Package: certeprojects (via r-universe)

October 31, 2024

Title A Certe R Package for Department Projects

Version 1.21.8

Description A Certe R Package with functions to automate department projects using MS Teams, MS Planner, Shiny, R Markdown and Quarto. This package is part of the 'certedata' universe.

URL https://certe-medical-epidemiology.github.io/certeprojects, https://github.com/certe-medical-epidemiology/certeprojects

Depends R (>= 4.1.0)

Imports certestyle, callr (>= 3.7.0), AzureAuth (>= 1.3.0), AzureGraph (>= 1.3.0), dplyr (>= 1.0.0), httr (>= 1.4.0), jsonlite (>= 1.7.2), Microsoft365R (>= 2.4.0), miniUI (>= 0.1.1), pins (>= 1.1.0), quarto (>= 1.2.0), rstudioapi (>= 0.10), shiny (>= 1.6.0), shinyjs (>= 2.0.0), shinyWidgets (>= 0.6.2), yaml (>= 2.2.0)

Suggests certetoolbox, certemail, clipr (>= 0.8.0), lubridate (>= 1.9.0), msgxtractr, rlang (>= 1.1.0), rmarkdown, rvest, testthat (>= 2.0.0)

License GPL-2

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Config/testthat/edition 2

Repository https://certe-medical-epidemiology.r-universe.dev

RemoteUrl https://github.com/certe-medical-epidemiology/certeprojects

RemoteRef HEAD

RemoteSha ddc127cdc93ccb97d0f8c6bfba3562d9b29b9e99

Contents

`connect` *Connect to Microsoft 365*

Description

These functions create a connection to Microsoft 365 and saves the connection to the `certeprojects` package environment. The `planner_*()` and `teams_*()` functions allow to work with these connections.

Usage

```
get_microsoft365_token(
  scope = read_secret("azure.scope_list"),
  tenant = read_secret("azure.tenant"),
  app_id = read_secret("azure.app_id"),
  auth_type = read_secret("azure.auth_type"),
  ...,
  overwrite = TRUE,
  error_on_fail = TRUE
)

connect_outlook(email = read_secret("department.mail"), overwrite = FALSE, ...)

connect_planner(
  plan = read_secret("planner.name"),
  team_name = read_secret("team.name"),
  overwrite = FALSE,
  ...
)

connect_teams(team_name = read_secret("team.name"), overwrite = FALSE, ...)
```

Arguments

scope	any (combination) of "outlook", "teams", "planner", "tasks" (which is "planner" without group rights), or "sharepoint". Can also be a manual vector of Microsoft Permissions, such as "User.Read.
tenant	the tenant to use, passed on to AzureGraph::create_graph_login()
app_id	the Azure app id to use, passed on to AzureGraph::create_graph_login()
auth_type	the authentication method to use, passed on to AzureGraph::create_graph_login()
...	arguments passed on to get_microsoft365_token()
overwrite	a logical to overwrite an existing connection, useful for switching accounts
error_on_fail	a logical to indicate whether an error must be thrown if no connection can be made
email	email address of the user, or a shared mailbox
plan	name of the team's plan if team_name is not empty. Otherwise, a plan ID (for individual use).
team_name	name of the team, can be left blank to connect to an individual planner

Details

Microsoft Outlook:

To switch between different Outlook accounts, run connect_outlook() with another email address, and set overwrite = TRUE. This will allow all certemail functions to use the newly set account.

```
# at default connects to the department mailbox:
connect_outlook()
# afterwards, this does nothing since `overwrite` is not set:
connect_outlook("user@certe.nl")
# this switches to the user account:
connect_outlook("user@certe.nl", overwrite = TRUE)
```

Using overwrite is needed, because running just connect_outlook() afterwards again (which many certemail functions do) will otherwise change back to the default account.

Microsoft Planner:

To connect to MS Planner with a personal account, retrieve the plan ID (e.g., from the URL of the plan), and pass it on to connect_planner() as plan, and set overwrite = TRUE over replace an existing connection. Make sure that team_name is left blank:

```
connect_planner(plan = "AAA-0aa0AaAa-aaaAAAAAAAAa", team_name = NULL, overwrite = TRUE)
```

Microsoft Teams:

Connecting to MS Teams can only be done on the group (= team) level. It is not possible to set up a connection without a valid team name.

