Package ‘RSuite’

June 10, 2019

Type   Package
Title  Supports Developing, Building and Deploying R Solution
Version 0.37-253
Maintainer Walerian Sokolowski <rsuite@wlogsolutions.com>
Description Supports safe and reproducible solutions development in R. It will help you with environment separation per project, dependency management, local packages creation and preparing deployment packs for your solutions.
URL https://rsuite.io
BugReports https://github.com/WLOGSolutions/RSuite/issues
License MIT + file LICENSE
Encoding UTF-8
Depends R (>= 3.2.0)
Imports logging, methods, devtools, roxygen2, git2r, jsonlite, processx, httr
Suggests knitr, rmarkdown, testthat
VignetteBuilder knitr
RoxygenNote 6.1.1
NeedsCompilation no
Author Walerian Sokolowski [aut, cre],
       Wit Jakuczun [aut],
       Yulia Yakimechko [aut],
       Mateusz Kalinowski [aut],
       Ryszard Szymanski [aut],
       Alfonso Reyes [ctb],
       WLOG Solutions [cph],
       R Consortium [ctb] (Definitions of system requirements are taken from sysreqsdb project.)
Repository CRAN
Date/Publication 2019-06-10 14:20:02 UTC
R topics documented:

build_bash_script ........................................... 3
build_win_script ............................................. 4
ci_adapter_create_base ...................................... 5
ci_adapter_get_version ....................................... 5
ci_adapter_is_building ...................................... 6
get_version_numbers ......................................... 7
inst_wrap_zip .................................................. 8
perform ......................................................... 8
pkgzip_build_bioc_package .................................. 9
pkgzip_build_ext_packages .................................. 10
pkgzip_build_github_package ................................. 12
pkgzip_build_package_files ................................. 13
pkgzip_build_prj_packages .................................. 14
prj_build ...................................................... 16
prj_clean_deps ............................................... 17
prj_config_set_repo_adapters ............................... 18
prj_config_set_rversion ..................................... 19
prj_init ....................................................... 20
prj_install_deps .............................................. 21
prj_load ....................................................... 22
prj_lock_env .................................................. 23
prj_pack ....................................................... 24
prj_start ....................................................... 26
prj_start_package ............................................ 27
prj_unload .................................................... 28
prj_unlock_env ............................................... 29
prj_zip ......................................................... 30
rc_adapter_create_base ...................................... 31
rc_adapter_get_version ...................................... 32
rc_adapter_is_under_control ................................ 33
rc_adapter_pkg_struct_add .................................. 34
rc_adapter_prj_struct_add .................................. 35
rc_adapter_remove_admins ................................... 35
repo_adapter_create_base .................................... 36
repo_adapter_create_manager ................................. 37
repo_adapter_get_info ....................................... 38
repo_adapter_get_path ....................................... 39
repo_manager_destroy ....................................... 40
repo_manager_get_info ....................................... 41
repo_manager_init .......................................... 42
repo_manager_remove ....................................... 43
repo_manager_upload ........................................ 44
repo_mng_init ................................................ 45
repo_mng_list ............................................... 46
repo_mng_remove ............................................ 47
repo_mng_start .............................................. 49
**build_bash_script**

- repo_mng_stop .................................................. 50
- repo_upload_bio_package ...................................... 51
- repo_upload_ext_packages ...................................... 52
- repo_upload_github_package ................................... 54
- repo_upload_package_files ..................................... 55
- repo_upload_pkgzip ............................................. 56
- repo_upload_prj_packages ...................................... 58
- RSuite ............................................................. 59
- rsuite_check_version .......................................... 63
- rsuite_getLogger ................................................ 64
- rsuite_get_ci_adapter_names ................................... 65
- rsuite_get_os_info .............................................. 65
- rsuite_get_rc_adapter_names .................................. 66
- rsuite_get_repo_adapter_names ................................. 67
- rsuite_register_ci_adapter ..................................... 67
- rsuite_register_rc_adapter ..................................... 68
- rsuite_register_repo_adapter .................................. 69
- rsuite_unregister_ci_adapter ................................... 70
- rsuite_unregister_rc_adapter ................................... 70
- rsuite_unregister_repo_adapter ................................. 71
- rsuite_update .................................................... 72
- sysreqs_check ................................................... 73
- sysreqs_collect ................................................ 74
- sysreqs_install ................................................ 75
- sysreqs_script .................................................. 76
- tmpl_list_registered ........................................... 77
- tmpl_register .................................................... 77
- tmpl_start ........................................................ 79

**Index**

---

**build_bash_script**  
*Creates a bash script to update the system to satisfy project requirements.*

**Description**

Creates a bash script to update the system to satisfy project requirements.

**Usage**

```
build_bash_script(recipe, plat_desc)
```
**Arguments**

- **recipe**: object of type `sysreqs_script_recipe`
- **plat_desc**: named list content:
  - **name**: One of Windows, MacOS, RedHat, Debian (type: character)
  - **distrib**: Distribution e.g. for Debian: Debian, Ubuntu; for RedHat: CentOS, RedHat, Fedora (type: character(1))
  - **release**: Distribution release e.g. for Debian: squeeze, wheezy, jessie (type: character(1))
  - **sysreq_type**: One of Windows, Pkg, RPM, DEB (type: character(1))
  - **build**: True if build environment is required (type: logical(1))

---

**Description**

Creates a cmd script to update the system to satisfy project requirements.

**Usage**

```r
build_win_script(recipe, plat_desc)
```

**Arguments**

- **recipe**: object of type `sysreqs_script_recipe`
- **plat_desc**: named list content:
  - **name**: One of Windows, MacOS, RedHat, Debian (type: character)
  - **distrib**: Distribution e.g. for Debian: Debian, Ubuntu; for RedHat: CentOS, RedHat, Fedora (type: character(1))
  - **release**: Distribution release e.g. for Debian: squeeze, wheezy, jessie (type: character(1))
  - **sysreq_type**: One of Windows, Pkg, RPM, DEB (type: character(1))
  - **build**: True if build environment is required (type: logical(1))
ci_adapter_create_base

Creates the base presentation for the CI adapter to use by concrete implementations.

Description

Creates the base presentation for the CI adapter to use by concrete implementations.

Usage

ci_adapter_create_base(name)

Arguments

name

name under which CI adapter will be registered in RSuite. It cannot contain whitespaces or comma. (type: character)

Value

object of type rsuite_ci_adapter

See Also

Other in extending RSuite with CI adapter: ci_adapter_get_version, ci_adapter_is_building

Examples

# create you own CI adapter
ci_adapter_create_own <- function() {
  result <- ci_adapter_create_base("Own")
  class(result) <- c("ci_adapter_own", class(result))
  return(result)
}

---

ci_adapter_get_version

Retrieves current CI build number.

Description

Retrieves current CI build number.

Usage

ci_adapter_get_version(ci_adapter)
ci_adapter_is_building

Arguments

  ci_adapter    ci adapter object

Value

  build number reported by CI. (type: character).

See Also

  Other in extending RSuite with CI adapter: **ci_adapter_create_base, ci_adapter_is_building**

Examples

```r
# create you own CI adapter
ci_adapter_create_own <- function() {
  result <- ci_adapter_create_base("Own")
  class(result) <- c("ci_adapter_own", class(result))
  return(result)
}

#' @export
.ci_adapter_get_version.ci_adapter_own <- function(ci_adapter) {
  # ... detect if build triggered by CI is currently running ...
  return("0.0")
}
```

---

ci_adapter_is_building

*Detects if build process triggered by CI is currently running.*

Description

Detects if build process triggered by CI is currently running.

