0. Please rename to redcap_file_upload_oneshot().

Value

Currently, a list is returned with the following elements:

- success: A boolean value indicating if the operation was apparently successful.
- status_code: The http status code of the operation.
- outcome_message: A human readable string indicating the operation's outcome.
- records_affected_count: The number of records inserted or updated.
- affected_ids: The subject IDs of the inserted or updated records.
- elapsed_seconds: The duration of the function.
- raw_text: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the raw_text is returned as an empty string to save RAM.

Author(s)

Will Beasley, John J. Aponte

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (ie, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectre examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
# Define some constants
uri    <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token  <- "F9CBFFF78C3D78F641BAE9623F6B7E6A" # simple-write
field  <- "mugshot"
event  <- "" # only for longitudinal events

# Upload a single image file.
record    <- 1
file_path <- system.file("test-data/mugshot-1.jpg", package = "REDCapR")

REDCapR::redcap_file_upload_oneshot(
  file_name  = file_path,
  record     = record,
  field      = field,
  redcap_uri = uri,
  token      = token
)

# Upload a collection of five images.
records    <- 1:5
file_paths <- system.file(
  paste0("test-data/mugshot-", records, ".jpg"),
  package="REDCapR"
)

for (i in seq_along(records)) {
  record    <- records[i]
  file_path <- file_paths[i]
  REDCapR::redcap_file_upload_oneshot(
    file_name  = file_path,
    record     = record,
    field      = field,
    redcap_uri = uri,
    token      = token
  )
}

## End(Not run)
```

`redcap_instruments` *Enumerate the instruments (forms)*

Description

Export a list of the data collection instruments for a project. This includes their unique instrument name as seen in the second column of the Data Dictionary, as well as each instrument's corresponding instrument label, which is seen on a project's left-hand menu when entering data. The instruments will be ordered according to their order in the project. (Copied from "Export Instruments (Data Entry Forms)" method of REDCap API documentation, v.10.5.1)

Usage

```
redcap_instruments(
  redcap_uri,
  token,
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

`redcap_uri`	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
`token`	The user-specific string that serves as the password for a project. Required.
`verbose`	A boolean value indicating if `messages` should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
`config_options`	A list of options passed to `httr::POST()`. See details at `httr::httr_options()`. Optional.
`handle_httr`	The value passed to the `handle` parameter of `httr::POST()`. This is useful for only unconventional authentication approaches. It should be `NULL` for most institutions. Optional.

Value

Currently, a list is returned with the following elements,

- `data`: A `tibble::tibble()` where each row represents one column in the REDCap dataset.
- `success`: A boolean value indicating if the operation was apparently successful.
- `status_code`: The http status code of the operation.
- `outcome_message`: A human readable string indicating the operation's outcome.
- `elapsed_seconds`: The duration of the function.
- `raw_text`: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the `raw_text` is returned as an empty string to save RAM.

Author(s)

Victor Castro, Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

See Also

redcap_event_instruments()

Examples

```
## Not run:
uri           <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token         <- "9A068C425B1341D69E83064A2D273A70"
ds_instrument <- REDCapR::redcap_instruments(redcap_uri=uri, token=token)$data

## End(Not run)
```

---

redcap_instrument_download

Download REDCap Instruments

Description

Download instruments as a pdf, with or without responses.

Usage

```
redcap_instrument_download(
  file_name = NULL,
  directory = NULL,
  overwrite = FALSE,
  redcap_uri,
  token,
  record = character(0),
  instrument = "",
  event = "",
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

file_name	The name of the file where the downloaded pdf is saved. Optional.
directory	The directory where the file is saved. By default current directory. Optional.
overwrite	Boolean value indicating if existing files should be overwritten. Optional.
redcap_uri	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
token	The user-specific string that serves as the password for a project. Required.
record	The record ID of the instrument(s). If empty, the responses are blank. Optional.
instrument	The instrument(s) to download. If empty, all instruments are returned. Optional.
event	The unique event name. For a longitudinal project, if record is not blank and event is blank, it will return data for all events from that record. If record is not blank and event is not blank, it will return data only for the specified event from that record. Optional.
verbose	A boolean value indicating if messages should be printed to the R console during the operation. Optional.
config_options	A list of options passed to httr::POST(). See details at httr::httr_options(). Optional.
handle_httr	The value passed to the handle parameter of httr::POST(). This is useful for only unconventional authentication approaches. It should be NULL for most institutions. Optional.

Details

Currently, the function doesn't modify any variable types to conform to REDCap's supported variables. See validate_for_write() for a helper function that checks for some common important conflicts.

The function redcap_download_instrument() is soft-deprecated as of REDCapR 1.2.0. Please rename to redcap_instrument_download().

Value

Currently, a list is returned with the following elements,

- success: A boolean value indicating if the operation was apparently successful.
- status_code: The http status code of the operation.
- outcome_message: A human readable string indicating the operation's outcome.
- record_id: The record_id of the instrument.
- elapsed_seconds: The duration of the function.
- raw_text: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the raw_text is returned as an empty string to save RAM.
- file_name: The name of the file persisted to disk. This is useful if the name stored in REDCap is used (which is the default).

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
uri     <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token   <- "F9CBFFF78C3D78F641BAE9623F6B7E6A" # simple-write
# event <- "" # only for longitudinal projects

(full_name <- base::tempfile(pattern="instruments-all-records-all", fileext = ".pdf"))
result_1   <- REDCapR::redcap_instrument_download(
  file_name     = full_name,
  redcap_uri    = uri,
  token         = token
)
base::unlink(full_name)

(full_name <- base::tempfile(pattern="instruments-all-record-1-", fileext = ".pdf"))
result_2   <- REDCapR::redcap_instrument_download(
  record        = 5,
  file_name     = full_name,
  redcap_uri    = uri,
  token         = token
)
base::unlink(full_name)
(full_name <- base::tempfile(pattern="instrument-1-record-1-", fileext=".pdf"))
result_3   <- REDCapR::redcap_instrument_download(
  record        = 5,
  instrument    = "health",
  file_name     = full_name,
  redcap_uri    = uri,
  token         = token
)
base::unlink(full_name)

## End(Not run)
```

`redcap_log_read` *Get the logging of a project.*

Description

This function reads the available logging messages from REDCap as a `tibble::tibble()`.

Usage

```
redcap_log_read(
  redcap_uri,
  token,
  log_begin_date = Sys.Date() - 7L,
  log_end_date = Sys.Date(),
  record = NULL,
  user = NULL,
  http_response_encoding = "UTF-8",
  locale = readr::default_locale(),
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

`redcap_uri`	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
`token`	The user-specific string that serves as the password for a project. Required.
`log_begin_date`	Return the events occurring after midnight of this date. Defaults to the past week; this default mimics the behavior in the browser and also reduces the strain on your server.
`log_end_date`	Return the events occurring before 24:00 of this date. Defaults to today.
`record`	Return the events belonging only to specific record (referring to an existing record name). Defaults to `NULL` which returns logging activity related to all records. If a record value passed, it must be a single value.
`user`	Return the events belonging only to specific user (referring to an existing username). Defaults to `NULL` which returns logging activity related to all users. If a user value passed, it must be a single value.
`http_response_encoding`	The encoding value passed to `httr::content()`. Defaults to 'UTF-8'.
`locale`	a `readr::locale()` object to specify preferences like number, date, and time formats. This object is passed to `readr::read_csv()`. Defaults to `readr::default_locale()`.
`verbose`	A boolean value indicating if `messages` should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
`config_options`	A list of options passed to `httr::POST()`. See details at `httr::httr_options()`. Optional.
`handle_httr`	The value passed to the `handle` parameter of `httr::POST()`. This is useful for only unconventional authentication approaches. It should be `NULL` for most institutions. Optional.

Value

Currently, a list is returned with the following elements:

- `data`: An R `tibble::tibble()` of all data access groups of the project.
- `success`: A boolean value indicating if the operation was apparently successful.
- `status_codes`: A collection of http status codes, separated by semicolons. There is one code for each batch attempted.
- `outcome_messages`: A collection of human readable strings indicating the operations' semicolons. There is one code for each batch attempted. In an unsuccessful operation, it should contain diagnostic information.
- `elapsed_seconds`: The duration of the function.

Author(s)

Jonathan M. Mang, Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
uri          <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token        <- "9A068C425B1341D69E83064A2D273A70"

ds_last_week <- REDCapR::redcap_log_read(redcap_uri=uri, token=token)$data
head(ds_last_week)

ds_one_day <-
  REDCapR::redcap_log_read(
    redcap_uri     = uri,
    token          = token,
    log_begin_date = as.Date("2024-10-11"),
    log_end_date   = as.Date("2024-10-11")
  )$data
head(ds_one_day)

ds_one_day_single_record <-
  REDCapR::redcap_log_read(
    redcap_uri     = uri,
    token          = token,
    log_begin_date = as.Date("2024-10-10"),
    log_end_date   = as.Date("2024-10-10"),
    record         = as.character(3),
    # user           = "unittestphifree"
  )$data
```

```
head(ds_one_day_single_record)

## End(Not run)
```

`redcap_metadata_coltypes` *Suggests a col_type for each field in a REDCap project*

Description

This function inspects a REDCap project to determine a `readr::cols()` object that is compatible with the the project's current definition. It can be copied and pasted into the R code so future calls to the server will produce a `tibble::tibble()` with an equivalent set of data types.