Pre-loaded Settings:

When attaching this certeprojects package using library(), and external R process will be run in the background using the callr package to connect to MS Outlook, MS Planner, and MS Teams. This will increase speed when connecting using connect_outlook(), connect_planner(), or connect_teams().

get_azure_property *Get Azure Property*

Description

This function retrieves a property from an Azure object, such as ms_plan, ms_plan_task, ms_team, ms_team_member, ms_drive_item.

Usage

```
get_azure_property(x, property)
```

Arguments

x	an Azure object
property	the name of the property, such as "id" or "displayName". This must exist in x or in x$properties, and will return NA otherwise.

knit *Knit R Markdown or Quarto Document*

Description

Allows Quarto to work with full paths (where Quarto itself requires relative paths) and replaces both rmarkdown::render() and quarto::quarto_render().

Usage

```
knit(input_file, output_file = NULL, quiet = TRUE, as_job = "auto", ...)

render(input_file, output_file = NULL, quiet = TRUE, as_job = "auto", ...)
```

Arguments

input_file	file to be rendered, can be R Markdown (.Rmd) or Quarto (.qmd), or a lot of other formats such as .md, .ipynb and many other formats that Quarto supports.
output_file	The name of the output file. If using NULL, the output filename will be based on the filename for the input file. output_file is mapped to the --output option flag of the quarto CLI. It is expected to be a filename only, not a path, relative or absolute.
quiet	Suppress warning and other messages.
as_job	Render as an RStudio background job. Default is "auto", which will render individual documents normally and projects as background jobs. Use the quarto.render_as_job R option to control the default globally.
...	other arguments passed on to quarto::quarto_render().

Details

Functions knit() and render() are identical.

---

pins *Work with Pins*

Description

These functions can be used to work with pins, developed by RStudio.

Usage

```
export_pin(
  x,
  name = NULL,
  title = NULL,
  type = NULL,
  description = NULL,
  board = pins_board()
)

import_pin(name, version = NULL, hash = NULL, board = pins_board())

remove_pin(name, version = NULL, board = pins_board())

pins_board(
  projects_channel_id = read_secret("teams.projects.channel_id"),
  account = connect_teams(),
  delete_by_item = TRUE
)
```

Arguments

x	An object (typically a data frame) to pin.
name	Pin name.
title	A title for the pin; most important for shared boards so that others can understand what the pin contains. If omitted, a brief description of the contents will be automatically generated.
type	File type used to save x to disk. Must be one of "csv", "json", "rds", "parquet", "arrow", or "qs". If not supplied, will use JSON for bare lists and RDS for everything else. Be aware that CSV and JSON are plain text formats, while RDS, Parquet, Arrow, and qs are binary formats.
description	A detailed description of the pin contents.
board	A pin board, created by board_folder(), board_connect(), board_url() or another board_ function.

version	Retrieve a specific version of a pin. Use pin_versions() to find out which versions are available and when they were created.
hash	Specify a hash to verify that you get exactly the dataset that you expect. You can find the hash of an existing pin by looking for pin_hash in pin_meta().
projects_channel_id	Teams channel ID of the projects
account	a Microsoft 365 account to use for looking up properties. This has to be an object as returned by connect_teams() or Microsoft365R::get_team().
delete_by_item	Whether to handle folder deletions on an item-by-item basis, rather than deleting the entire folder at once. You may need to set this to TRUE for a board in SharePoint Online or OneDrive for Business, due to document protection policies that prohibit deleting non-empty folders.

Details

These functions from the pins package are integrated into the team's Microsoft 365 account, using the "pins" folder in the given MS Teams channel.

For Pins functions of the pins package, use pins_board() as input, e.g.:

```
pin_list(pins_board())
```

The following pins functions are re-exported by this package: pin_list(), pin_meta(), and pin_versions().

The pins_board() function returns a pins::board_ms365 object based on the "pins" folder in the Teams channel *Projects*, which is retrieved with teams_projects_channel().

---

planner *Connect to Microsoft Planner via Microsoft 365*

Description

These functions use the connection to Microsoft Planner set up with connect_planner().