Usage

```r
ci_adapter_is_building(ci_adapter)
```

Arguments

  ci_adapter    ci adapter object

Value

  TRUE if build triggered by CI is currently running.
See Also

Other in extending RSuite with CI adapter: `ci_adapter_create_base, ci_adapter_get_version`

Examples

```r
# create you own CI adapter
ci_adapter_create_own <- function() {
  result <- ci_adapter_create_base("Own")
  class(result) <- c("ci_adapter_own", class(result))
  return(result)
}

#' @export
ci_adapter_is_building.ci_adapter_own <- function(ci_adapter) {
  # ... check ...
  return(TRUE)
}
```

---

**get_version_numbers**

Retrieves version numbers from the input version string e.g. 1.2.0 returns c(1, 2, 0)

**Description**

Retrieves version numbers from the input version string e.g. 1.2.0 returns c(1, 2, 0)

**Usage**

```r
get_version_numbers(vers)
```

**Arguments**

- **vers**
  - list of versions which can contain blocks of digits separated with a dot or dash character (type: character).

**Value**

- list of versions digit vectors (type: list)
inst_wrap_zip  Wraps deployment zip into bash installer script.

Description
Wraps deployment zip into bash installer script.

Usage
inst_wrap_zip(zip_fpath)

Arguments
zip_fpath  path to zip package to wrap. It must exist. (type: character)

Details
Bash installer is just script containing also binary data at the end of file. It has also some logic to install the package into more intelligent way than just unzip.
Shell script is created at the same location as zip package passed and will have the same name with sh extension.

Value
path to created bash installer script (invisible)

Examples

inst_wrap_zip("myproj_0.1-1.zip")  # creates myproj_0.1-1.sh

perform  Performs(runs) all recipes from the sysreqs_recipe object.

Description
Performs(runs) all recipes from the sysreqs_recipe object.

Usage
perform(recipe)

Arguments
recipe  sysreqs_recipe object (type: sysreqs_recipe)
pkgzip_build_bioc_package

Builds PKGZIP out of a package on Bioconductor

Description

Loads package from the Bioconductor repository, packages it into package file and builds a PKGZIP out of it. It uses the project to detect repositories to look for dependencies and to detect rversion if required.

Usage

pkgzip_build_bioc_package(repo, ..., prj = NULL,
pkg_type = .Platform$pkgType, path = getwd(), with_deps = FALSE,
filter_repo = NULL, skip_build_steps = NULL, keep_sources = FALSE)

Arguments

repo    repository address in format [username:password@][release/][repo[#version]. See devtools::install_bioc for more information.
...    Bioconductor specific parameters passed to devtools::install_bioc.
prj    project object to use. If not passed will init project from working directory. (type: rsuite_project, default: NULL)
pkg_type    type of packages to build (type: character, default: platform default)
path    folder path to put output zip into. The folder must exist. (type: character, default: getwd())
with_deps    If TRUE will include dependencies pkg dependencies into final zip. (type: logical, default: FALSE)
filter_repo    repository address to not include dependencies available in. If NULL will not filter dependencies. Will be omitted if with_deps is FALSE. (type: character, default: NULL)
skip_build_steps    character vector with steps to skip while building project packages. Can contain following entries:
specs Process packages specifics
docs Try build documentation with roxygen
imps Perform imports validation
tests Run package tests
rppp_attribs Run rppAttribs on the package
vignettes Build package vignettes
(type: character(N), default: NULL).
keep_sources    if TRUE downloaded package sources will not be removed after building. (type: logical, default: FALSE)
pkgzip_build_ext_packages

Details
Logs all messages onto rsuite logger. Use logging::setLevel to control logs verbosity.

Value
created pkgzip file path (invisible).

See Also
Other in PKGZIP building: pkgzip_build_ext_packages, pkgzip_build_github_package, pkgzip_build_package_files, pkgzip_build_prj_packages

Examples

```r
# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# build PKGZIP with logging package from cran repository
pkgzip_fpath <- pkgzip_build_bioc_package("BiocGenerics", prj = prj, path = tempdir())

# list content of pkgzip created
unzip(pkgzip_fpath, list = TRUE)
```

pkgzip_build_ext_packages
Builds PKGZIP out of passed external packages.

Description
Builds PKGZIP out of passed external packages.

Usage

```
pkgzip_build_ext_packages(pkgs, prj = NULL,
    pkg_type = .Platform$pkgType, path = getwd(), with_deps = FALSE,
    filter_repo = NULL)
```
pkgzip_build_ext_packages

Arguments

pkgs vector of names of external packages which should be included in PKGZIP. (type: character)

prj project object to use. If not passed will init project from working directory. (type: rsuite_project, default: NULL)

pkg_type type of packages to build (type: character, default: platform default)

path folder path to put output zip into. The folder must exist. (type: character(1), default: getwd())

with_deps If TRUE will include dependencies pkgs dependencies into final zip. (type: logical, default: FALSE)

filter_repo repository address to not include dependencies available in. If NULL will not filter dependencies. Will be omitted if with_deps is FALSE. (type: character(1), default: NULL)

Details

It uses the project to detect repositories to look for packages in.
Logs all messages onto rsuite logger. Use logging::setLevel to control logs verbosity.

Value

created pkgzip file path (invisible).

See Also

Other in PKGZIP building: pkgzip_build_bioc_package, pkgzip_build_github_package, pkgzip_build_package_files, pkgzip_build_prj_packages

Examples

```r
# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# build PKGZIP with logging package
pkgzip_fpath <- pkgzip_build_ext_packages("logging", prj = prj, path = tempdir())

# list content of pkgzip created
unzip(pkgzip_fpath, list = TRUE)
```
pkgzip_build_github_package

Builds PKGZIP out of a package on GitHub.

Description

Loads package from the GitHub repository, packages it into package file and builds a PKGZIP out of it. It uses the project to detect repositories to look for dependencies and to detect rversion if required.

Usage

pkgzip_build_github_package(repo, ..., prj = NULL, 
 pkg_type = .Platform$pkgType, path = getwd(), with_deps = FALSE, 
 filter_repo = NULL, skip_build_steps = NULL, keep_sources = FALSE)

Arguments

repo repository address in format username/repo[/subdir][@ref|#pull]. See devtools::install_github for more information.

... GitHub specific parameters passed to devtools::install_github.

prj project object to use. If not passed will init project from working directory. 
(type: rsuite_project, default: NULL)

pkg_type type of packages to build (type: character, default: platform default)

path folder path to put output zip into. The folder must exist. (type: character: default: getwd())

with_deps If TRUE will include dependencies pkgs dependencies into final zip. (type: logical, default: FALSE)

filter_repo repository address to not include dependencies available in. If NULL will not filter dependencies. Will be omitted if with_deps is FALSE. (type: character(1), default: NULL)

skip_build_steps character vector with steps to skip while building project packages. Can contain following entries:

specs Process packages specifics

docs Try build documentation with roxygen

imps Perform imports validation

tests Run package tests

rcpp_attribs Run rppAttribs on the package

vignettes Build package vignettes

(type: character(N), default: NULL).

keep_sources if TRUE downloaded package sources will not be removed after building. (type: logical, default: FALSE)
pkgzip_build_package_files

Details
Logs all messages onto rsuite logger. Use logging::setLevel to control logs verbosity.

Value
created pkgzip file path (invisible).

See Also
Other in PKGZIP building: pkgzip_build_bioc_package, pkgzip_build_ext_packages, pkgzip_build_package_files, pkgzip_build_prj_packages

Examples

```r
# create exemplary project base folder
tp_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# build PKGZIP with logging package from cran repository
pkgzip_fpath <- pkgzip_build_github_package("cran/logging", prj = prj, path = tempdir())

# list content of pkgzip created
unzip(pkgzip_fpath, list = TRUE)
```

pkgzip_build_package_files

Builds PKGZIP out of passed package files.

Description
Builds PKGZIP out of passed package files.

Usage
`pkgzip_build_package_files(files, path = getwd())`

Arguments
- `files`: vector of files to upload. (type: character)
- `path`: folder path to put output zip into. The folder must exist. (type: character: default: getwd())
pkgzip_build_prj_packages

Details

Logs all messages onto rsuite logger. Use logging::setLevel to control logs verbosity.

Value

created pkgzip file path (invisible).