Usage

```
redcap_metadata_coltypes(
  redcap_uri,
  token,
  print_col_types_to_console = TRUE,
  http_response_encoding = "UTF-8",
  locale = readr::default_locale(),
  verbose = FALSE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

`redcap_uri`	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
`token`	The user-specific string that serves as the password for a project. Required.
`print_col_types_to_console`	Should the `readr::cols()` object be printed to the console?
`http_response_encoding`	The encoding value passed to `httr::content()`. Defaults to 'UTF-8'.
`locale`	a `readr::locale()` object to specify preferences like number, date, and time formats. This object is passed to `readr::read_csv()`. Defaults to `readr::default_locale()`.
`verbose`	A boolean value indicating if `messages` should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
`config_options`	A list of options passed to `httr::POST()`. See details at `httr::httr_options()`. Optional.
`handle_httr`	The value passed to the `handle` parameter of `httr::POST()`. This is useful for only unconventional authentication approaches. It should be `NULL` for most institutions. Optional.

Details

`redcap_metadata_coltypes()` returns a `readr::cols()` object in two ways. First, a literal object is returned that can be passed to `redcap_read()` or `redcap_read_oneshot()`.

Second, the function acts as a code generator. It prints text to the console so that it can be copied and pasted into an R file. This is useful to (a) document what fields and data types are expected, and (b) adjust those fields and data types if the defaults can be customized for your needs. For instance, you may choose to exclude some variables or tweak a data type (*e.g.*, changing a patient's height from an integer to a double).

When printing to the console, each data type decision is accompanied by an explanation on the far right. See the output from the examples below. Please file an issue if you think something is too restrictive or can be improved.

The overall heuristic is assign a data type down a waterfall of decisions:

1. Is the field built into REDCap? This includes an autonumber `record_id`, `redcap_event_name`, `redcap_repeat_instrument`, `redcap_repeat_instance`, and an instrument's completion status.
2. What is the field's type? For example, sliders should be an integer, while check marks should be logical(https://stat.ethz.ch/R-manual/R-devel/library/base/html/logical.html.
3. If the field type is "text", what is the validation type? For instance, a postal code should be a character (even though it looks like a number), a "mdy" should be cast to a date, and a "number_2dp" should be cast to a floating point
4. If the field type or validation type is not recognized, the field will be cast to character. This will happen when REDCap develops & releases a new type. If you see something like, "# validation doesn't have an associated col_type. Tell us in a new REDCapR issue", please make sure REDCapR is running the newest GitHub release and file a new issue if it's still not recognized.

For details of the current implementation, the decision logic starts about half-way down in the function's source code

**Validation does NOT Guarantee Conformity*

If you're coming to REDCap from a database world, this will be unexpected. A validation type does NOT guarantee that all retrieved values will conform to complementary the data type. The validation setting affects only the values entered *after* the validation was set.

For example, if values like "abcd" where entered in a field for a few months, then the project manager selected the "integer" validation option, all those "abcd" values remain untouched.

This is one reason `redcap_metadata_coltypes()` prints it suggestions to the console. It allows the developer to adjust the specifications to match the values returned by the API. The the "abcd" scenario, consider (a) changing the type from `col_integer` to `col_character`, (b) excluding the trash values, then (c) in a `dplyr::mutate()` statement, use `readr::parse_integer()` to cast it to the desired type.

Value

A `readr::cols()` object is returned, which can be passed to `redcap_read()` or `redcap_read_oneshot()`.

Additionally objected is printed to the console, see the Details below.

Author(s)

Will Beasley, Philip Chase

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
uri      <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"

# A simple project
token      <- "9A068C425B1341D69E83064A2D273A70" # simple
col_types  <- redcap_metadata_coltypes(uri, token)
redcap_read_oneshot(uri, token, col_types = col_types)$data

# A longitudinal project
token      <- "DA6F2BB23146BD5A7EA3408C1A44A556" # longitudinal
col_types  <- redcap_metadata_coltypes(uri, token)
redcap_read_oneshot(uri, token, col_types = col_types)$data

# A repeating instruments project
token      <- "64720C527CA236880FBA785C9934F02A" # repeating-instruments-sparse
col_types  <- redcap_metadata_coltypes(uri, token)
redcap_read_oneshot(uri, token, col_types = col_types)$data

# A project with every field type and validation type.
#   Notice it throws a warning that some fields use a comma for a decimal,
#   while other fields use a period/dot as a decimal
token      <- "EB1FD5DDE583364AE605629AB7619397" # validation-types-1
col_types  <- redcap_metadata_coltypes(uri, token)
redcap_read_oneshot(uri, token, col_types = col_types)$data

## End(Not run)
```

---

redcap_metadata_read *Export the metadata of a REDCap project*

Description

Export the metadata (as a data dictionary) of a REDCap project as a `tibble::tibble()`. Each row in the data dictionary corresponds to one field in the project's dataset.

Usage

```
redcap_metadata_read(
  redcap_uri,
  token,
  forms = NULL,
  fields = NULL,
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

redcap_uri	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
token	The user-specific string that serves as the password for a project. Required.
forms	An array, where each element corresponds to the REDCap form of the desired fields. Optional.
fields	An array, where each element corresponds to a desired project field. Optional.
verbose	A boolean value indicating if messages should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
config_options	A list of options passed to httr::POST(). See details at httr::httr_options(). Optional.
handle_httr	The value passed to the handle parameter of httr::POST(). This is useful for only unconventional authentication approaches. It should be NULL for most institutions. Optional.

Value

Currently, a list is returned with the following elements:

- data: An R tibble::tibble() of the desired fields (as rows).
- success: A boolean value indicating if the operation was apparently successful.
- status_codes: A collection of http status codes, separated by semicolons. There is one code for each batch attempted.
- outcome_messages: A collection of human readable strings indicating the operations' semicolons. There is one code for each batch attempted. In an unsuccessful operation, it should contain diagnostic information.
- forms_collapsed: The desired records IDs, collapsed into a single string, separated by commas.
- fields_collapsed: The desired field names, collapsed into a single string, separated by commas.
- elapsed_seconds: The duration of the function.

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
uri   <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"

# A simple project
token <- "9A068C425B1341D69E83064A2D273A70" # simple
REDCapR::redcap_metadata_read(redcap_uri=uri, token=token)

# A longitudinal project
token <- "0434F0E9CF53ED0587847AB6E51DE762" # longitudinal
REDCapR::redcap_metadata_read(redcap_uri=uri, token=token)

# A repeating measures
token <- "77842BD8C18D3408819A21DD0154CCF4" # vignette-repeating
REDCapR::redcap_metadata_read(redcap_uri=uri, token=token)

## End(Not run)
```

---

redcap_metadata_write *Import metadata of a REDCap project*

Description

Import metadata (*i.e.*, data dictionary) into a project. Because of this method's destructive nature, it works for only projects in Development status.

Usage

```
redcap_metadata_write(
  ds,
  redcap_uri,
  token,
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

ds	The base::data.frame() or tibble::tibble() to be imported into the REDCap project. Required.
redcap_uri	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
token	The user-specific string that serves as the password for a project. Required.
verbose	A boolean value indicating if messages should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
config_options	A list of options passed to httr::POST(). See details at httr::httr_options(). Optional.
handle_httr	The value passed to the handle parameter of httr::POST(). This is useful for only unconventional authentication approaches. It should be NULL for most institutions. Optional.

Value

Currently, a list is returned with the following elements:

- success: A boolean value indicating if the operation was apparently successful.
- status_code: The http status code of the operation.
- outcome_message: A human readable string indicating the operation's outcome.
- field_count: Number of fields imported.
- elapsed_seconds: The duration of the function.
- raw_text: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the raw_text is returned as an empty string to save RAM.

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki. If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
# Please don't run this example without changing the token to
# point to your server.  It could interfere with our testing suite.
uri            <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token          <- "06C38F6D76B3863DFAE84069D8DBCFFC" # metadata-write

# Read in the dictionary in R's memory from a csv file.
ds_to_write <-
```

```
  readr::read_csv(
    file = system.file(
      "test-data/projects/simple/metadata.csv",
      package = "REDCapR"
    ),
    col_types = readr::cols(.default = readr::col_character())
  )
ds_to_write

# Import the dictionary into the REDCap project
REDCapR::redcap_metadata_write(
  ds          = ds_to_write,
  redcap_uri  = uri,
  token       = token
)

## End(Not run)
```

`redcap_next_free_record_name` *Determine free available record ID*

Description

Determines the next available record ID.

Usage

```
redcap_next_free_record_name(
  redcap_uri,
  token,
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

`redcap_uri`	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
`token`	The user-specific string that serves as the password for a project. Required.
`verbose`	A boolean value indicating if `messages` should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
`config_options`	A list of options passed to `httr::POST()`. See details at `httr::httr_options()`. Optional.

handle_httr — The value passed to the handle parameter of httr::POST(). This is useful for only unconventional authentication approaches. It should be NULL for most institutions. Optional.

Details

If the API call is unsuccessful, a value of character(0) will be returned (*i.e.*, an empty vector). This ensures that a the function will always return an object of class base::character.

Value

a base::character vector of either length 1 (if successful) or length 0 (if not successful).

Note

Documentation in REDCap 8.4.0

To be used by projects with record auto-numbering enabled, this method exports the next potential record ID for a project. It generates the next record name by determining the current maximum numerical record ID and then incrementing it by one.

Note: This method does not create a new record, but merely determines what the next record name would be.

If using Data Access Groups (DAGs) in the project, this method accounts for the special formatting of the record name for users in DAGs (e.g., DAG-ID); in this case, it only assigns the next value for ID for all numbers inside a DAG. For example, if a DAG has a corresponding DAG number of 223 wherein records 223-1 and 223-2 already exist, then the next record will be 223-3 if the API user belongs to the DAG that has DAG number 223. (The DAG number is auto-assigned by REDCap for each DAG when the DAG is first created.)

When generating a new record name in a DAG, the method considers all records in the entire project when determining the maximum record ID, including those that might have been originally created in that DAG but then later reassigned to another DAG.

Note: This method functions the same even for projects that do not have record auto-numbering enabled.

Examples

```
uri      <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token    <- "9A068C425B1341D69E83064A2D273A70"
# Returns 6
REDCapR::redcap_next_free_record_name(redcap_uri = uri, token = token)
```

redcap_project *A Reference Class to make later calls to REDCap more convenient*

Description

This `Reference Class` represents a REDCap project. Once some values are set that are specific to a REDCap project (such as the URI and token), later calls are less verbose (such as reading and writing data).

Fields

`redcap_uri` The URI (uniform resource identifier) of the REDCap project. Required.

`token` token The user-specific string that serves as the password for a project. Required.

Methods

`read( batch_size = 100L, interbatch_delay = 0, records = NULL, fields = NULL, forms = NULL, events = NULL, ra`
Exports records from a REDCap project.

`write( ds_to_write, batch_size = 100L, interbatch_delay = 0, continue_on_error = FALSE, verbose = TRUE, co`
Imports records to a REDCap project.

Examples