Usage

```
planner_browse(account = connect_planner())

planner_bucket_create(bucket_name, account = connect_planner())

planner_buckets_list(account = connect_planner(), plain = FALSE)

planner_task_create(
  title,
  description = NULL,
  startdate = NULL,
```

```
  duedate = NULL,
  requested_by = NULL,
  priority = read_secret("planner.default.priority"),
  checklist_items = NULL,
  categories = NULL,
  assigned = NULL,
  bucket_name = read_secret("planner.default.bucket"),
  attachment_urls = NULL,
  account = connect_planner(),
  project_number = planner_highest_project_id() + 1,
  consult_number = planner_highest_consult_id() + 1
)

planner_task_update(
  task,
  title = NULL,
  description = NULL,
  startdate = NULL,
  duedate = NULL,
  priority = NULL,
  checklist_items = NULL,
  categories = NULL,
  categories_keep = FALSE,
  assigned = NULL,
  assigned_keep = FALSE,
  bucket_name = NULL,
  percent_completed = NULL,
  attachment_urls = NULL,
  account = connect_planner()
)

planner_tasks_list(
  account = connect_planner(),
  plain = FALSE,
  include_completed = TRUE
)

planner_task_search(
  search_term = ".*",
  limit = 50,
  include_completed = TRUE,
  include_description = FALSE,
  account = connect_planner()
)

planner_task_find(task, account = connect_planner())

planner_categories_list(account = connect_planner())
```

```
planner_retrieve_project_id(task, account = connect_planner())

planner_task_request_validation(
  task,
  category_text = read_secret("planner.label.authorise"),
  account = connect_planner()
)

planner_task_validate(
  task,
  category_text = read_secret("planner.label.authorised"),
  account = connect_planner()
)

planner_create_project_from_path(
  path,
  projects_path = read_secret("projects.path"),
  account = connect_planner(),
  title = basename(path),
  ...
)

planner_user_property(
  user,
  team_name = read_secret("team.name"),
  account = connect_planner(),
  property = "id",
  as_list = FALSE
)

planner_highest_project_id(
  task = read_secret("planner.dummy.project.id"),
  account = connect_planner()
)

planner_highest_consult_id(
  task = read_secret("planner.dummy.consult.id"),
  account = connect_planner()
)

## S3 method for class 'ms_object'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  account = connect_planner(),
  ...
```

```
)

## S3 method for class 'ms_object'
as_tibble(x, account = connect_planner(), ...)
```

Arguments

account	a Microsoft 365 account to use for looking up properties. This has to be an object as returned by connect_planner() or via AzureGraph::create_graph_login()$get_group(name)$g
bucket_name	name of the bucket
plain	return as plain names, not as Azure objects
title	title of the task
description	a description for the task. A vector of length > 1 will be added as one text separated by white lines.
startdate	a date to use as start date, use FALSE to remove it
duedate	a date to use as due date, use FALSE to remove it
requested_by	name of the person(s) who requested the task, this will be added as first line to description
priority	a priority to set. Can be ranged between 0 (highest) and 10 (lowest), or: "urgent" or "dringend" for 1, "important" or "belangrijk" for 3, "medium" or "gemiddeld" or FALSE for 5, "low" or "laag" for 9. Priorities cannot be removed - the default setting is 5.
checklist_items	character vector of checklist items
categories	names of categories to add, can be multiple, but must exactly match existing category names
assigned	names of members within the plan - use NULL to not add members in planner_task_create(), and use FALSE to remove all existing members in planner_task_update()
attachment_urls	URLs to add as attachment, can be named characters to give the URLs a title. If they are Excel, PowerPoint or Word files, a preview will be shown on the task.
project_number	the new project number to assign. Use NULL or FALSE to not assign a project number. Defaults to the currently highest project ID + 1.
consult_number	the new consult number to assign. Use NULL or FALSE to not assign a consult number. Defaults to the currently highest consult ID + 1.
task	exact task ID or title, will be searched with %like%
categories_keep	add categories that are set in categories instead of replacing them, defaults to FALSE
assigned_keep	add members that are set in assigned instead of replacing them, defaults to FALSE
percent_completed	percentage of task completion between 0-100

include_completed
: also search completed tasks

search_term
: search term, can contain a regular expression. When searching for project numbers (such as "p201 - Some text", or "p201" or "201"), only titles will be searched for the project number.