See Also

Other in PKGZIP building: pkgzip_build_bioc_package, pkgzip_build_ext_packages, pkgzip_build_github_package, pkgzip_build_prj_packages

Examples

# download logging package
pkg_fpath <- utils::download.packages("logging",
      repos = "https://cloud.r-project.org/",
      destdir = tempdir())[1,2]

# build PKGZIP
pkgzip_fpath <- pkgzip_build_package_files(files = pkg_fpath, path = tempdir())

# list content of pkgzip created
unzip(pkgzip_fpath, list = TRUE)

pkgzip_build_prj_packages

Builds PKGZIP out of project packages.

Description

Builds PKGZIP out of project packages.

Usage

pkgzip_build_prj_packages(pkgs = NULL, prj = NULL, zip_ver = NULL,
pkg_type = .Platform$pkgType, path = getwd(), with_deps = FALSE,
filter_repo = NULL, skip_build_steps = NULL)

Arguments

pkgs vector of project packages which should be included in PKGZIP or NULL to include all project packages (type: character, default: NULL)
prj project object to use. If not passed will init project from working directory. (type: rsuite_project, default: NULL)
zip_ver if passed enforce the version of PKGZIP package to the passed value. Expected form of version is DD.DD. (type: character, default: NULL)

pkg_type type of packages to build (type: character, default: platform default)

path folder path to put output zip into. The folder must exist. (type: character, default: getwd())

with_deps If TRUE will include dependencies pkgs dependencies into final zip. (type: logical, default: FALSE)

filter_repo repository address to not include dependencies available in. In a project, dependencies will never be filtered. If NULL will not filter dependencies. Will be omitted if with_deps is FALSE. (type: character(1), default: NULL)

skip_build_steps character vector with steps to skip while building project packages. Can contain following entries:

specs Process packages specifics
docs Try build documentation with roxygen
imps Perform imports validation
tests Run package tests	rcppAttribs Run rppAttribs on the package
vignettes Build package vignettes

(type: character(N), default: NULL).

Details
PKGZIP will be tagged with the same way as project zip.
Logs all messages onto rsuite logger. Use logging::setLevel to control logs verbosity.

Value
created pkgzip file path (invisible).

See Also
Other in PKGZIP building: pkgzip_build_bioc_package, pkgzip_build_ext_packages, pkgzip_build_github_package, pkgzip_build_package_files

Examples

```r
# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# start package in my_project
prj_start_package("mypackage", skip_rc = TRUE, prj = prj)
```
prj_build

Builds project internal packages and installs them.

Description

Builds project internal packages and installs them.

Usage

prj_build(prj = NULL, type = NULL, rebuild = FALSE, vignettes = TRUE, tag = FALSE)

Arguments

- **prj**: project to build if not passed will build project for working directory. (type: rsuite_project, default: NULL)
- **type**: type of packages to build. If NULL will build platform default. (type: character)
- **rebuild**: if TRUE will force rebuild all project packages event if no changes detected (type: logical)
- **vignettes**: if FALSE will not build vignettes which can highly decrease package building time (type: logical, default: TRUE)
- **tag**: if TRUE will tag packages with RC revision. Enforces rebuild. (type: logical; default: FALSE)

Details

Logs all messages from the building process onto the rsuite logger. Use logging::setLevel to control logs verbosity. DEBUG level turns on building and downloading messages.

See Also

Other in project management: prj_clean_deps, prj_init, prj_install_deps, prj_load, prj_lock_env, prj_pack, prj_start_package, prj_start, prj_unload, prj_zip
Examples

```r
# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# create package in the project
prj.start_package("mypackage", prj = prj, skip_rc = TRUE)

# build project local environment
prj.install_deps(prj = prj)

# build mypackage and install it into project environment
prj.build(prj = prj)
```

---

**prj_clean_deps**  
*Uninstalls unused packages from the local project environment.*

**Description**

Checks if all dependencies installed are required by project packages or master scripts and removes those which are not required any more.

**Usage**

`prj_clean_deps(prj = NULL)`

**Arguments**

`prj`  
project to clean dependencies of. If not passed will use the project base in the working directory. *(type: rsuite_project, default: NULL)*

**Details**

Logs all messages from the building process onto the rsuite logger. Use `logging::setLogLevel` to control logs verbosity. DEBUG level turns on building and downloading messages.

**See Also**

Other in project management: `prj_build, prj_init, prj_install_deps, prj_load, prj_lock_env, prj_pack, prj_start_package, prj_start, prj_unload, prj_zip`
Examples

# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# add colorspace to master script
master_script_fpath <- file.path(prj$path, "R", "master.R")
write("library(colorspace)", file = master_script_fpath, append = TRUE)

# install colorspace into project local environment
prj_install_deps(prj = prj)

# remove dependency to colorspace
writeLines(head(readLines(master_script_fpath), n = -1),
           con = master_script_fpath)

# uninstall colorspace from project local environment
prj_clean_deps(prj = prj)

---

prj_config_set_repo_adapters

*Updates project configuration to use only specified repository adapters.*

Description

Updates project configuration to use only specified repository adapters.

Usage

```
prj_config_set_repo_adapters(repos, prj = NULL)
```

Arguments

- **repos**
  - vector of repository adapters configuration to use by the project. Each should be in form `<repo_adapter_name>[<arg>]`. They should be all registered. (type: character)

- **prj**
  - project object to update configuration for. If not passed the loaded project will be used or the default whichever exists. Will init the default project from the working directory if no default project exists. (type: rsuite_project, default: NULL)
Details

Project configuration (together with repositories to be used) is stored in PARAMETERS file in the project folder.

After project configuration have been changed repository adapters are initialized on the project.

Repository adapters will be used for dependencies detection in the same order as passed in names.

See Also

Other in project configuration: prj_config_set_rversion

Examples

# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# present initial project configuration
cat(readLines(file.path(prj$path, "PARAMETERS")), sep = "\n")

# set repositories to use
prj_config_set_repo_adapters(c("CRAN", "MRAN[2018-01-01]")
, prj = prj)

# present final project configuration
cat(readLines(file.path(prj$path, "PARAMETERS")), sep = "\n")

prj_config_set_rversion

Updates project configuration to use specified R Version.

Description

Project configuration (together with R version to be used) is stored in PARAMETERS file in the project folder.

Usage

prj_config_set_rversion(rver, prj = NULL, validate = TRUE)

Arguments

rver
project object to update configuration for. If not passed the loaded project will used or the default whichever exists. Will init default project from the working directory if no default project exists. (type: rsuite_project, default: NULL)

prj
R version to be used by the project. (type: character)
prj_init

Loads project settings without loading them into the environment.

Description

Loads project settings without loading them into the environment.

Usage

prj_init(path = getwd())

Arguments

path  
path to start searching project base folder from. Search is performed upwards folder structure. Should be existing directory. (type: character, default: getwd())

Details

Project parameters are searched and loaded. If the project has been loaded previously from the path the same project instance will be used without reloading.

If the project is the first one loaded it will become the default project (used then NULL is passed as the project for project management functions).

validate If TRUE will check if R version is valid for the platform. (type: logical, default: TRUE)

See Also

Other in project configuration: prj_config_set_repo_adapters

Examples

# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# present initial project configuration
cat(readLines(file.path(prj$path, "PARAMETERS")), sep = "\n")

# set repositories to use
prj_config_set_rversion("3.2", prj = prj, validate = FALSE)

# present final project configuration
cat(readLines(file.path(prj$path, "PARAMETERS")), sep = "\n")
**prj_install_deps**

Value

object of type `rsuite_project`

See Also

Other in project management: `prj_build`, `prj_clean_deps`, `prj_install_deps`, `prj_load`, `prj_lock_env`, `prj_pack`, `prj_start_package`, `prj_start`, `prj_unload`, `prj_zip`

Examples

```r
# create exemplary project base folder
prj_base <- tempfile("example_.")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj_start("my_project", skip_rc = TRUE, path = prj_base)

# init project
prj <- prj_init(path = file.path(prj_base, "my_project"))
```

**Description**

Installs project dependencies and needed supportive packages.