```
uri     <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token   <- "9A068C425B1341D69E83064A2D273A70"
## Not run:
project <- REDCapR::redcap_project$new(redcap_uri=uri, token=token)
ds_all  <- project$read()

# Demonstrate how repeated calls are more concise when the token and
#   url aren't always passed.
ds_skinny <- project$read(fields=c("record_id", "sex", "height"))$data

ids_of_males    <- ds_skinny$record_id[ds_skinny$sex==1]
ids_of_shorties <- ds_skinny$record_id[ds_skinny$height < 40]

ds_males        <- project$read(records=ids_of_males, batch_size=2)$data
ds_shorties     <- project$read(records=ids_of_shorties)$data


## End(Not run)
if (FALSE) {
# Switch the Genders
sex_original   <- ds_skinny$sex
ds_skinny$sex  <- (1 - ds_skinny$sex)
project$write(ds_skinny)

# Switch the Genders back
ds_skinny$sex <- sex_original
```

```
project$write(ds_skinny)
}
```

`redcap_project_info_read` *Export project information.*

Description

This function exports some of the basic attributes of a given REDCap project, such as the project's title, if it is longitudinal, if surveys are enabled, the time the project was created and moved to production. Returns a `tibble::tibble()`.

Usage

```
redcap_project_info_read(
  redcap_uri,
  token,
  http_response_encoding = "UTF-8",
  locale = readr::default_locale(),
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

`redcap_uri`	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
`token`	The user-specific string that serves as the password for a project. Required.
`http_response_encoding`	The encoding value passed to `httr::content()`. Defaults to 'UTF-8'.
`locale`	a `readr::locale()` object to specify preferences like number, date, and time formats. This object is passed to `readr::read_csv()`. Defaults to `readr::default_locale()`.
`verbose`	A boolean value indicating if `messages` should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
`config_options`	A list of options passed to `httr::POST()`. See details at `httr::httr_options()`. Optional.
`handle_httr`	The value passed to the `handle` parameter of `httr::POST()`. This is useful for only unconventional authentication approaches. It should be `NULL` for most institutions. Optional.

Details

Timezones

Several datetime variables are returned, such as the project's `creation_time`. For the time to be meaningful, you'll need to set the time zone because this function uses `readr::read_csv()`, which assigns UTC if no timezone is specified. Find your server's location listed in `base::OlsonNames()`, and pass it to readr's `readr::locale()` function, and then pass that to `redcap_project_info_read()`. See the examples below

For more timezone details see the Timezones section of readr's locales vignette.

Columns not yet added This function casts columns into data types that we think are most natural to use. For example, `in_production` is cast from a 0/1 to a boolean TRUE/FALSE. All columns not (yet) recognized by REDCapR will be still be returned to the client, but cast as a character. If the REDCap API adds a new column in the future, please alert us in a REDCapR issue and we'll expand the casting specifications.

External Modules

A project's `external_modules` cell contains all its approved EMs, separated by a column. Consider using a function like `tidyr::separate_rows()` to create a long data structure.

Value

Currently, a list is returned with the following elements:

- `data`: An R `tibble::tibble()` of all data access groups of the project.
- `success`: A boolean value indicating if the operation was apparently successful.
- `status_codes`: A collection of http status codes, separated by semicolons. There is one code for each batch attempted.
- `outcome_messages`: A collection of human readable strings indicating the operations' semicolons. There is one code for each batch attempted. In an unsuccessful operation, it should contain diagnostic information.
- `elapsed_seconds`: The duration of the function.

Author(s)

Will Beasley, Stephan Kadauke

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
# Specify your project uri and token(s).
uri                 <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token_simple        <- "9A068C425B1341D69E83064A2D273A70"
```

```
token_longitudinal   <- "DA6F2BB23146BD5A7EA3408C1A44A556"

# ---- Simple examples
REDCapR::redcap_project_info_read(uri, token_simple      )$data
REDCapR::redcap_project_info_read(uri, token_longitudinal)$data

# ---- Specify timezone
# Specify the server's timezone, for example, US Central
server_locale <- readr::locale(tz = "America/Chicago")
d3 <-
  REDCapR::redcap_project_info_read(
    uri,
    token_simple,
    locale     = server_locale
  )$data
d3$creation_time

# Alternatively, set timezone to the client's location.
client_locale <- readr::locale(tz = Sys.timezone())

# ---- Inspect multiple projects in the same tibble
# Stack all the projects on top of each other in a (nested) tibble,
#   starting from a csv of REDCapR test projects.
# The native pipes in this snippet require R 4.1+.
d_all <-
  system.file("misc/dev-2.credentials", package = "REDCapR") |>
  readr::read_csv(
    comment     = "#",
    col_select  = c(redcap_uri, token),
    col_types   = readr::cols(.default = readr::col_character())
  ) |>
  dplyr::filter(32L == nchar(token)) |>
  purrr::pmap_dfr(REDCapR::redcap_project_info_read, locale = server_locale)

# Inspect values stored on the server.
d_all$data
# or: View(d_all$data)

# Inspect everything returned, including values like the http status code.
d_all

## End(Not run)
```

---

redcap_read *Read records from a REDCap project in subsets, and stacks them together before returning a dataset*

Description

From an external perspective, this function is similar to redcap_read_oneshot(). The internals differ in that redcap_read retrieves subsets of the data, and then combines them before returning

(among other objects) a single `tibble::tibble()`. This function can be more appropriate than `redcap_read_oneshot()` when returning large datasets that could tie up the server.

Usage

```
redcap_read(
  batch_size = 100L,
  interbatch_delay = 0.5,
  continue_on_error = FALSE,
  redcap_uri,
  token,
  records = NULL,
  fields = NULL,
  forms = NULL,
  events = NULL,
  raw_or_label = "raw",
  raw_or_label_headers = "raw",
  export_checkbox_label = FALSE,
  export_survey_fields = FALSE,
  export_data_access_groups = FALSE,
  filter_logic = "",
  datetime_range_begin = as.POSIXct(NA),
  datetime_range_end = as.POSIXct(NA),
  blank_for_gray_form_status = FALSE,
  col_types = NULL,
  na = c("", "NA"),
  guess_type = TRUE,
  guess_max = NULL,
  http_response_encoding = "UTF-8",
  locale = readr::default_locale(),
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL,
  id_position = 1L
)
```

Arguments

`batch_size`	The maximum number of subject records a single batch should contain. The default is 100.
`interbatch_delay`	The number of seconds the function will wait before requesting a new subset from REDCap. The default is 0.5 seconds.
`continue_on_error`	If an error occurs while reading, should records in subsequent batches be attempted. The default is `FALSE`, which prevents subsequent batches from running. Required.
`redcap_uri`	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.

`token` The user-specific string that serves as the password for a project. Required.

`records` An array, where each element corresponds to the ID of a desired record. Optional.

`fields` An array, where each element corresponds to a desired project field. Optional.

`forms` An array, where each element corresponds to a desired project form. Optional.

`events` An array, where each element corresponds to a desired project event. Optional.

`raw_or_label` A string (either `'raw'` or `'label'` that specifies whether to export the raw coded values or the labels for the options of multiple choice fields. Default is `'raw'`.

`raw_or_label_headers`
A string (either `'raw'` or `'label'` that specifies for the CSV headers whether to export the variable/field names (raw) or the field labels (label). Default is `'raw'`.

`export_checkbox_label`
specifies the format of checkbox field values specifically when exporting the data as labels. If `raw_or_label` is `'label'` and `export_checkbox_label` is TRUE, the values will be the text displayed to the users. Otherwise, the values will be 0/1.

`export_survey_fields`
A boolean that specifies whether to export the survey identifier field (e.g., 'redcap_survey_identifier') or survey timestamp fields (e.g., instrument+'_timestamp'). The timestamp outputs reflect the survey's *completion* time (according to the time and timezone of the REDCap server.)

`export_data_access_groups`
A boolean value that specifies whether or not to export the `redcap_data_access_group` field when data access groups are utilized in the project. Default is `FALSE`. See the details below.

`filter_logic` String of logic text (e.g., `[gender] = 'male'`) for filtering the data to be returned by this API method, in which the API will only return the records (or record-events, if a longitudinal project) where the logic evaluates as TRUE. An blank/empty string returns all records.

`datetime_range_begin`
To return only records that have been created or modified *after* a given datetime, provide a POSIXct value. If not specified, REDCap will assume no begin time.

`datetime_range_end`
To return only records that have been created or modified *before* a given datetime, provide a POSIXct value. If not specified, REDCap will assume no end time.

`blank_for_gray_form_status`
A boolean value that specifies whether or not to export blank values for instrument complete status fields that have a gray status icon. All instrument complete status fields having a gray icon can be exported either as a blank value or as "0" (Incomplete). Blank values are recommended in a data export if the data will be re-imported into a REDCap project. Default is `FALSE`.

`col_types` A `readr::cols()` object passed internally to `readr::read_csv()`. Optional.

`na` A character vector passed internally to `readr::read_csv()`. Defaults to `c("", "NA")`.

guess_type — A boolean value indicating if all columns should be returned as character. If true, readr::read_csv() guesses the intended data type for each column. Ignored if col_types is not null.

guess_max — Deprecated.

http_response_encoding — The encoding value passed to httr::content(). Defaults to 'UTF-8'.

locale — a readr::locale() object to specify preferences like number, date, and time formats. This object is passed to readr::read_csv(). Defaults to readr::default_locale().

verbose — A boolean value indicating if messages should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.

config_options — A list of options passed to httr::POST(). See details at httr::httr_options(). Optional.

handle_httr — The value passed to the handle parameter of httr::POST(). This is useful for only unconventional authentication approaches. It should be NULL for most institutions. Optional.

id_position — The column position of the variable that unique identifies the subject (typically record_id). This defaults to the first variable in the dataset.

Value

Currently, a list is returned with the following elements:

- data: A tibble::tibble() of the desired records and columns.
- success: A boolean value indicating if the operation was apparently successful.
- status_codes: A collection of http status codes, separated by semicolons. There is one code for each batch attempted.
- outcome_messages: A collection of human readable strings indicating the operations' semicolons. There is one code for each batch attempted. In an unsuccessful operation, it should contain diagnostic information.
- records_collapsed: The desired records IDs, collapsed into a single string, separated by commas.
- fields_collapsed: The desired field names, collapsed into a single string, separated by commas.
- filter_logic: The filter statement passed as an argument.
- elapsed_seconds: The duration of the function.

If no records are retrieved (such as no records meet the filter criteria), a zero-row tibble is returned. Currently the empty tibble has zero columns, but that may change in the future.

Batching subsets of data

redcap_read() internally uses multiple calls to redcap_read_oneshot() to select and return data. Initially, only the primary key is queried through the REDCap API. The long list is then subsetted into batches, whose sizes are determined by the batch_size parameter. REDCap is then queried

for all variables of the subset's subjects. This is repeated for each subset, before returning a unified `tibble::tibble()`.

The function allows a delay between calls, which allows the server to attend to other users' requests (such as the users entering data in a browser). In other words, a delay between batches does not bog down the webserver when exporting/importing a large dataset.

A second benefit is less RAM is required on the webserver. Because each batch is smaller than the entire dataset, the webserver tackles more manageably sized objects in memory. Consider batching if you encounter the error:

```
ERROR: REDCap ran out of server memory. The request cannot be processed.
Please try importing/exporting a smaller amount of data.
```

A third benefit (compared to `redcap_read()`) is that important fields are included, even if not explicitly requested. As a result:

1. `record_id` (or it's customized name) will always be returned
2. `redcap_event_name` will be returned for longitudinal projects
3. `redcap_repeat_instrument` and `redcap_repeat_instance` will be returned for projects with repeating instruments

Export permissions

For `redcap_read_oneshot()` to function properly, the user must have Export permissions for the 'Full Data Set'. Users with only 'De-Identified' export privileges can still use `redcap_read_oneshot`. To grant the appropriate permissions:

- go to 'User Rights' in the REDCap project site,
- select the desired user, and then select 'Edit User Privileges',
- in the 'Data Exports' radio buttons, select 'Full Data Set'.

Pseudofields

The REDCap project may contain "pseudofields", depending on its structure. Pseudofields are exported for certain project structures, but are not defined by users and do not appear in the codebook. If a recognized pseudofield is passed to the `fields` api parameter, it is suppressed by `redcap_read()` and `redcap_read_oneshot()` so the server doesn't throw an error. Requesting a pseudofield is discouraged, so a message is returned to the user.

Pseudofields include:

- `redcap_event_name`: for longitudinal projects or multi-arm projects.
- `redcap_repeat_instrument`: for projects with repeating instruments.
- `redcap_repeat_instance`: for projects with repeating instruments.
- `redcap_data_access_group`: for projects with DAGs when the `export_data_access_groups` api parameter is TRUE.
- `redcap_survey_identifier`: for projects with surveys when the `export_survey_fields` api parameter is TRUE.

- *instrument_name*_timestamp: for projects with surveys. For example, an instrument called "demographics" will have a pseudofield named demographics_timestamp. REDCapR does not suppress requests for timestamps, so the server will throw an error like

  ERROR: The following values in the parameter fields are not valid: 'demographics_timestamp'

Events

The event argument is a vector of characters passed to the server. It is the "event-name", not the "event-label". The event-label is the value presented to the users, which contains uppercase letters and spaces, while the event-name can contain only lowercase letters, digits, and underscores.

If event is nonnull and the project is not longitudinal, redcap_read() will throw an error. Similarly, if a value in the event vector is not a current event-name, redcap_read() will throw an error.

The simpler redcap_read_oneshot() function does not check for invalid event values, and will not throw errors.

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
uri     <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token   <- "9A068C425B1341D69E83064A2D273A70"

# Return the entire dataset
REDCapR::redcap_read(batch_size=2, redcap_uri=uri, token=token)$data

# Return a subset of columns while also specifying the column types.
col_types <- readr::cols(
  record_id  = readr::col_integer(),
  race___1   = readr::col_logical(),
  race___2   = readr::col_logical(),
  race___3   = readr::col_logical(),
  race___4   = readr::col_logical(),
  race___5   = readr::col_logical(),
  race___6   = readr::col_logical()
)
REDCapR::redcap_read(
  redcap_uri = uri,
  token      = token,
  col_types  = col_types,
```

```
  batch_size = 2
)$data

## End(Not run)
```

`redcap_read_eav_oneshot` *Read/Export records from a REDCap project, returned as eav*

Description

This function uses REDCap's API to select and return data in an eav

Usage

```
redcap_read_eav_oneshot(
  redcap_uri,
  token,
  records = NULL,
  fields = NULL,
  forms = NULL,
  events = NULL,
  export_survey_fields = FALSE,
  export_data_access_groups = FALSE,
  filter_logic = "",
  datetime_range_begin = as.POSIXct(NA),
  datetime_range_end = as.POSIXct(NA),
  blank_for_gray_form_status = FALSE,
  http_response_encoding = "UTF-8",
  locale = readr::default_locale(),
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

`redcap_uri`	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
`token`	The user-specific string that serves as the password for a project. Required.
`records`	An array, where each element corresponds to the ID of a desired record. Optional.
`fields`	An array, where each element corresponds to a desired project field. Optional.
`forms`	An array, where each element corresponds to a desired project form. Optional.
`events`	An array, where each element corresponds to a desired project event. Optional.

`export_survey_fields`
: A boolean that specifies whether to export the survey identifier field (e.g., 'redcap_survey_identifier') or survey timestamp fields (e.g., instrument+'_timestamp'). The timestamp outputs reflect the survey's *completion* time (according to the time and timezone of the REDCap server.)

`export_data_access_groups`
: A boolean value that specifies whether or not to export the `redcap_data_access_group` field when data access groups are utilized in the project. Default is `FALSE`. See the details below.

`filter_logic`
: String of logic text (e.g., `[gender] = 'male'`) for filtering the data to be returned by this API method, in which the API will only return the records (or record-events, if a longitudinal project) where the logic evaluates as TRUE. An blank/empty string returns all records.

`datetime_range_begin`
: To return only records that have been created or modified *after* a given datetime, provide a POSIXct value. If not specified, REDCap will assume no begin time.

`datetime_range_end`
: To return only records that have been created or modified *before* a given datetime, provide a POSIXct value. If not specified, REDCap will assume no end time.

`blank_for_gray_form_status`
: A boolean value that specifies whether or not to export blank values for instrument complete status fields that have a gray status icon. All instrument complete status fields having a gray icon can be exported either as a blank value or as "0" (Incomplete). Blank values are recommended in a data export if the data will be re-imported into a REDCap project. Default is `FALSE`.

`http_response_encoding`
: The encoding value passed to `httr::content()`. Defaults to 'UTF-8'.

`locale`
: a `readr::locale()` object to specify preferences like number, date, and time formats. This object is passed to `readr::read_csv()`. Defaults to `readr::default_locale()`.

`verbose`
: A boolean value indicating if `messages` should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.

`config_options`
: A list of options passed to `httr::POST()`. See details at `httr::httr_options()`. Optional.

`handle_httr`
: The value passed to the `handle` parameter of `httr::POST()`. This is useful for only unconventional authentication approaches. It should be `NULL` for most institutions. Optional.

Details

If you do not pass an `export_data_access_groups` value, it will default to `FALSE`. The following is from the API help page for version 10.5.1: *This flag is only viable if the user whose token is being used to make the API request is* **not** *in a data access group. If the user is in a group, then this flag will revert to its default value.*

Value

Currently, a list is returned with the following elements:

- `data`: A `tibble::tibble()` of the desired records and columns.
- `success`: A boolean value indicating if the operation was apparently successful.
- `status_code`: The http status code of the operation.
- `outcome_message`: A human readable string indicating the operation's outcome.
- `records_collapsed`: The desired records IDs, collapsed into a single string, separated by commas.
- `fields_collapsed`: The desired field names, collapsed into a single string, separated by commas.
- `filter_logic`: The filter statement passed as an argument.
- `elapsed_seconds`: The duration of the function.
- `raw_text`: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the `raw_text` is returned as an empty string to save RAM.

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
uri      <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token    <- "9A068C425B1341D69E83064A2D273A70"

# Return all records and all variables.
ds <- REDCapR:::redcap_read_eav_oneshot(redcap_uri=uri, token=token)$data

# Return only records with IDs of 1 and 3
desired_records_v1 <- c(1, 3)
ds_some_rows_v1 <- REDCapR:::redcap_read_eav_oneshot(
  redcap_uri = uri,
  token      = token,
  records    = desired_records_v1
)$data

# Return only the fields record_id, name_first, and age
desired_fields_v1 <- c("record_id", "name_first", "age")
ds_some_fields_v1 <- REDCapR:::redcap_read_eav_oneshot(
  redcap_uri = uri,
```

```
  token      = token,
  fields     = desired_fields_v1
)$data

# Repeating
token <- "56F43A10D01D6578A46393394D76D88F" # PHI-free demo: Repeating Instruments --Sparse # 2603
ds <- REDCapR:::redcap_read_eav_oneshot(redcap_uri=uri, token=token)$data

## End(Not run)
```

---

redcap_read_oneshot *Read/Export records from a REDCap project*

Description

This function uses REDCap's API to select and return data.