limit
: maximum number of tasks to show

include_description
: also search the description, which requires additional queries and lowers speed

category_text
: text of the category to use

path
: location of the folder that has to be converted to a project. This folder will be renamed to contain the new project number.

projects_path
: location of the folder that contains all department projects

...
: arguments passed on to planner_task_create()

user
: a user name, mail adress, or Certe login name

team_name
: name of the team, can be left blank to connect to an individual planner

property
: property to return, can be "id", "name" or "mail"

as_list
: return the full list of members as list, split into Eigenaars (Owners) / Leden (Members). This ignores user.

x
: an ms_object

row.names
: NULL or a character vector giving the row names for the data frame. Missing values are not allowed.

optional
: logical. If TRUE, setting row names and converting column names (to syntactic names: see make.names) is optional. Note that all of R's **base** package as.data.frame() methods use optional only for column names treatment, basically with the meaning of data.frame(*, check.names = !optional). See also the make.names argument of the matrix method.

Details

planner_task_search() searches the title and description using case-insensitive regular expressions and returns an ms_plan_task object. In interactive mode and with multiple hits, a menu will be shown to pick from.

planner_task_find() searches task title or ID, and returns an ms_plan_task object. It is used internally b a lot of planner_* functions, very fast, and does not support interactive use.

planner_retrieve_project_id() retrieves the p-number from the task title and returns it as integer.

Use planner_create_project_from_path() to convert any folder (and any location) to a project folder, by (1) assigning a project number, (2) creating a Planner task and (3) moving the old folder to the department's projects folder.

planner_highest_project_id() retrieves the currently highest project ID from the dummy project.

planner_highest_project_id() retrieves the currently highest project ID from the dummy project. planner_highest_consult_id() does this for consults.

Using as.data.frame() or as_tibble() on an ms_object, such as ms_plan_task, will return the properties and details of the object as a data.frame. For transforming many ms_objects to a data.frame, use as.data.frame() or as_tibble() in lapply() and bind the list of objects together. For example, this retrieves a tibble with the properties and details of all tasks:

```
library(dplyr)
planner_tasks_list() |>
  lapply(as_tibble) |>
  bind_rows()

# also works for other 'ms_object's, such as 'ms_channel':
teams_channels_list(plain = FALSE) |>
  lapply(as_tibble) |>
  bind_rows()
```

`project_add` *Add Project Or Consult Using Shiny*

Description

This is a Shiny app to add a new project: it creates a project folder locally or in Teams, generates the required Quarto or R Markdown or R file, and creates a new task in Planner. These functions come with RStudio addins to quickly access existing projects. For consults, it only adds a Planner task and creates the card in the background.

Usage

```
project_add(planner = connect_planner(), teams = NULL, channel = NULL)

project_update(
  current_task_id = project_get_current_id(ask = TRUE),
  planner = connect_planner()
)

consult_add(planner = connect_planner(), teams = NULL, channel = NULL)
```

Arguments

planner	Microsoft Planner account, as returned by e.g. connect_planner()
teams	Microsoft Teams account, as returned by e.g. connect_teams()
channel	Microsoft Teams Channel folder, as returned by e.g. teams_projects_channel()
current_task_id	Project (p-)number of the project to update

project_properties *Project Properties*

Description

Retrieve project properties, such as the title, folder location and project number.

Usage

```
project_get_current_id(ask = NULL, account = connect_planner())

project_identifier(project_number = project_get_current_id())

project_get_folder(
  project_number = project_get_current_id(),
  account = connect_planner()
)

project_get_folder_full(
  project_number = project_get_current_id(),
  projects_path = read_secret("projects.path"),
  account = connect_planner()
)

project_get_title(
  project_number = project_get_current_id(),
  account = connect_planner()
)

project_get_file(
  filename = ".*",
  project_number = project_get_current_id(),
  fixed = FALSE,
  account = connect_planner()
)

project_set_file(
  filename,
  project_number = project_get_current_id(),
  account = connect_planner()
)

project_set_folder(
  foldername,
  project_number = project_get_current_id(),
  account = connect_planner()
)
```

```
project_open_analysis_file(
  project_number = project_get_current_id(ask = TRUE),
  account = connect_planner()
)

project_open_folder(
  project_number = project_get_current_id(ask = TRUE),
  account = connect_planner()
)

project_add_qmd_skeleton(
  filename = NULL,
  project_number = project_get_current_id(),
  account = connect_planner()
)
```