**Usage**

```r
prj_install_deps(prj = NULL, clean = FALSE, vanilla_sups = FALSE, relock = FALSE)
```

**Arguments**

- **prj**: project to collect dependencies for if not passed will build project for working directory. (type: `rsuite_project`, default: `NULL`)
- **clean**: if TRUE clear environment before installing package dependencies. (type: logical, default: `FALSE`)
- **vanilla_sups**: if TRUE install only base supportive packages (like devtools & roxygen2). (type: logical, default: `FALSE`)
- **relock**: if TRUE allows updating the env.lock file (type: logical, default: `FALSE`)

**Details**

Logs all messages from the building process onto the rsuite logger. Use `logging::.setLevel` to control logs verbosity. DEBUG level turns on building and downloading messages.
prj_load

Value
TRUE if all build successfully.

See Also
Other in project management: prj_build, prj_clean_deps, prj_init, prj_load, prj_lock_env, prj_pack, prj_start_package, prj_start, prj_unload, prj_zip

Examples

# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# reinstall logging package into project environment
prj_install_deps(prj = prj, clean = TRUE)

desc

prj_load
 Loads project into the environment so all master scripts can run.

Description
It changes .libPaths() so project internal environment is visible for R. Use prj_unload to restore your environment.

Usage
prj_load(path, prj = NULL)

Arguments

path if prj is NULL, the path will be used to init new project to load. If passed must be existing folder path. (type: character)

prj project to load or NULL to use path for new project initialization. If not path passed project will be initialized from working folder. (type: rsuite_project, default: NULL)

Value
previously loaded project or NULL if no project has been loaded.
prj_lock_env

See Also

Other in project management: prj_build, prj_clean_deps, prj_init, prj_install_deps, prj_lock_env, prj_pack, prj_start_package, prj_start, prj_unload, prj_zip

Examples

# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

cat(.libPaths(), sep = "\n") # show initial contents of .libPaths()

prj_load(prj = prj) # load project

cat(.libPaths(), sep = "\n") # show contents of .libPaths()

prj_unload() # restore environment

cat(.libPaths(), sep = "\n") # show final contents of .libPaths()

prj_lock_env

Locks the project environment.

Description

It collects all dependencies’ versions and stores them in lock file to enforce exact dependency versions in the future.

Usage

prj_lock_env(prj = NULL)

Arguments

prj  
project object to be locked. If not passed the loaded project will be locked or the default whichever exists. Will init default project from the working directory if no default project exists. (type: rsuite_project, default: NULL)

Details

The lock file is saved in <my_project>/deployment/ under 'env.lock' name. It is in dcf format with information about packages installed in local project environment together with their versions. A sample record from the lock file:

Package: RSuite
Version: 0.26.235
When dependencies are being installed (using `prj_install_deps`) the 'env.lock' file will be used to detect whether any package will change versions. If that’s the case a warning message will be displayed like this:

```r
...:rsuite: The following packages will be updated from the last lock: colorspace
```

The feature allows preventing errors caused by newer versions of packages which might work differently than previous versions used in the project.

See Also

Other in project management: `prj_build`, `prj_clean_deps`, `prj_init`, `prj_install_deps`, `prj_load`, `prj_pack`, `prj_start_package`, `prj_start`, `prj_unload`, `prj_zip`

Examples

```r
# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# build project local environment
prj_install_deps(prj = prj)

# lock project environment
prj_lock_env(prj = prj)

# present contents of lock file created
cat(readlines(file.path(prj$path, "deployment", "env.lock")), sep = "\n")
```

---

**prj_pack**

Prepares project source pack tagged with version.

Description

It collects all sources and assemblies found in the project folder and packs them into a single zip file.

Usage

```
prj_pack(prj = NULL, path = getwd(), pkgs = NULL, inc_master = TRUE, pack_ver = NULL, rver = NULL)
```
prj_pack

Arguments

- **prj**: project object to pack. If not passed the loaded project will be packed or the default whichever exists. Will init default project from the working directory if no default project exists. (type: rsuite_project, default: NULL)
- **path**: folder path to put output pack into. The folder must exist. (type: character(1), default: getwd())
- **pkgs**: names of packages to include in the pack. If NULL will include all project packages (type: character, default: NULL)
- **inc_master**: if TRUE will include master scripts in the pack. (type: logical(1), default: TRUE)
- **pack_ver**: if passed enforce the version of the pack to the passed value. Expected form of version is DD.DD. (type: character(1), default: NULL)
- **rver**: if passed enforce destination R version of the pack. (type: character(1), default: NULL)

Details

The function is heavily used for building projects for alternative environments (like in docker).

Pack generated is stamped with version. It can be enforced with pack_ver parameter (zip will have suffix <pack_ver>x in the case). If the version is not enforced it is detected out of ZipVersion setting in project PARAMETERS file or from the maximal project packages version number. In that case, the revision number is appended to version: version number will be <ZipVersion>_<rc_ver>. Check for changes in project sources is performed for pack consistency. The resulted pack is marked with the version detected so while building zip after unpacking will have the same version as the original project.

Before building pack project packages will have version altered: revision will be added as the least number to package version.

Logs all messages onto rsuite logger. Use `logging::setLevel` to control logs verbosity.

Value

invisible file path to pack file created. The file name will be in form `prjpack_<ProjectName>_<version>.zip`

See Also

Other in project management: `prj_build, prj_clean_deps, prj_init, prj_install_deps, prj_load, prj_lock_env, prj_start_package, prj_start, prj_unload, prj_zip`

Examples

```r
# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)
```
prj_start

Creates project structure at the specified path.

Description

Creates project structure at the specified path.

Usage

prj_start(name = NULL, path = getwd(), skip_rc = FALSE, tmpl = "builtin")

Arguments

- **name**: name of the project to create. It must not contain special characters like /\"<> otherwise project folder could not be created. It can be NULL. If so project will be created at path directly with the name of the first folder. (type: character).
- **path**: path to the folder where project structure should be created.
- **skip_rc**: if TRUE skip adding project under revision control. (type: logical, default: FALSE)
- **tmpl**: name of the project template (or path to it) to use for project structure creation. (type: character).

Details

The project is not loaded, just created.

If name passed folder under such name will be created and project structure will be placed under it. If not passed folder under path will contain project structure and project name will be assumed to be basename of the path.

Logs all messages from the building process onto the rsuite logger. Use logging::setLevel to control logs verbosity. DEBUG level turns on building and downloading messages.

Project templates have to include a PARAMETERS file

Value

rsuite_project object for the project just created.
prj_start_package

See Also

Other in project management: prj_build, prj_clean_deps, prj_init, prj_install_deps, prj_load, prj_lock_env, prj_pack, prj_start_package, prj_unload, prj_zip

Examples

# create exemplary project base folder
prj_base <- tempfile("example")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

prj_start_package  Creates package structure inside the project.

Description

Creates package structure inside the project.

Usage

prj_start_package(name, prj = NULL, skip_rc = FALSE, tmpl = "builtin")

Arguments

name  name of the package to create. It must not contain special characters like \"\'<> otherwise package folder could not be created. It must not contain _ also as it is requirement enforced on R package names. The folder must not exist. (type: character).

prj  project object to create the package in. If not passed will init project from working directory. (type: rsuite_project, default: NULL)

skip_rc  if TRUE skip adding package under revision control. (type: logical, default: FALSE)

tmpl  name of the package template (or path to it) to use for package structure creation. (type: character).

Details

It fails if the package exists already in the project.

Logs all messages from the building process onto the rsuite logger. Use logging::setLevel to control logs verbosity. DEBUG level turns on building and downloading messages.

Package templates have to include the following files: DESCRIPTION, NAMESPACE, NEWS
prj_unload

See Also

Other in project management: prj_build, prj_clean_deps, prj_init, prj_install_deps, prj_load, prj_lock_env, prj_pack, prj_start_package, prj_start, prj_unload, prj_zip

Examples

# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# start package in it
prj_start_package("mypackage", prj = prj, skip_rc = TRUE)

---

prj_unload

Unloads last loaded project.

Description

It changes .libPaths() removing all references to currently loaded project internal environment.

Usage

prj_unload()

Value

Project unloaded or NULL if there was no project to unload.