Usage

```
redcap_read_oneshot(
  redcap_uri,
  token,
  records = NULL,
  fields = NULL,
  forms = NULL,
  events = NULL,
  raw_or_label = "raw",
  raw_or_label_headers = "raw",
  export_checkbox_label = FALSE,
  export_survey_fields = FALSE,
  export_data_access_groups = FALSE,
  filter_logic = "",
  datetime_range_begin = as.POSIXct(NA),
  datetime_range_end = as.POSIXct(NA),
  blank_for_gray_form_status = FALSE,
  col_types = NULL,
  na = c("", "NA"),
  guess_type = TRUE,
  guess_max = 1000,
  http_response_encoding = "UTF-8",
  locale = readr::default_locale(),
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

`redcap_uri`	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
`token`	The user-specific string that serves as the password for a project. Required.
`records`	An array, where each element corresponds to the ID of a desired record. Optional.
`fields`	An array, where each element corresponds to a desired project field. Optional.
`forms`	An array, where each element corresponds to a desired project form. Optional.
`events`	An array, where each element corresponds to a desired project event. Optional.
`raw_or_label`	A string (either `'raw'` or `'label'`) that specifies whether to export the raw coded values or the labels for the options of multiple choice fields. Default is `'raw'`.
`raw_or_label_headers`	A string (either `'raw'` or `'label'` that specifies for the CSV headers whether to export the variable/field names (raw) or the field labels (label). Default is `'raw'`.
`export_checkbox_label`	specifies the format of checkbox field values specifically when exporting the data as labels. If `raw_or_label` is `'label'` and `export_checkbox_label` is TRUE, the values will be the text displayed to the users. Otherwise, the values will be 0/1.
`export_survey_fields`	A boolean that specifies whether to export the survey identifier field (e.g., 'redcap_survey_identifier') or survey timestamp fields (e.g., instrument+'_timestamp'). The timestamp outputs reflect the survey's *completion* time (according to the time and timezone of the REDCap server.)
`export_data_access_groups`	A boolean value that specifies whether or not to export the `redcap_data_access_group` field when data access groups are utilized in the project. Default is `FALSE`. See the details below.
`filter_logic`	String of logic text (e.g., `[gender] = 'male'`) for filtering the data to be returned by this API method, in which the API will only return the records (or record-events, if a longitudinal project) where the logic evaluates as TRUE. An blank/empty string returns all records.
`datetime_range_begin`	To return only records that have been created or modified *after* a given datetime, provide a POSIXct value. If not specified, REDCap will assume no begin time.
`datetime_range_end`	To return only records that have been created or modified *before* a given datetime, provide a POSIXct value. If not specified, REDCap will assume no end time.
`blank_for_gray_form_status`	A boolean value that specifies whether or not to export blank values for instrument complete status fields that have a gray status icon. All instrument complete status fields having a gray icon can be exported either as a blank value or as "0" (Incomplete). Blank values are recommended in a data export if the data will be re-imported into a REDCap project. Default is `FALSE`.

col_types	A readr::cols() object passed internally to readr::read_csv(). Optional.
na	A character vector passed internally to readr::read_csv(). Defaults to c("", "NA").
guess_type	A boolean value indicating if all columns should be returned as character. If true, readr::read_csv() guesses the intended data type for each column. Ignored if col_types is not null.
guess_max	A positive base::numeric value passed to readr::read_csv() that specifies the maximum number of records to use for guessing column types.
http_response_encoding	The encoding value passed to httr::content(). Defaults to 'UTF-8'.
locale	a readr::locale() object to specify preferences like number, date, and time formats. This object is passed to readr::read_csv(). Defaults to readr::default_locale().
verbose	A boolean value indicating if messages should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
config_options	A list of options passed to httr::POST(). See details at httr::httr_options(). Optional.
handle_httr	The value passed to the handle parameter of httr::POST(). This is useful for only unconventional authentication approaches. It should be NULL for most institutions. Optional.

Details

If you do not pass an export_data_access_groups value, it will default to FALSE. The following is from the API help page for version 10.5.1: *This flag is only viable if the user whose token is being used to make the API request is* **not** *in a data access group. If the user is in a group, then this flag will revert to its default value*.

The REDCap project may contain "pseudofields", depending on its structure. Pseudofields are exported for certain project structures, but are not defined by users and do not appear in the codebook. If a recognized pseudofield is passed to the fields api parameter, it is suppressed by redcap_read() and redcap_read_oneshot() so the server doesn't throw an error. Requesting a pseudofield is discouraged, so a message is returned to the user.

Pseudofields include:

- redcap_event_name: for longitudinal projects or multi-arm projects.
- redcap_repeat_instrument: for projects with repeating instruments.
- redcap_repeat_instance: for projects with repeating instruments.
- redcap_data_access_group: for projects with DAGs when the export_data_access_groups api parameter is TRUE.
- redcap_survey_identifier: for projects with surveys when the export_survey_fields api parameter is TRUE.
- *instrument_name*_timestamp: for projects with surveys. For example, an instrument called "demographics" will have a pseudofield named demographics_timestamp. REDCapR does not suppress requests for timestamps, so the server will throw an error like

```
ERROR: The following values in the parameter fields are not valid: 'demographics_timestamp'
```

Value

Currently, a list is returned with the following elements:

- `data`: A `tibble::tibble()` of the desired records and columns.
- `success`: A boolean value indicating if the operation was apparently successful.
- `status_code`: The http status code of the operation.
- `outcome_message`: A human readable string indicating the operation's outcome.
- `records_collapsed`: The desired records IDs, collapsed into a single string, separated by commas.
- `fields_collapsed`: The desired field names, collapsed into a single string, separated by commas.
- `filter_logic`: The filter statement passed as an argument.
- `elapsed_seconds`: The duration of the function.
- `raw_text`: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the `raw_text` is returned as an empty string to save RAM.

If no records are retrieved (such as no records meet the filter criteria), a zero-row tibble is returned. Currently the empty tibble has zero columns, but that may change in the future.

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
uri      <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token    <- "9A068C425B1341D69E83064A2D273A70"

# Return all records and all variables.
ds <- REDCapR::redcap_read_oneshot(redcap_uri=uri, token=token)$data

# Return only records with IDs of 1 and 3
desired_records_v1 <- c(1, 3)
ds_some_rows_v1 <- REDCapR::redcap_read_oneshot(
  redcap_uri = uri,
  token      = token,
  records    = desired_records_v1
)$data

# Return only the fields record_id, name_first, and age
```

```
desired_fields_v1 <- c("record_id", "name_first", "age")
ds_some_fields_v1 <- REDCapR::redcap_read_oneshot(
  redcap_uri = uri,
  token      = token,
  fields     = desired_fields_v1
)$data

# Specify the column types.
col_types <- readr::cols(
  record_id  = readr::col_integer(),
  race___1   = readr::col_logical(),
  race___2   = readr::col_logical(),
  race___3   = readr::col_logical(),
  race___4   = readr::col_logical(),
  race___5   = readr::col_logical(),
  race___6   = readr::col_logical()
)
ds_col_types <- REDCapR::redcap_read_oneshot(
  redcap_uri = uri,
  token      = token,
  col_types  = col_types
)$data

## End(Not run)
```

---

redcap_read_oneshot_eav

Read/Export records from a REDCap project –still in development

---

Description

This function uses REDCap's API to select and return data. This function is still in development.

Usage

```
redcap_read_oneshot_eav(
  redcap_uri,
  token,
  records = NULL,
  fields = NULL,
  forms = NULL,
  events = NULL,
  raw_or_label = "raw",
  raw_or_label_headers = "raw",
  export_data_access_groups = FALSE,
  filter_logic = "",
  datetime_range_begin = as.POSIXct(NA),
  datetime_range_end = as.POSIXct(NA),
  blank_for_gray_form_status = FALSE,
```

```
  http_response_encoding = "UTF-8",
  locale = readr::default_locale(),
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

redcap_uri	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
token	The user-specific string that serves as the password for a project. Required.
records	An array, where each element corresponds to the ID of a desired record. Optional.
fields	An array, where each element corresponds to a desired project field. Optional.
forms	An array, where each element corresponds to a desired project field. Optional.
events	An array, where each element corresponds to a desired project event. Optional.
raw_or_label	A string (either 'raw' or 'label' that specifies whether to export the raw coded values or the labels for the options of multiple choice fields. Default is 'raw'.
raw_or_label_headers	A string (either 'raw' or 'label' that specifies for the CSV headers whether to export the variable/field names (raw) or the field labels (label). Default is 'raw'.
export_data_access_groups	A boolean value that specifies whether or not to export the redcap_data_access_group field when data access groups are utilized in the project. Default is FALSE. See the details below.
filter_logic	String of logic text (e.g., [gender] = 'male') for filtering the data to be returned by this API method, in which the API will only return the records (or record-events, if a longitudinal project) where the logic evaluates as TRUE. An blank/empty string returns all records.
datetime_range_begin	To return only records that have been created or modified *after* a given datetime, provide a POSIXct (https://stat.ethz.ch/R-manual/R-devel/library/base/html/as.POSIXlt.html) value. If not specified, REDCap will assume no begin time.
datetime_range_end	To return only records that have been created or modified *before* a given datetime, provide a POSIXct (https://stat.ethz.ch/R-manual/R-devel/library/base/html/as.POSIXlt.html) value. If not specified, REDCap will assume no end time.
blank_for_gray_form_status	A boolean value that specifies whether or not to export blank values for instrument complete status fields that have a gray status icon. All instrument complete status fields having a gray icon can be exported either as a blank value or as "0" (Incomplete). Blank values are recommended in a data export if the data will be re-imported into a REDCap project. Default is FALSE.
http_response_encoding	The encoding value passed to `httr::content()`. Defaults to 'UTF-8'.

locale	a readr::locale() object to specify preferences like number, date, and time formats. This object is passed to readr::read_csv(). Defaults to readr::default_locale().
verbose	A boolean value indicating if messages should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
config_options	A list of options passed to httr::POST(). See details at httr::httr_options(). Optional.
handle_httr	The value passed to the handle parameter of httr::POST(). This is useful for only unconventional authentication approaches. It should be NULL for most institutions. Optional.

Details

If you do not pass in this export_data_access_groups value, it will default to FALSE. The following is from the API help page for version 5.2.3: This flag is only viable if the user whose token is being used to make the API request is *not* in a data access group. If the user is in a group, then this flag will revert to its default value.

As of REDCap 6.14.3, this field is not exported in the EAV API call.

Value

Currently, a list is returned with the following elements:

- data: A tibble::tibble() of the desired records and columns.
- success: A boolean value indicating if the operation was apparently successful.
- status_code: The http status code of the operation.
- outcome_message: A human readable string indicating the operation's outcome.
- records_collapsed: The desired records IDs, collapsed into a single string, separated by commas.
- fields_collapsed: The desired field names, collapsed into a single string, separated by commas.
- filter_logic: The filter statement passed as an argument.
- elapsed_seconds: The duration of the function.
- raw_text: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the raw_text is returned as an empty string to save RAM.

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
uri      <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token    <- "9A068C425B1341D69E83064A2D273A70"

# Return all records and all variables.
ds <- REDCapR:::redcap_read_oneshot_eav(redcap_uri=uri, token=token)$data

# Return only records with IDs of 1 and 3
desired_records_v1 <- c(1, 3)
ds_some_rows_v1    <- REDCapR:::redcap_read_oneshot_eav(
   redcap_uri = uri,
   token      = token,
   records    = desired_records_v1
)$data

# Return only the fields record_id, name_first, and age
desired_fields_v1 <- c("record_id", "name_first", "age")
ds_some_fields_v1 <- REDCapR:::redcap_read_oneshot_eav(
   redcap_uri = uri,
   token      = token,
   fields     = desired_fields_v1
)$data

## End(Not run)
```

`redcap_report` *Read/Export records that populate a REDCap report*

Description

Exports the data set of a report created on a project's 'Data Exports, Reports, and Stats' page.