Arguments

ask	logical to indicate whether the project number should always be asked. The default, NULL, will show a popup in interactive R sessions, allowing to search for projects. In non-interactive sessions, such as in Quarto and R Markdown, it will use the current working directory to determine the project number.
account	a Microsoft 365 account to use for looking up properties. This has to be an object as returned by connect_planner() or via AzureGraph::create_graph_login()$get_group(name)$g
project_number	Planner project number
projects_path	location of the folder that contains all department projects
filename	name for the new Quarto file
fixed	logical to turn off regular expressions
foldername	foldername to set

Details

project_get_current_id() uses planner_task_search() to find a specific project based on any search string.

project_identifier() generates the project identifier for print on reports and in mails: a combination of the currently logged in user (in your case: 'root'), the current date/time (format: YYMMDDHHMM), and the project number. If the project number is not available, it will only return the current user and date/time (format: YYMMDDHHMM).

project_set_folder() will create the folder if it does not exist.

project_add_qmd_skeleton() initializes a new Quarto skeleton for a project.

Examples

```
project_identifier(123)
```

schedule_task *Schedule Task (CRON-like)*

Description

This will source a project file if time and user requirements are met, using a CRON-like syntax (https://cron.help).

Usage

```
schedule_task(
  minute,
  hour,
  day,
  month,
  weekday,
  users,
  file,
  project_number,
  log = TRUE,
  ref_time = Sys.time(),
  account = connect_planner(),
  check_mail = length(users) > 1,
  check_log = length(users) > 1,
  sent_delay = 15,
  sent_account = connect_outlook(),
  sent_to = read_secret("mail.error_to"),
  log_folder = read_secret("projects.log_path")
)
```

Arguments

minute	one or more values between 0-59, or . or "*" for each minute
hour	one or more values between 0-23, or . or "*" for each hour
day	one or more values between 1-31, or . or "*" for each day
month	one or more values between 1-12, or . or "*" for each mpnth
weekday	one or more values between 0-7 (Sunday is both 0 and 7; Monday is 1), or . or "*" for each weekday
users	logged in users, must correspond with Sys.info()["user"]. Currently logged in user is "root". This must be length > 1 if check_mail is TRUE.
file	file name within the project, supports regular expression
project_number	number of the project, must be numeric and exist in planner_tasks_list()
log	a logical to indicate whether this message should be printed: *Running scheduled task at...*

ref_time	time to use for reference, defaults to Sys.time()
account	Planner account
check_mail	a logical to indicate whether a project was sent by a previous user, by running certemail::mail_is_sent()
check_log	a logical to indicate whether a log file exist for the project from a previous user
sent_delay	delay in minutes. This will be multiplied by the position of the current user in users minus 1. For example, when sent_delay = 15, this will be 15 for user 2, and 30 for user 3.
sent_account	Outlook account, to search sent mails
sent_to	users to send error mail to
log_folder	path that contains log files

Details

The Windows Task Scheduler must be set up to use this function. Most convenient is to:

1. Create an R file such as R_cron.R with calls to schedule_task()
2. Create a batch file such as R_cron.bat that runs R_cron.R with R CMD BATCH
3. Set up a Task Scheduler task that runs R_cron.bat every minute