See Also

Other in project management: prj_build, prj_clean_deps, prj_init, prj_install_deps, prj_load, prj_lock_env, prj_pack, prj_start_package, prj_start, prj_zip

Examples

# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

cat(.libPaths(), sep = "\n") # show initial contents of .libPaths()
prj_load(prj = prj) # load project
prj_unlock_env

Unlocks the project environment.

Description

It removes the lock file created with prj_lock_env. If the project environment is not locked (there is no lock file) the prj_unlock_env will fail.

Usage

prj_unlock_env(prj = NULL)

Arguments

prj project object to be unlocked. if not passed the loaded project will be locked or the default whichever exists. Will init default project from the working directory if no default project exists. (type: rsuite_project, default: NULL)

Examples

# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# build project local environment
prj_install_deps(prj = prj)

# lock project environment
prj_lock_env(prj = prj)

# unlock project environment
prj_unlock_env(prj = prj)

---

cat(.libPaths(), sep = "\n") # show contents of .libPaths()

prj_unload() # restore environment
cat(.libPaths(), sep = "\n") # show final contents of .libPaths()
**prj_zip**  
*Prepares deployment zip tagged with version.*

**Description**

It collects all dependencies and project packages installed in local project environment together with master scripts and artifacts and zips them into a single zip file.

**Usage**

`prj_zip(prj = NULL, path = getwd(), zip_ver = NULL)`

**Arguments**

- **prj**  
  project object to zip. if not passed will zip the loaded project or the default whichever exists. Will init default project from the working directory if no default project exists. (type: rsuite_project, default: NULL)

- **path**  
  folder path to put output zip into. If the folder does not exist, will create it. (type: character, default: getwd())

- **zip_ver**  
  if passed enforce the version of the zip package to the passed value. Expected form of version is DD.DD. (type: character, default: NULL)

**Details**

Zip package generated is stamped with version. It can be enforced with zip_ver parameter (zip will have suffix `<zip_ver>x` in the case). If the version is not enforced it is detected out of ZipVersion setting in project PARAMETERS file or from the maximal project packages version number. In that case, revision number is appended to version: version number will be `<zip_ver>_<rc_ver>`. Check for changes in project sources is performed for zip package consistency.

Before building zip package project is built. If revision number detected project packages will have version altered: revision will be added as least number to package version.

Logs all messages from the building process onto rsuite logger. Use `logging::setLevel` to control logs verbosity. DEBUG level turns on building and downloading messages.

**Value**

invisible file path to pack file created. The file name will be in form `<ProjectName>_<version>.zip`

**See Also**

Other in project management: `prj_build, prj_clean_deps, prj_init, prj_install_deps, prj_load, prj_lock_env, prj_pack, prj_start_package, prj_start, prj_unload`
Examples

# create exemplary project base folder
prj_base <- tempfile("example")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# build deployment zip
zip_fpath <- prj_zip(prj = prj, path = tempdir(), zip_ver = "1.0")

---

rc_adapter_create_base

*Creates the base presentation for the RC adapter to use by concrete implementations.*

Description

Creates the base presentation for the RC adapter to use by concrete implementations.

Usage

`rc_adapter_create_base(name)`

Arguments

- **name**: name under which RC adapter will be registered in RSuite. It cannot contain whitespaces or comma. (type: character)

Value

object of type `rsuite_rc_adapter`

See Also

Other in extending RSuite with RC adapter: `rc_adapter_get_version`, `rc_adapter_is_under_control`, `rc_adapter_pkg_struct_add`, `rc_adapter_prj_struct_add`, `rc_adapter_remove_admins`

Examples

# create you own RC adapter
rc_adapter_create_own <- function() {
  result <- rc_adapter_create_base("Own")
  class(result) <- c("rc_adapter_own", class(result))
  return(result)
}
rc_adapter_get_version

Retrieves current RC version number for working copy at directory passed.

Description
Retrieves current RC version number for working copy at directory passed.

Usage
rc_adapter_get_version(rc_adapter, dir)

Arguments
rc_adapter rc adapter object
dir path to the directory to get the version for. The folder must exist (type: character)

Value
named list with following entries:

- **has_changes** TRUE if changes detected by RC in the directory. (type: logical)
- **revision** revision reported by RC. (type: character)
- **latest** the latest revision reported by RC at the repository. (type: character)

See Also
Other in extending RSuite with RC adapter: rc_adapter_create_base, rc_adapter_is_under_control, rc_adapter_pkg_struct_add, rc_adapter_prj_struct_add, rc_adapter_remove_admins

Examples

```r
# create you own RC adapter
rc_adapter_create_own <- function() {
  result <- rc_adapter_create_base("Own")
  class(result) <- c("rc_adapter_own", class(result))
  return(result)
}

#' @export
rc_adapter_get_version.rc_adapter_own <- function(rc_adapter, dir) {
  # ... detect if working copy is consistent with repository state ...
  return(list(has_changes = TRUE,
               revision = "0.0",
               latest = FALSE))
}
```
rc_adapter_is_under_control

Detects if dir is under adapter’s managed version control.

Description

Detects if dir is under adapter’s managed version control.

Usage

rc_adapter_is_under_control(rc_adapter, dir)

Arguments

- rc_adapter: rc adapter object
- dir: path to the directory to check. The folder must exist (type: character)

Value

TRUE if dir is under version control.

See Also

Other in extending RSuite with RC adapter: rc_adapter_create_base, rc_adapter_get_version, rc_adapter_pkg_struct_add, rc_adapter_prj_struct_add, rc_adapter_remove_admins

Examples

```r
# create you own RC adapter
rc_adapter_create_own <- function() {
  result <- rc_adapter_create_base("Own")
  class(result) <- c("rc_adapter_own", class(result))
  return(result)
}

# @export
rc_adapter_is_under_control.rc_adapter_own <- function(rc_adapter, dir) {
  # ... check ...
  return(TRUE)
}
```
rc_adapter_pkg_struct_add

*Puts the package structure under RC adapter’s managed version control.*

---

Description

Puts the package structure under RC adapter’s managed version control.

Usage

```
rc_adapter_pkg_struct_add(rc_adapter, params, name)
```

Arguments

- `rc_adapter`: rc adapter object
- `params`: rsuite_project_params object of the project.
- `name`: name of the package to put under RC adapter’s managed version control. Appropriate sub-folder must exist in project packages folder. (type: character)

See Also

Other in extending RSuite with RC adapter: `rc_adapter_create_base`, `rc_adapter_get_version`, `rc_adapter_is_under_control`, `rc_adapter_prj_struct_add`, `rc_adapter_remove_admins`

Examples

```
# create you own RC adapter
rc_adapter_create_own <- function() {
  result <- rc_adapter_create_base("Own")
  class(result) <- c("rc_adapter_own", class(result))
  return(result)
}

#' @export
rc_adapter_pkg_struct_add.rc_adapter_own <- function(rc_adapter, params, name) {
  # ... add package specified by name under RC in project specified by params ...
}
```
**Description**

Puts project structure under RC adapter’s managed version control.

**Usage**

```plaintext
rc_adapter_prj_struct_add(rc_adapter, params)
```

**Arguments**

- `rc_adapter`: rc adapter object
- `params`: rsuite_project_params object of the project.

**See Also**

Other in extending RSuite with RC adapter: `rc_adapter_create_base`, `rc_adapter_get_version`, `rc_adapter_is_under_control`, `rc_adapter_pkg_struct_add`, `rc_adapter_remove_admins`

**Examples**

```plaintext
# create you own RC adapter
rc_adapter_create_own <- function() {
  result <- rc_adapter_create_base("Own")
  class(result) <- c("rc_adapter_own", class(result))
  return(result)
}

# @export
rc_adapter_prj_struct_add.rc_adapter_own <- function(rc_adapter, params) {
  # ... add project specified by params under RC ...
}
```

---

**rc_adapter_remove_admins**

*Remove all RC related administrative entries from folder tree at dir.*

**Description**

This is required for cleaning temporary folder during collecting entries to put into project zip package.
repo_adapter_create_base

Creates base presentation for repo adapter to use by concrete implementations.

Description

Creates base presentation for repo adapter to use by concrete implementations.