Usage

```
redcap_report(
  redcap_uri,
  token,
  report_id,
  raw_or_label = "raw",
  raw_or_label_headers = "raw",
  export_checkbox_label = FALSE,
  col_types = NULL,
  guess_type = TRUE,
  guess_max = 1000,
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

redcap_uri	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
token	The user-specific string that serves as the password for a project. Required.
report_id	A single integer, provided next to the report name on the report list page. Required.
raw_or_label	A string (either 'raw' or 'label') that specifies whether to export the raw coded values or the labels for the options of multiple choice fields. Default is 'raw'.
raw_or_label_headers	A string (either 'raw' or 'label' that specifies for the CSV headers whether to export the variable/field names (raw) or the field labels (label). Default is 'raw'.
export_checkbox_label	specifies the format of checkbox field values specifically when exporting the data as labels. If raw_or_label is 'label' and export_checkbox_label is TRUE, the values will be the text displayed to the users. Otherwise, the values will be 0/1.
col_types	A readr::cols() object passed internally to readr::read_csv(). Optional.
guess_type	A boolean value indicating if all columns should be returned as character. If true, readr::read_csv() guesses the intended data type for each column. Ignored if col_types is not null.
guess_max	A positive base::numeric value passed to readr::read_csv() that specifies the maximum number of records to use for guessing column types.
verbose	A boolean value indicating if messages should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
config_options	A list of options passed to httr::POST(). See details at httr::httr_options(). Optional.
handle_httr	The value passed to the handle parameter of httr::POST(). This is useful for only unconventional authentication approaches. It should be NULL for most institutions. Optional.

Value

Currently, a list is returned with the following elements:

- data: A tibble::tibble() of the desired records and columns.
- success: A boolean value indicating if the operation was apparently successful.
- status_code: The http status code of the operation.
- outcome_message: A human readable string indicating the operation's outcome.
- elapsed_seconds: The duration of the function.
- raw_text: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the raw_text is returned as an empty string to save RAM.

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
uri          <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token        <- "9A068C425B1341D69E83064A2D273A70"
report_1_id  <- 12L
report_2_id  <- 13L

# Return all records and all variables.
ds_1a <-
  REDCapR::redcap_report(
    redcap_uri = uri,
    token      = token,
    report_id  = report_1_id
  )$data


# Specify the column types.
col_types_1 <- readr::cols(
  record_id          = readr::col_integer(),
  height             = readr::col_double(),
  health_complete    = readr::col_integer(),
  address            = readr::col_character(),
  ethnicity          = readr::col_integer()
)
ds_1b <-
  REDCapR::redcap_report(
    redcap_uri = uri,
    token      = token,
    report_id  = report_1_id,
    col_types  = col_types_1
  )$data


# Return condensed checkboxes Report option:
#   "Combine checkbox options into single column of only the checked-off
#   options (will be formatted as a text field when exported to
#   stats packages)"
col_types_2 <- readr::cols(
  record_id          = readr::col_integer(),
  race               = readr::col_character()
)
```

```
ds_2 <-
  REDCapR::redcap_report(
    redcap_uri = uri,
    token      = token,
    report_id  = report_2_id,
    col_types  = col_types_2
  )$data

## End(Not run)
```

`redcap_survey_link_export_oneshot` *Get survey link from REDCap*

Description

This function uses REDCap's API to get the link for a survey.

Usage

```
redcap_survey_link_export_oneshot(
  redcap_uri,
  token,
  record,
  instrument,
  event = "",
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

`redcap_uri`	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
`token`	The user-specific string that serves as the password for a project. Required.
`record`	The record ID associated with the survey link. Required
`instrument`	The name of the instrument associated with the survey link. Required
`event`	The name of the event associated with the survey link. Optional
`verbose`	A boolean value indicating if `messages` should be printed to the R console during the operation. Optional.
`config_options`	A list of options passed to `httr::POST()`. See details at `httr::httr_options()`. Optional.
`handle_httr`	The value passed to the `handle` parameter of `httr::POST()`. This is useful for only unconventional authentication approaches. It should be `NULL` for most institutions. Optional.

Details

Currently, the function doesn't modify any variable types to conform to REDCap's supported variables. See `validate_for_write()` for a helper function that checks for some common important conflicts.

Permissions Required To use this method, you must have API Export privileges in the project. (As stated in the 9.0.0 documentation.)

Value

Currently, a list is returned with the following elements,

- `survey_link`: a character string containing the URL for the survey.
- `success`: A boolean value indicating if the operation was apparently successful.
- `status_code`: The http status code of the operation.
- `outcome_message`: A human readable string indicating the operation's outcome.
- `instrument`: The instrument associated with the survey link.
- `records_affected_count`: The number of records associated with the survey link.
- `affected_ids`: The subject IDs associated with the survey link.
- `elapsed_seconds`: The duration of the function.
- `raw_text`: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the `raw_text` is returned as an empty string to save RAM.

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
uri     <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token   <- "4780D038A2080BA2E7CC904A14218662" # survey
record  <- 1
instrument   <- "participant_morale_questionnaire"
# event <- "" # only for longitudinal projects

result <- REDCapR::redcap_survey_link_export_oneshot(
  record         = record,
  instrument     = instrument,
  redcap_uri     = uri,
  token          = token
```

```
)
result$survey_link

## End(Not run)
```

`redcap_users_export` *List authorized users*

Description

List users authorized for a project.

Usage

```
redcap_users_export(
  redcap_uri,
  token,
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

`redcap_uri`	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
`token`	The user-specific string that serves as the password for a project. Required.
`verbose`	A boolean value indicating if `messages` should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
`config_options`	A list of options passed to `httr::POST()`. See details at `httr::httr_options()`. Optional.
`handle_httr`	The value passed to the `handle` parameter of `httr::POST()`. This is useful for only unconventional authentication approaches. It should be `NULL` for most institutions. Optional.

Value

- `data_user`: A `tibble::tibble()` of all users associated with the project. One row represents one user.
- `data_user_form`: A `tibble::tibble()` of permissions for users and forms. One row represents a unique user-by-form combination.
- `success`: A boolean value indicating if the operation was apparently successful.
- `status_codes`: A collection of http status codes, separated by semicolons. There is one code for each batch attempted.

- outcome_messages: A collection of human readable strings indicating the operations' semicolons. There is one code for each batch attempted. In an unsuccessful operation, it should contain diagnostic information.
- elapsed_seconds: The duration of the function.

Note

Documentation in REDCap 8.4.0

```
This method allows you to export the list of users for a project,
including their user privileges and also email address, first name,
and last name.

Note: If the user has been assigned to a user role, it will return
the user with the role's defined privileges.
```

Examples

```
## Not run:
uri      <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token    <- "0BF920AAF9566A8E603F528A498A5729" # dag
result   <- REDCapR::redcap_users_export(redcap_uri=uri, token=token)
result$data_user
result$data_user_form

## End(Not run)
```

---

redcap_variables *Enumerate the exported variables*

Description

This function calls the 'exportFieldNames' function of the REDCap API.

Usage

```
redcap_variables(
  redcap_uri,
  token,
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

redcap_uri	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
token	The user-specific string that serves as the password for a project. Required.
verbose	A boolean value indicating if messages should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
config_options	A list of options passed to httr::POST(). See details at httr::httr_options(). Optional.
handle_httr	The value passed to the handle parameter of httr::POST(). This is useful for only unconventional authentication approaches. It should be NULL for most institutions. Optional.

Details

As of REDCap version 6.14.2, three variable types are *not* returned in this call: calculated, file, and descriptive. All variables returned are writable/uploadable.

Value

Currently, a list is returned with the following elements,

- data: A tibble::tibble() where each row represents one column in the REDCap dataset.
- success: A boolean value indicating if the operation was apparently successful.
- status_code: The http status code of the operation.
- outcome_message: A human readable string indicating the operation's outcome.
- elapsed_seconds: The duration of the function.
- raw_text: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the raw_text is returned as an empty string to save RAM.

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
uri         <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token       <- "9A068C425B1341D69E83064A2D273A70"
ds_variable <- REDCapR::redcap_variables(redcap_uri=uri, token=token)$data
```

```
## End(Not run)
```

`redcap_version` *Determine version of REDCap instance*

Description

This function uses REDCap's API to query its version.

Usage

```
redcap_version(
  redcap_uri,
  token,
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

`redcap_uri`	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
`token`	The user-specific string that serves as the password for a project. Required.
`verbose`	A boolean value indicating if `messages` should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
`config_options`	A list of options passed to `httr::POST()`. See details at `httr::httr_options()`. Optional.
`handle_httr`	The value passed to the `handle` parameter of `httr::POST()`. This is useful for only unconventional authentication approaches. It should be `NULL` for most institutions. Optional.