Examples

```
something_to_run <- function() {
  1 + 1
}

# units:      M  H  d  m  wd
schedule_task(., ., ., ., ., "user", "file", 123) # every minute
schedule_task(0, ., ., ., ., "user", "file", 123) # start of each hour
schedule_task(0, 7, ., ., ., "user", "file", 123) # everyday at 7h00
schedule_task(0, 7, 1, ., ., "user", "file", 123) # first day of month at 7h00
schedule_task(0, 7, ., 2, ., "user", "file", 123) # everyday day in February at 7h00
schedule_task(0, 7, ., ., 1, "user", "file", 123) # every Monday at 7h00
schedule_task(0, 7, 1, 2, ., "user", "file", 123) # every 1st of February at 7h00
schedule_task(0, 7, ., 2, 1, "user", "file", 123) # every Monday in February at 7h00
schedule_task(0, 7, 1, 2, 1, "user", "file", 123) # each February 1st if it's a Monday at 7h00
schedule_task(0, 7,29, 2, ., "user", "file", 123) # once every 4 years at 7h00

# examples of combinations

# everyday at 7h00 and 7h30
schedule_task(c(0, 30), 7,        .,       .,             ., "user", "file", 123)
# everyday at 7h00 and 15h00
schedule_task(0,        c(7, 15), .,       .,             ., "user", "file", 123)
# everyday at 7h00 and 7h30 and 15h00 and 15h30
schedule_task(c(0, 30), c(7, 15), .,       .,             ., "user", "file", 123)
# every second Monday of the month at 7h00:
schedule_task(0,        7,        c(8:14), .,             1, "user", "file", 123)
# every 15th of April at 8h30 and 16h30:
```

```
schedule_task(30,       c(8, 16), 15,      4,              ., "user", "file", 123)
# once per quarter at 8h00 on the first day of the month:
schedule_task(0,        8,        1,       c(1, 4, 7, 10), ., "user", "file", 123)

# fall-back for failed jobs

# this will run at 8h00 if current user is "user1"
schedule_task(0, 8, ., ., ., c("user1", "user2"), "file", 123)
# it will run again at default 15 minutes later (so, 8h15), if:
# - current user is "user2"
# - project 123 has no mail in Sent Items or log of "user1" contains errors
```

---

teams *Connect to Microsoft Teams via Microsoft 365*

Description

These functions use the connection to Microsoft Teams set up with `connect_teams()`.

Usage

```
teams_projects_channel(
  projects_channel_id = read_secret("teams.projects.channel_id"),
  overwrite = FALSE,
  account = connect_teams()
)

teams_new_project(
  task,
  channel = teams_projects_channel(),
  planner = connect_planner()
)

teams_browse_project(
  task,
  channel = teams_projects_channel(),
  planner = connect_planner()
)

teams_list_project_files(
  task,
  channel = teams_projects_channel(),
  planner = connect_planner()
)

teams_download_project_file(
  file,
  task,
```

```
  channel = teams_projects_channel(),
  planner = connect_planner()
)

teams_get_project_file(
  file,
  task,
  channel = teams_projects_channel(),
  planner = connect_planner()
)

teams_open_project_analysis_file(
  task,
  channel = teams_projects_channel(),
  planner = connect_planner()
)

teams_render_project_file(
  file,
  task,
  output_file = NULL,
  fun = rmarkdown::render,
  ...,
  channel = teams_projects_channel(),
  planner = connect_planner()
)

teams_view_project_file(
  file,
  task,
  channel = teams_projects_channel(),
  planner = connect_planner()
)

teams_upload_project_file(
  files,
  task,
  channel = teams_projects_channel(),
  planner = connect_planner()
)

teams_download_file(
  full_teams_path = NULL,
  account = connect_teams(),
  destination_dir = getwd(),
  overwrite = FALSE
)
```

```
teams_download_folder(
  full_teams_path = NULL,
  account = connect_teams(),
  destination_dir = getwd(),
  recursive = TRUE,
  overwrite = FALSE
)

teams_upload_file(
  file_path,
  full_teams_path = NULL,
  account = connect_teams(),
  file_name = NULL
)

teams_upload_folder(
  folder_path,
  full_teams_path = NULL,
  account = connect_teams(),
  recursive = TRUE
)

pick_teams_item(
  full_teams_path = NULL,
  account = connect_teams(),
  only_folders = FALSE
)

teams_name(account = connect_teams())

teams_channels_list(account = connect_teams(), plain = TRUE)

teams_view_sharepoint(channel, account = connect_teams())

teams_send_message(
  body,
  channel,
  content_type = c("text", "html"),
  attachments = NULL,
  account = connect_teams()
)

teams_open(teams_path, channel = NULL, account = connect_teams())

teams_get_link(
  teams_path,
  share_type = c("view", "edit"),
  expire_after = "1 month",
```

```
  password = NULL,
  channel = NULL,
  account = connect_teams()
)
```