Usage

repo_adapter_create_base(name)

Arguments

name name under which repository adapter will be registered in RSuite. It cannot contain whitespaces or comma. (type: character)
repo_adapter_create_manager

Value

object of type rsuite_repo_adapter

See Also

Other in extending RSuite with Repo adapter: repo_adapter_create_manager, repo_adapter_get_info, repo_adapter_get_path, repo_manager_destroy, repo_manager_get_info, repo_manager_init, repo_manager_remove, repo_manager_upload

Examples

# create you own Repo adapter
repo_adapter_create_own <- function() {
  result <- repo_adapter_create_base("Own")
  class(result) <- c("repo_adapter_own", class(result))
  return(result)
}

repo_adapter_create_manager

Creates repo manager to manage its repository.

Description

For repositories which need some kind of connection to manage it initializes appropriate resources.

Usage

repo_adapter_create_manager(repo_adapter, ...)

Arguments

repo_adapter repo adapter on which manager is base. (type: rsuite_repo_adapter)
...
manager specific parameters.

Details

Raises an error if fails to create the manager.

Value

object of type rsuite_repo_adapter

See Also

Other in extending RSuite with Repo adapter: repo_adapter_create_base, repo_adapter_get_info, repo_adapter_get_path, repo_manager_destroy, repo_manager_get_info, repo_manager_init, repo_manager_remove, repo_manager_upload
Examples

```r
# create you own Repo adapter
repo_adapter_create_own <- function() {
  result <- repo_adapter_create_base("Own")
  class(result) <- c("repo_adapter_own", class(result))
  return(result)
}

#' @export
repo_adapter_create_manager.repo_adapter_own <- function(repo_adapter, ...) {
  repo_manager <- list() # create you own repo manager
  class(repo_manager) <- c("repo_manager_own", "rsuite_repo_manager")
  return(repo_manager)
}
```

---

repo_adapter_get_info  Returns information about repository the adapter is working on.

Description

Returns information about repository the adapter is working on.

Usage

`repo_adapter_get_info(repo_adapter, params)`

Arguments

- `repo_adapter`  repo adapter object
- `params`  rsuite_project_params object

Value

named list with following entries:

- **readonly**  TRUE if the repository is for reading only (type:logical)
- **reliable**  TRUE if the content of the repository does not change over time unless repository changes enforce changes of the project itself (like project local repository) (type: logical).

See Also

Other in extending RSuite with Repo adapter: `repo_adapter_create_base, repo_adapter_create_manager, repo_adapter_get_path, repo_manager_destroy, repo_manager_get_info, repo_manager_init, repo_manager_remove, repo_manager_upload`
repo_adapter_get_path

Examples

```r
# create you own Repo adapter
repo_adapter_create_own <- function() {
  result <- repo_adapter_create_base("Own")
  class(result) <- c("repo_adapter_own", class(result))
  return(result)
}

#' @export
repo_adapter_get_info.repo_adapter_own <- function(repo_adapter, params) {
  return(list(
    readonly = TRUE, # cannot be managed
    reliable = FALSE # package versions can change in time
  ))
}
```

---

repo_adapter_get_path  Returns the adapter path related to the project to use for dependencies resolution.

Description

Returns the adapter path related to the project to use for dependencies resolution.

Usage

```r
repo_adapter_get_path(repo_adapter, params, ix = NA)
```

Arguments

- `repo_adapter`: repo adapter object
- `params`: rsuite_project_params object
- `ix`: repo adapter index in project repositories or NA to retrieve all paths for the adapter. (type: integer, default: NA)

Value

- path to the repository for the project.

See Also

Other in extending RSuite with Repo adapter: `repo_adapter_create_base`, `repo_adapter_create_manager`, `repo_adapter_get_info`, `repo_manager_destroy`, `repo_manager_get_info`, `repo_manager_init`, `repo_manager_remove`, `repo_manager_upload`
repo_manager_destroy

Releases resources allocated to manage the repository.

Description

For repositories which need some kind of connection to manage it cleans up previously initialized connection and releases all appropriate resources.

Usage

repo_manager_destroy(repo_manager)

Arguments

repo_manager repo adapter object.

See Also

Other in extending RSuite with Repo adapter: repo_adapter_create_base, repo_adapter_create_manager, repo_adapter_get_info, repo_adapter_get_path, repo_manager_get_info, repo_manager_init, repo_manager_remove, repo_manager_upload

Examples

# create you own repo adapter
repo_adapter_create_own <- function() {
  result <- repo_adapter_create_base("Own")
  class(result) <- c("repo_adapter_own", class(result))
  return(result)
}

# create own repo manager

repo_manager_get_info

# @export
repo_adapter_create_manager.repo_adapter_own <- function(repo_adapter, ...) {
  repo_manager <- list() # create you own repo manager
  class(repo_manager) <- c("repo_manager_own", "rsuite_repo_manager")
  return(repo_manager)
}

# @export
repo_manager_destroy.repo_manager_own <- function(repo_manager) {
  # ... release resources ...
}

repo_manager_get_info  Returns information on repo manager.

Description

Returns information on repo manager.

Usage

repo_manager_get_info(repo_manager)

Arguments

repo_manager  repo manager object

Value

named list with following entries:

  types  Types of packages manager can manage. (type: character)
  server R version repo manager is managing. NA if repo manager is managing source packages. (type: character)
  url   Url to the repository. (type: character)

See Also

Other in extending RSuite with Repo adapter: repo_adapter_create_base, repo_adapter_create_manager, repo_adapter_get_info, repo_adapter_get_path, repo_manager_destroy, repo_manager_init, repo_manager_remove, repo_manager_upload
Examples

```r
# create you own Repo adapter
repo_adapter_create_own <- function() {
  result <- repo_adapter_create_base("Own")
  class(result) <- c("repo_adapter_own", class(result))
  return(result)
}

#' create own repo manager
#' @export
repo_adapter_create_manager.repo_adapter_own <- function(repo_adapter, ...) {
  repo_manager <- list() # create you own repo manager
  class(repo_manager) <- c("repo_manager_own", "rsuite_repo_manager")
  return(repo_manager)
}

#' @export
repo_manager_get_info.repo_manager_own <- function(repo_manager) {
  return(list(
    types = c("source", "win-binary"), # package types supported by the manager
    rver = "3.5", # R version supported by the manager
    url = "file:///..." # base URL of repository
  ))
}
```

---

**repo_manager_init**  
Initializes managed repository structure.

**Description**

Initializes managed repository structure.

**Usage**

`repo_manager_init(repo_manager, types)`

**Arguments**

- `repo_manager` repo manager object
- `types` package types for which repository should be initialized. If missing all project supported package types will be initialized (type: character)

**Value**

TRUE if initialized repository for at least one type, FALSE if the structure was fully initialized already. (type: logical, invisible)
repo_manager_remove

Removes specified packages from the repository.

Description

Removes specified packages from the repository.

Usage

repo_manager_remove(repo_manager, toremove, type)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>repo_manager</td>
<td>repo manager object.</td>
</tr>
<tr>
<td>toremove</td>
<td>data.frame with at least Package(type: character) and Version(type: character) columns. (type: data.frame)</td>
</tr>
<tr>
<td>type</td>
<td>package type to remove</td>
</tr>
</tbody>
</table>

Examples

# create you own Repo adapter
repo_adapter_create_own <- function() {
  result <- repo_adapter_create_base("Own")
  class(result) <- c("repo_adapter_own", class(result))
  return(result)
}

# create own repo manager
reposuite_adapter_create_manager.reposuite_adapter_own <- function(repo_adapter, ...) {
  repo_manager <- list() # create you own repo manager
  class(repo_manager) <- c("repo_manager_own", "rsuite_reposuite_manager")
  return(repo_manager)
}

reposuite_manager_init.reposuite_manager_own <- function(repo_manager, types) {
  was_inited_already <- TRUE
  # ... if repository structure was not initialized initialize it ...
  return(invisible(was_inited_already))
}

See Also

Other in extending RSuite with Repo adapter: repo_adapter_create_base, repo_adapter_create_manager, repo_adapter_get_info, repo_adapter_get_path, repo_manager_destroy, repo_manager_get_info, repo_manager_remove, repo_manager_upload
repo_manager_upload

Value

data.frame containing packages removed with Package and Version columns.