Details

If the API call is unsuccessful, a value of `base::package_version("0.0.0")` will be returned. This ensures that a the function will always return an object of class base::numeric_version. It guarantees the value can always be used in `utils::compareVersion()`.

Value

a utils::packageDescription

Examples

```
uri      <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token    <- "9A068C425B1341D69E83064A2D273A70"
REDCapR::redcap_version(redcap_uri = uri, token = token)
```

`redcap_write` *Write/Import records to a REDCap project*

Description

This function uses REDCap's APIs to select and return data.

Usage

```
redcap_write(
  ds_to_write,
  batch_size = 100L,
  interbatch_delay = 0.5,
  continue_on_error = FALSE,
  redcap_uri,
  token,
  overwrite_with_blanks = TRUE,
  convert_logical_to_integer = FALSE,
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

`ds_to_write`	The `base::data.frame()` or `tibble::tibble()` to be imported into the REDCap project. Required.
`batch_size`	The maximum number of subject records a single batch should contain. The default is 100.
`interbatch_delay`	The number of seconds the function will wait before requesting a new subset from REDCap. The default is 0.5 seconds.
`continue_on_error`	If an error occurs while writing, should records in subsequent batches be attempted. The default is `FALSE`, which prevents subsequent batches from running. Required.
`redcap_uri`	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
`token`	The user-specific string that serves as the password for a project. Required.

`overwrite_with_blanks`
: A boolean value indicating if blank/NA values in the R data frame will overwrite data on the server. This is the default behavior for REDCapR, which essentially deletes the cell's value If FALSE, blank/NA values in the data.frame will be ignored. Optional.

`convert_logical_to_integer`
: If TRUE, all base::logical columns in ds are cast to an integer before uploading to REDCap. Boolean values are typically represented as 0/1 in REDCap radio buttons. Optional.

`verbose`
: A boolean value indicating if messages should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.

`config_options`
: A list of options passed to `httr::POST()`. See details at `httr::httr_options()`. Optional.

`handle_httr`
: The value passed to the handle parameter of `httr::POST()`. This is useful for only unconventional authentication approaches. It should be NULL for most institutions. Optional.

Details

Currently, the function doesn't modify any variable types to conform to REDCap's supported variables. See `validate_for_write()` for a helper function that checks for some common important conflicts.

For `redcap_write` to function properly, the user must have Export permissions for the 'Full Data Set'. Users with only 'De-Identified' export privileges can still use `redcap_write_oneshot()`. To grant the appropriate permissions:

- go to 'User Rights' in the REDCap project site,
- select the desired user, and then select 'Edit User Privileges',
- in the 'Data Exports' radio buttons, select 'Full Data Set'.

Value

Currently, a list is returned with the following elements:

- `success`: A boolean value indicating if the operation was apparently successful.
- `status_code`: The http status code of the operation.
- `outcome_message`: A human readable string indicating the operation's outcome.
- `records_affected_count`: The number of records inserted or updated.
- `affected_ids`: The subject IDs of the inserted or updated records.
- `elapsed_seconds`: The duration of the function.

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
if (FALSE) {
# Define some constants
uri           <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token         <- "F9CBFFF78C3D78F641BAE9623F6B7E6A" # simple-write

# Read the dataset for the first time.
result_read1  <- REDCapR::redcap_read_oneshot(redcap_uri=uri, token=token)
ds1           <- result_read1$data
ds1$telephone

# Manipulate a field in the dataset in a VALID way
ds1$telephone <- paste0("(405) 321-000", seq_len(nrow(ds1)))

ds1 <- ds1[1:3, ]
ds1$age       <- NULL; ds1$bmi <- NULL # Drop the calculated fields before writing.
result_write  <- REDCapR::redcap_write(ds1, redcap_uri=uri, token=token)

# Read the dataset for the second time.
result_read2  <- REDCapR::redcap_read_oneshot(redcap_uri=uri, token=token)
ds2           <- result_read2$data
ds2$telephone

# Manipulate a field in the dataset in an INVALID way.  A US exchange can't be '111'.
ds1$telephone <- paste0("(405) 321-000", seq_len(nrow(ds1)))

# This next line will throw an error.
result_write <- REDCapR::redcap_write(ds1, redcap_uri=uri, token=token)
result_write$raw_text
}
```

`redcap_write_oneshot` *Write/Import records to a REDCap project*

Description

This function uses REDCap's API to select and return data.

Usage

```
redcap_write_oneshot(
  ds,
```

```
  redcap_uri,
  token,
  overwrite_with_blanks = TRUE,
  convert_logical_to_integer = FALSE,
  verbose = TRUE,
  config_options = NULL,
  handle_httr = NULL
)
```

Arguments

ds	The base::data.frame() or tibble::tibble() to be imported into the REDCap project. Required.
redcap_uri	The uri/url of the REDCap server typically formatted as "https://server.org/apps/redcap/api/". Required.
token	The user-specific string that serves as the password for a project. Required.
overwrite_with_blanks	A boolean value indicating if blank/NA values in the R data frame will overwrite data on the server. This is the default behavior for REDCapR, which essentially deletes the cell's value If FALSE, blank/NA values in the data frame will be ignored. Optional.
convert_logical_to_integer	If TRUE, all base::logical columns in ds are cast to an integer before uploading to REDCap. Boolean values are typically represented as 0/1 in REDCap radio buttons. Optional.
verbose	A boolean value indicating if messages should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
config_options	A list of options passed to httr::POST(). See details at httr::httr_options(). Optional.
handle_httr	The value passed to the handle parameter of httr::POST(). This is useful for only unconventional authentication approaches. It should be NULL for most institutions. Optional.

Details

Currently, the function doesn't modify any variable types to conform to REDCap's supported variables. See validate_for_write() for a helper function that checks for some common important conflicts.

Value

Currently, a list is returned with the following elements:

- success: A boolean value indicating if the operation was apparently successful.
- status_code: The http status code of the operation.
- outcome_message: A human readable string indicating the operation's outcome.

- `records_affected_count`: The number of records inserted or updated.
- `affected_ids`: The subject IDs of the inserted or updated records.
- `elapsed_seconds`: The duration of the function.
- `raw_text`: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the `raw_text` is returned as an empty string to save RAM.

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
if (FALSE) {
# Define some constants
uri            <- "https://redcap-dev-2.ouhsc.edu/redcap/api/"
token          <- "F9CBFFF78C3D78F641BAE9623F6B7E6A" # simple-write

# Read the dataset for the first time.
result_read1   <- REDCapR::redcap_read_oneshot(redcap_uri=uri, token=token)
ds1            <- result_read1$data
ds1$telephone

# Manipulate a field in the dataset in a VALID way
ds1$telephone  <- paste0("(405) 321-000", seq_len(nrow(ds1)))

ds1 <- ds1[1:3, ]
ds1$age        <- NULL; ds1$bmi <- NULL # Drop the calculated fields before writing.
result_write   <- REDCapR::redcap_write_oneshot(ds=ds1, redcap_uri=uri, token=token)

# Read the dataset for the second time.
result_read2   <- REDCapR::redcap_read_oneshot(redcap_uri=uri, token=token)
ds2            <- result_read2$data
ds2$telephone

# Manipulate a field in the dataset in an INVALID way.  A US exchange can't be '111'.
ds1$telephone  <- paste0("(405) 321-000", seq_len(nrow(ds1)))

# This next line will throw an error.
result_write   <- REDCapR::redcap_write_oneshot(ds=ds1, redcap_uri=uri, token=token)
result_write$raw_text
}
```

`replace_nas_with_explicit` *Create explicit factor level for missing values*

Description

Missing values are converted to a factor level. This explicit assignment can reduce the chances that missing values are inadvertently ignored. It also allows the presence of a missing to become a predictor in models.

Usage

```
replace_nas_with_explicit(
  scores,
  new_na_label = "Unknown",
  create_factor = FALSE,
  add_unknown_level = FALSE
)
```

Arguments

`scores`	An array of values, ideally either factor or character. Required
`new_na_label`	The factor label assigned to the missing value. Defaults to `Unknown`.
`create_factor`	Converts `scores` into a factor, if it isn't one already. Defaults to `FALSE`.
`add_unknown_level`	Should a new factor level be created? (Specify `TRUE` if it already exists.) Defaults to `FALSE`.

Value

An array of values, where the `NA` values are now a factor level, with the label specified by the `new_na_label` value.

Note

The `create_factor` parameter is respected only if `scores` isn't already a factor. Otherwise, levels without any values would be lost.

A `stop` error will be thrown if the operation fails to convert all the `NA` values.

Author(s)

Will Beasley

Examples

```
library(REDCapR) # Load the package into the current R session.
```

`retrieve_credential` *Read a token and other credentials from a (non-REDCap) database or file*

Description

These functions are not essential to calling the REDCap API, but instead are functions that help manage tokens securely.

Usage