Arguments

projects_channel_id	Teams channel ID of the projects
overwrite	a logical to overwrite an existing connection, useful for switching accounts
account	a Microsoft 365 account to use for looking up properties. This has to be an object as returned by connect_teams() or Microsoft365R::get_team().
task	any task title, task ID, or ms_plan_task object (e.g. from planner_task_find())
channel	a Teams folder object. This has to be an object as returned by teams_projects_channel().
planner	a Microsoft 365 account for Planner. This has to be an object as returned by connect_planner().
file	the file name to open
output_file	path of the output file
fun	function to use for rendering. Can be e.g. rmarkdown::render or quarto::quarto_render.
...	arguments passed on to fun
files	the files to upload
full_teams_path	a full path in Teams, **including the Team name and the channel name**. Leave blank to use interactive mode, which allows file/folder picking from a list in the console.
destination_dir	a folder to download the file or folder to, defaults to the current working directory.
recursive	download/upload all files within the folder
file_path	local path of the file to upload. Can also be an R object to save it as RDS to Teams.
file_name	a file name to use if file_path is an R object
folder_path	local path of the folder to upload
only_folders	only show folders, not files
plain	return as plain names, not as Azure objects
body	text of the message
content_type	type of content, must be "text" or "html"
attachments	vector of file locations of attachments to add to the message
teams_path	file location in Microsoft Teams, may also contain the channel name if channel is NULL, e.g., teams_path = "channel name/test.xlsx"
share_type	type of share, must be "view" (default) or "edit"
expire_after	time span after which the share link expires, defaults to "1 month", can also be e.g. "7 days"
password	password to set for share link, defaults to blank

Details

The `teams_new_project()` function:

1. Checks if there is a Planner task with the correct task title
2. Creates a new folder in Teams in the projects channel
3. Updates the task to contain the project folder URL as an attachment

The `teams_download_project_file()` function will download the given project file to a temporary location, and will return the path of this location. This makes it possible to use `source()`, `rmarkdown::render()` or `quarto::quarto_render()` using the `teams_download_project_file()` function as input.

The `teams_render_project_file()` function allows to render a Teams file. It downloads the Teams file using `teams_download_project_file()`, runs the rendering function set in `fun`, and uploads the resulting output file back to Teams using the same file name a `file`, but with the new file extension (such as pdf, html, or docx). It **invisibly returns** the temporary local file location, so that the output of `teams_render_project_file()` can be given to e.g. `certemail::mail()` as an attachment.

The `teams_download_file()` and `teams_download_folder()` functions use `pick_teams_item()` to select a file or folder, after which they will be downloaded to the destination folder.

The `teams_upload_file()` and `teams_upload_folder()` functions use `pick_teams_item()` to select the destination folder on Teams. **Notice** that these upload functions have not `overwrite` argument - Microsoft365R does not support them since overwrite means that a new file version will be created on Teams.

The `pick_teams_item()` function provides an interactive way to select a file in any Team, any channel. It returns a list with the properties `group_id`, `is_private`, `channel_id`, `item_id`, `item_name`, `item_path`, and `full_path` of the Team item.

`teams_send_message()` can also take a data.frame, which will be converted to HTML with plain_html_table(). If the input is a vector length > 1, the input will be collapsed with linebreaks.

Examples

```
## Not run:
# PROJECT-RELATED -----------------------------------------------------

# Project-related Teams function rely on existing Planner tasks.

# create a new project, which will be a folder in the Teams channel
# for this, the task 'My Planner task' must already exist
teams_new_project("My Planner task")

# the task 'My Planner task' will now contain the URL to the project

# upload a file there
teams_upload_project_file("analysis.Rmd", "My Planner task")

# render R markdown or Quarto from and to the cloud
teams_render_project_file("analysis.Rmd", "My Planner task")
# this will put the output file in the same Teams folder as 'analysis.Rmd'
```

```
teams_open("test.xlsx", "My Channel")
teams_open("my channel/test.xlsx") # shorter version, tries to find channel


# PROJECT-UNRELATED ------------------------------------------------

# by not specifying a remote location, a file picker will show in the console:
teams_download_file()
teams_download_folder("MyTeamName/MyChannelName/MySubFolder/")

teams_upload_file("myfile.docx", full_teams_path = "MyTeamName/MyChannelName/MySubFolder/")

# also supports data frames, they will be saved as RDS
mtcars |>
  teams_upload_file("MyTeamName/MyChannelName/MySubFolder/")

## End(Not run)
```

Index