See Also

Other in extending RSuite with Repo adapter: repo_adapter_create_base, repo_adapter_create_manager, repo_adapter_get_info, repo_adapter_get_path, repo_manager_destroy, repo_manager_get_info, repo_manager_init, repo_manager_upload

Examples

# create your own repo adapter
repo_adapter_create_own <- function() {
  result <- repo_adapter_create_base("Own")
  class(result) <- c("repo_adapter_own", class(result))
  return(result)
}

#' create own repo manager
#' @export
repo_adapter_create_manager.repo_adapter_own <- function(repo_adapter, ...) {
  repo_manager <- list() # create you own repo manager
  class(repo_manager) <- c("repo_manager_own", "rsuite_repo_manager")
  return(repo_manager)
}

#' @export
repo_manager_remove.repo_manager_remove <- function(repo_manager, toremove, type) {
  # ... remove packages from the repository ...
  # ... update PACKAGES ... 
  return(data.frame(Package = c(), # return packages removed
                   Version = c(),
                   stringsAsFactors = FALSE))
}

---

repo_manager_upload  Adds packages to the managed repository.

Description

Adds packages to the managed repository.

Usage

repo_manager_upload(repo_manager, src_dir, types)
repo_mng_init

Initializes repository (creates its structure).

Description

Initializes repository (creates its structure).

Usage

repo_mng_init(repo_manager)

Arguments

repo_manager  repo manager object.
src_dir  local directory repository path. The directory must exist. (type: character)
types  type of packages to sync. If missing all project supported package types will be synced. (type: character(N))

See Also

Other in extending RSuite with Repo adapter: repo_adapter_create_base, repo_adapter_create_manager, repo_adapter_get_info, repo_adapter_get_path, repo_manager_destroy, repo_manager_get_info, repo_manager_init, repo_manager_remove

Examples

# create you own repo adapter
repo_adapter_create_own <- function() {
  result <- repo_adapter_create_base("Own")
  class(result) <- c("repo_adapter_own", class(result))
  return(result)
}

#' create own repo manager
#' @export
repo_adapter_create_manager.repo_adapter_own <- function(repo_adapter, ...) {
  repo_manager <- list() # create you own repo manager
  class(repo_manager) <- c("repo_manager_own", "rsuite_repo_manager")
  return(repo_manager)
}

#' @export
repo_manager_upload.repo_manager_own <- function(repo_manager, src_dir, types) {
  # ... upload packages in src_dir into the repository ...
  # ... update PACKAGES ...
}
repo_mng_list

Arguments

repo_manager repo manager object retrieved with repo_mgr_start. (type: rsuite_repo_manager)

See Also

Other in repository management: repo_mng_list, repo_mng_remove, repo_mng_start, repo_mng_stop,
repo_upload_bioc_package, repo_upload_ext_packages, repo_upload_github_package, repo_upload_package_files,
repo_upload_pkgzip, repo_upload_prj_packages

Examples

# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# set it to use in project repository and CRAN
prj_config_set_repo_adapters(c("Dir", "CRAN"), prj = prj)

# start managing in project repository
rmgr <- repo_mng_start("Dir", prj = prj, ix = 1)

# initialize its structure
repo_mng_init(rmgr)

# stop repository management
repo_mng_stop(rmgr)

---

repo_mng_list Retrieve the list of available packages in the repository.

Description

Retrieve the list of available packages in the repository.

Usage

repo_mng_list(repo_manager, pkg_type = .Platform$pkgType,
              no.cache = FALSE)

Arguments

repo_manager repo manager to retrieve package list from. (type: rsuite_repo_manager)

pkg_type type of packages to retrieve list of. (type: character, default to platform default package type)

no.cache if TRUE will delete cached list before retrieving. (type: logical(1), default: FALSE)
repo_mng_remove

Value

data.frame of the same structure as available.packages returns.

See Also

Other in repository management: repo_mng_init, repo_mng_remove, repo_mng_start, repo_mng_stop, repo_upload_bioc_package, repo_upload_ext_packages, repo_upload_github_package, repo_upload_package_files, repo_upload_pkgzip, repo_upload_prj_packages

Examples

# create exemplary project base folder
prj_base <- tempfile("example_"
)
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base
)

# set it to use in project repository and CRAN
prj_config_set_repo_adapters(c("Dir", "CRAN"), prj = prj
)

# start managing in project repository
rmgr <- repo_mng_start("Dir", prj = prj, ix = 1
)

# upload logging package from CRAN into the repository
repo_upload_ext_packages(rmgr, pkgs = "logging", prj = prj
)

# list available packages
repo_mng_list(rmgr)

# stop repository management
repo_mng_stop(rmgr)

---

repo_mng_remove Removes packages from the repository.

Description

Removes packages from the repository.

Usage

repo_mng_remove(repo_manager, toremove, pkg_type = .Platform$pkgType)
repo_mng_remove

Arguments

repo_manager repo manager to remove packages from. (type: rsuite_repo_manager)
toremove data.frame with same structure as available.packages returns. At least Package and Version columns must be present. (type: data.frame)
pkg_type type of packages to remove. (type: character, default: .Platform$pkgType)

See Also

Other in repository management: repo_mng_init, repo_mng_list, repo_mng_start, repo_mng_stop, repo_upload_bioc_package, repo_upload_ext_packages, repo_upload_github_package, repo_upload_package_files, repo_upload_pkgzip, repo_upload_prj_packages

Examples

# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# set it to use in project repository and CRAN
prj_config_set_repo_adapters(c("Dir", "CRAN"), prj = prj)

# start managing in project repository
rmgr <- repo_mng_start("Dir", prj = prj, ix = 1)

# upload logging package from CRAN into the repository
repo_upload_ext_packages(rmgr, pkgs = "logging", prj = prj)

# list available packages before removal
avail_pkgs <- repo_mng_list(rmgr)
avail_pkgs

# remove logging from the repository
repo_mng_remove(rmgr, avail_pkgs[avail_pkgs$Package == "logging", ])

# list available packages after removal
repo_mng_list(rmgr)

# stop repository management
repo_mng_stop(rmgr)
repo_mng_start

repo_mng_start  Starts management over the repository.

Description

Creates object to manage the repository.

Usage

repo_mng_start(ra_name, ...)

Arguments

ra_name  name of the repository to whose adapter will be re-initialized. (type: character)

...  repository specific parameters. See repo_adapter_create_manager for the concrete implementation of repo adapter for more details.

Value

repo manager object.

See Also

Other in repository management: repo_mng_init, repo_mng_list, repo_mng_remove, repo_mng_stop, repo_upload_bioc_package, repo_upload_ext_packages, repo_uploadgithub_package, repo_upload_package_files, repo_upload_pkgzip, repo_upload_prj_packages

Examples

# create exemplary project base folder
prj_base <- tempfile("example")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# set it to use in project repository and CRAN
prj_config_set_repo_adapters(c("Dir", "CRAN"), prj = prj)

# start managing in project repository
rmgr <- repo_mng_start("Dir", prj = prj, ix = 1)

# stop repository management
repo_mng_stop(rmgr)
repo_mng_stop

Description

Stops management over the repository.

Usage

repo_mng_stop(repo_manager)

Arguments

repo_manager  repo manager object retrieved with repo_mgr_start. (type: rsuite_repo_manager)

See Also

Other in repository management: repo_mng_init, repo_mng_list, repo_mng_remove, repo_mng_start, repo_upload_bioc_package, repo_upload_ext_packages, repo_upload_github_package, repo_upload_package_files, repo_upload_pkgzip, repo_upload_prj_packages

Examples

# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# set it to use in project repository and CRAN
prj_config_set_repo_adapters(c("Dir", "CRAN"), prj = prj)

# start managing in project repository
rmgr <- repo_mng_start("Dir", prj = prj, ix = 1)

# stop repository management
repo_mng_stop(rmgr)
repo_upload_bioc_package

Loads package from the Bioconductor repository.