```
retrieve_credential_local(
  path_credential,
  project_id,
  check_url            = TRUE,
  check_username       = FALSE,
  check_token_pattern  = TRUE,
  username             = NA_character_
)

retrieve_credential_mssql(
  project_id,
  instance,
  dsn,
  channel    = NULL
)

create_credential_local(
  path_credential
)
```

Arguments

`path_credential`	The file path to the CSV containing the credentials. Required.
`project_id`	The ID assigned to the project withing REDCap. This allows the user to store tokens to multiple REDCap projects in one file. Required
`check_url`	A `logical` value indicates if the url in the credential file should be checked to have approximately the correct form. Defaults to TRUE. `retrieve_credential_local()`.
`check_username`	A `logical` value indicates if the username in the credential file should be checked against the username returned by R. Defaults to FALSE.
`check_token_pattern`	A `logical` value indicates if the token in the credential file is a 32-character hexadecimal string. Defaults to FALSE.
`username`	A character value used to retrieve a credential. See the Notes below. Optional.

`instance`	The casual name associated with the REDCap instance on campus. This allows one credential system to accommodate multiple instances on campus. Required
`dsn`	A DSN on the local machine that points to the desired MSSQL database. Required.
`channel`	An *optional* connection handle as returned by `DBI::dbConnect()`. See Details below. Optional.

Details

If the database elements are created with the script provided in package's 'Security Database' vignette, the default values will work.

The `create_credential_local()` function copies a static file to the location specified in the `path_credential` argument. Each record represents one accessible project per user. Follow these steps to adapt to your desired REDCap project(s):

1. Modify the credential file for the REDCap API with a text editor like Notepad++, Visual Studio Code, or nano. Replace existing records with the information from your projects. Delete the remaining example records.
2. Make sure that the file (with the sensitive password-like) tokens is stored securely!
3. Contact your REDCap admin to request the URI & token and discuss institutional policies.
4. Ask your institution's IT security team for their recommendation
5. For more info, see https://ouhscbbmc.github.io/REDCapR/articles/workflow-read.html and https://ouhscbbmc.github.io
6. Double-check the file is secured and not accessible by other users.

Value

A list of the following elements are returned from `retrieve_credential_local()` and `retrieve_credential_mssql()`:

- `redcap_uri`: The URI of the REDCap Server.
- `username`: Username.
- `project_id`: The ID assigned to the project within REDCap.
- `token`: The token to pass to the REDCap server
- `comment`: An optional string that is ignored by REDCapR but can be helpful for humans.

Note

Storing credentials on a server is preferred

Although we strongly encourage storing all the tokens on a central server (*e.g.*, see the `retrieve_credential_mssql()` function and the "SecurityDatabase" vignette), there are times when this approach is not feasible and the token must be stored locally. Please contact us if your institution is using something other than SQL Server, and would like help adapting this approach to your infrastructure.

Storing credentials locally

When storing credentials locally, typically the credential file should be dedicated to just one user. Occasionally it makes sense to store tokens for multiple users –usually it's for the purpose of testing.

The `username` field is connected only in the local credential file. It does not need to be the same as the official username in REDCap.

Author(s)

Will Beasley

Examples

```
# ---- Local File Example ----------------------------
path <- system.file("misc/dev-2.credentials", package = "REDCapR")
(p1  <- REDCapR::retrieve_credential_local(path, 33L))
(p2  <- REDCapR::retrieve_credential_local(path, 34L))


## Not run:
# Create a skeleton of the local credential file to modify
path_demo <- base::tempfile(pattern = "temp", fileext = ".credentials")

create_credential_local(path_demo)

base::unlink(path_demo) # This is just a demo; don't delete the real file!

## End(Not run)
```

---

sanitize_token *Validate and sanitize the user's REDCap token*

Description

Verifies the token is nonmissing and conforms to the legal pattern of a 32-character hexadecimal value. Each character must be an (a) digit 0-9, (b) uppercase letter A-F, or (c) lowercase letter a-f. Trailing line endings are removed.

A typical user does not call this function directly. However functions like `redcap_read()` call it to provide a more informative error message to the user.

Some institutions create their own tokens –not the standard 32-character hexadecimal value. The pattern that validates their tokens can be specified with the system environmental variable `REDCAP_TOKEN_PATTERN` using base::Sys.setenv.

For example, the following regex pattern captures a base64 encoded value with 40 characters (as opposed to a hexadecimal/base16 value with 32 characters): `^([A-Za-z\\d+/\\+=]{40})$`. See `https://rgxdb.com/r/1NUN74O6` or `https://regex101.com/library/lXFWqM` for alternative approaches to validate base64 values.

If no pattern is specified, the default is a 32-character hex token: `^([0-9A-Fa-f]{32})(?:\\n)?$`. The important segment is contained in the first (and only) capturing group (*i.e.*, `[0-9A-Fa-f]{32}`). Any trailing newline character is removed.

Usage

```
sanitize_token(token)
```

Arguments

token	The REDCap token. Required.

Details

Although the function does not accept a parameter, it is influenced by the REDCAP_TOKEN_PATTERN environmental variable.

Value

The token, without a terminal newline character.

Note

Contact your institution's REDCap administrator for more information about your project-specific token.

Author(s)

Hao Zhu, Benjamin Nutter, Will Beasley, Jordan Mark Barbone

Examples

```
secret_token_1 <- "12345678901234567890123456ABCDEF"
secret_token_2 <- "12345678901234567890123456ABCDEF\n"
secret_token_3 <- "12345678901234567890123456abcdef"
REDCapR::sanitize_token(secret_token_1)
REDCapR::sanitize_token(secret_token_2)
REDCapR::sanitize_token(secret_token_3)

# Some institutions use a token system that follows a different pattern
Sys.setenv("REDCAP_TOKEN_PATTERN" = "^([A-Za-z\\d+/\\+=]{10})$")

secret_token_4 <- "abcde1234="
REDCapR::sanitize_token(secret_token_4)
Sys.getenv("REDCAP_TOKEN_PATTERN")
Sys.unsetenv("REDCAP_TOKEN_PATTERN")
```

---

to_api_array *Convert a vector to the array format expected by the REDCap API*

Description

Utility function to convert a vector into the array format expected by the some REDCap API calls. It is called internally by REDCapR functions, and is not intended to be called directly.

Usage

```
to_api_array(x, element_names)
```

Arguments

x	A vector to convert to array format. Can be NULL.
element_names	A string containing the name of the API request parameter for the array. Must be either "fields" or "forms".

Value

If x is not NULL a list is returned with one element for each element of x in the format: `list(`element_names[0]` = x[1], `element_names[1]` = x[2], ...)`.

If x is NULL then NULL is returned.

---

validate *Inspect a dataset to anticipate problems before writing to a REDCap project*

Description

This set of functions inspect a data frame to anticipate problems before writing with REDCap's API.

Usage

```
validate_for_write( d, convert_logical_to_integer, record_id_name )

validate_data_frame_inherits( d )

validate_no_logical( d, stop_on_error = FALSE )

validate_field_names( d, stop_on_error = FALSE )

validate_record_id_name( d, record_id_name = "record_id", stop_on_error = FALSE )

validate_repeat_instance( d, stop_on_error = FALSE )

validate_uniqueness( d, record_id_name, stop_on_error = FALSE)
```

Arguments

d	The base::data.frame() or tibble::tibble() containing the dataset used to update the REDCap project.
record_id_name	The name of the field that represents one record. The default name in REDCap is "record_id".
stop_on_error	If TRUE, an error is thrown for violations. Otherwise, a dataset summarizing the problems is returned.
convert_logical_to_integer	This mimics the convert_logical_to_integer parameter in redcap_write() when checking for potential importing problems. Defaults to FALSE.

Details

All functions listed in the Usage section above inspect a specific aspect of the dataset. The `validate_for_write()` function executes all these individual validation checks. It allows the client to check everything with one call.

Currently, the individual checks include:

1. `validate_data_frame_inherits(d)`: `d` inherits from `base::data.frame()`
2. `validate_field_names(d)`: The columns of `d` start with a lowercase letter, and subsequent optional characters are a sequence of (a) lowercase letters, (b) digits 0-9, and/or (c) underscores. (The exact regex is `^[a-z][0-9a-z_]*$`.)
3. `validate_record_id_name(d)`: `d` contains a field called "record_id", or whatever value was passed to `record_id_name`.
4. `validate_no_logical(d)` (unless `convert_logical_to_integer` is TRUE): `d` does not contain logical values (because REDCap typically wants `0`/`1` values instead of `FALSE`/`TRUE`).
5. `validate_repeat_instance(d)`: `d` has an integer for `redcap_repeat_instance`, if the column is present.
6. `validate_uniqueness(d, record_id_name = record_id_name)`: `d` does not contain multiple rows with duplicate values of `record_id`, `redcap_event_name`, `redcap_repeat_instrument`, and `redcap_repeat_instance` (depending on the longitudinal & repeating structure of the project).

   Technically duplicate rows are not errors, but we feel that this will almost always be unintentional, and lead to an irrecoverable corruption of the data.

If you encounter additional types of problems when attempting to write to REDCap, please tell us by creating a new issue, and we'll incorporate a new validation check into this function.

Value

A `tibble::tibble()`, where each potential violation is a row. The two columns are:

- `field_name`: The name of the field/column/variable that might cause problems during the upload.
- `field_index`: The position of the field. (For example, a value of '1' indicates the first column, while a '3' indicates the third column.)
- `concern`: A description of the problem potentially caused by the `field`.
- `suggestion`: A *potential* solution to the concern.

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
d1 <- data.frame(
  record_id      = 1:4,
  flag_logical   = c(TRUE, TRUE, FALSE, TRUE),
  flag_Uppercase = c(4, 6, 8, 2)
)
REDCapR::validate_for_write(d = d1)

REDCapR::validate_for_write(d = d1, convert_logical_to_integer = TRUE)

# If `d1` is not a data.frame, the remaining validation checks are skipped:
# REDCapR::validate_for_write(as.matrix(mtcars))
# REDCapR::validate_for_write(c(mtcars, iris))

d2 <- tibble::tribble(
  ~record_id, ~redcap_event_name, ~redcap_repeat_instrument, ~redcap_repeat_instance,
  1L, "e1", "i1", 1L,
  1L, "e1", "i1", 2L,
  1L, "e1", "i1", 3L,
  1L, "e1", "i1", 4L,
  1L, "e1", "i2", 1L,
  1L, "e1", "i2", 2L,
  1L, "e1", "i2", 3L,
  1L, "e1", "i2", 4L,
  2L, "e1", "i1", 1L,
  2L, "e1", "i1", 2L,
  2L, "e1", "i1", 3L,
  2L, "e1", "i1", 4L,
)
validate_uniqueness(d2)
validate_for_write(d2)

d3 <- tibble::tribble(
  ~record_id, ~redcap_event_name, ~redcap_repeat_instrument, ~redcap_repeat_instance,
  1L, "e1", "i1", 1L,
  1L, "e1", "i1", 3L,
  1L, "e1", "i1", 3L, # Notice this duplicates the row above
)
# validate_uniqueness(d3)
# Throws error:
# validate_uniqueness(d3, stop_on_error = TRUE)
```

Index