Description

It will download Bioconductor repository, build package into package file and will upload it into the repository. It will search dependencies in provided project’s repositories.

Usage

repo_upload_bioc_package(repo_manager, repo, ..., prj = NULL,
 pkg_type = .Platform$pkgType, with_deps = FALSE,
 skip_build_steps = NULL, keep_sources = FALSE)

Arguments

repo_manager repo manager to use for uploading. (type: rsuite_repo_manager)
repo repository address in format [username:password@[release/][repo[#revision]].
See devtools::install_bioc for more information.
... Bioconductor specific parameters passed to devtools::install_bioc.
prj project object to use. If not passed will init project from working directory.
(type: rsuite_project, default: NULL)
pkg_type type of packages to upload (type: character, default: platform default)
with_deps If TRUE will include pkgs dependencies while uploading into the repository.
Packages in repository satisfying pkgs requirements will not be included. (type:
logical, default: FALSE)
skip_build_steps character vector with steps to skip while building project packages. Can contain
following entries:
specs Process packages specifics
docs Try build documentation with roxygen
imps Perform imports validation
tests Run package tests
rcpp_attribs Run rppAttribs on the package
vignettes Build package vignettes
(type: character(N), default: NULL).
keep_sources if TRUE downloaded package sources will not be removed after building. (type:
logical, default: FALSE)

Details

Logs all messages onto rsuite logger. Use logging::setLevel to control logs verbosity.
See Also

Other in repository management: `repo_mng_init`, `repo_mng_list`, `repo_mng_remove`, `repo_mng_start`, `repo_mng_stop`, `repo_upload_ext_packages`, `repo_upload_github_package`, `repo_upload_package_files`, `repo_upload_pkgzip`, `repo_upload_prj_packages`

Examples

```
# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# set it to use in project repository and CRAN
prj_config_set_repo_adapters(c("Dir", "CRAN"), prj = prj)

# start managing in project repository
rmgr <- repo_mng_start("Dir", prj = prj, ix = 1)

# upload logging package from cran repository
repo_upload_bioc_package(rmgr, repo = "BiocGenerics",
                           prj = prj, pkg_type = "source")

# list available packages
repo_mng_list(rmgr, pkg_type = "source")

# stop repository management
repo_mng_stop(rmgr)
```

---

**repo_upload_ext_packages**

*Uploads external packages into the managed repository.*

**Description**

It uses the project to detect repositories to look for external packages in.

**Usage**

```r
repo_upload_ext_packages(repo_manager, pkgs, prj = NULL,
                         pkg_type = .Platform$pkgType, with_deps = FALSE)
```
repo_upload_ext_packages

Arguments

- **repo_manager**: repo manager to use for uploading. (type: rsuite_repo_manager)
- **pkgs**: vector of names of external packages which should be included in PKGZIP. (type: character)
- **prj**: project object to use. If not passed will init project from working directory. (type: rsuite_project, default: NULL)
- **pkg_type**: type of packages to upload (type: character, default: platform default)
- **with_deps**: If TRUE will include pkgs dependencies while uploading into the repository. Packages in repository satisfying pkgs requirements will not be included. (type: logical, default: FALSE)

Details

Logs all messages onto rsuite logger. Use `logging::setLevel` to control logs verbosity.

See Also

Other in repository management: `repo_mng_init`, `repo_mng_list`, `repo_mng_remove`, `repo_mng_start`, `repo_mng_stop`, `repo_upload_bioc_package`, `repo_upload_github_package`, `repo_upload_package_files`, `repo_upload_pkgzip`, `repo_upload_prj_packages`

Examples

```r
# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# set it to use in project repository and CRAN
prj_config_set_repo_adapters(c("Dir", "CRAN"), prj = prj)

# start managing in project repository
rmgr <- repo_mng_start("Dir", prj = prj, ix = 1)

# upload logging package from CRAN into the repository
repo_upload_ext_packages(rmgr, "logging", prj = prj, pkg_type = "source")

# list available packages
repo_mng_list(rmgr, pkg_type = "source")

# stop repository management
repo_mng_stop(rmgr)
```
repo_upload_github_package

Loads package from the GitHub repository.

Description

It will download GitHub repository, build package into package file and will upload it into the repository. It will search dependencies in provided project’s repositories.

Usage

repo_upload_github_package(repo_managerL repoL ...., prj = NULL,
   pkg_type = .Platform$pkgType, with_deps = FALSE,
   skip_build_steps = NULL, keep_sources = FALSE)

Arguments

repo_manager repo manager to use for uploading. (type: rsuite_repo_manager)
repo repository address in format username/repo[/subdir][@ref|#pull]. See devtools::install_github for more information.
... GitHub specific parameters passed to devtools::install_github.
prj project object to use. If not passed will init project from working directory. (type: rsuite_project, default: NULL)
pkg_type type of packages to upload (type: character, default: platform default)
with_deps If TRUE will include pkgs dependencies while uploading into the repository. Packages in repository satisfying pkgs requirements will not be included. (type: logical, default: FALSE)
skip_build_steps character vector with steps to skip while building project packages. Can contain following entries:
specs Process packages specifics
docs Try build documentation with roxygen
imps Perform imports validation
tests Run package tests
rcpp_atts Run rcppAtttibs on the package
vignettes Build package vignettes
   (type: character(N), default: NULL).
keep_sources if TRUE downloaded package sources will not be removed after building. (type: logical, default: FALSE)

Details

Logs all messages onto rsuite logger. Use logging::setLevel to control logs verbosity.
repo_upload_package_files

See Also
Other in repository management: `repo_mng_init`, `repo_mng_list`, `repo_mng_remove`, `repo_mng_start`, `repo_mng_stop`, `repo_upload_bioc_package`, `repo_upload_ext_packages`, `repo_upload_package_files`, `repo_upload_pkgzip`, `repo_upload_prj_packages`

Examples

```r
# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# set it to use in project repository and CRAN
prj_config_set_repo_adapters(c("Dir", "CRAN"), prj = prj)

# start managing in project repository
rmgr <- repo_mng_start("Dir", prj = prj, ix = 1)

# upload logging package from cran repository
repo_upload_github_package(rmgr, repo = "cran/logging",
                           prj = prj, pkg_type = "source")

# list available packages
repo_mng_list(rmgr, pkg_type = "source")

# stop repository management
repo_mng_stop(rmgr)
```

---

repo_upload_package_files

*Uploads package file(s) into the managed repository.*

Description

Uploads package file(s) into the managed repository.

Usage

```r
repo_upload_package_files(repo_manager, files)
```

Arguments

- `repo_manager`  
  repo manager to use for uploading. (type: rsuite_repo_manager)
- `files`  
  vector of files to upload. (type: character)
repo_upload_pkgzip

Details

Logs all messages onto the rsuite logger. Use logging::setLevel to control logs verbosity.

See Also

Other in repository management: repo_mng_init, repo_mng_list, repo_mng_remove, repo_mng_start, repo_mng_stop, repo_upload_bioc_package, repo_upload_ext_packages, repo_upload_github_package, repo_upload_pkgzip, repo_upload_prj_packages

Examples

```r
# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# set it to use in project repository and CRAN
prj_config_set_repo_adapters(c("Dir", "CRAN"), prj = prj)

# start managing in project repository
rmgr <- repo_mng_start("Dir", prj = prj, ix = 1)

# download logging package
pkg_fpath <- utils::download.packages("logging",
    repos = "https://cloud.r-project.org/",
    destdir = tempdir(),
    type = "source")[1,2]

# upload downloaded package into the repository
repo_upload_package_files(rmgr, files = pkg_fpath)

# list available packages
repo_mng_list(rmgr, pkg_type = "source")

# stop repository management
repo_mng_stop(rmgr)
```

Description

Uploads PKGZIPI into the managed repository.
repo_upload_pkgzip

Usage

repo_upload_pkgzip(repo_manager, pkgzip)

Arguments

repo_manager  repo manager to use for uploading. (type: rsuite_repo_manager)
pkgzip        PKGZIP path to upload. It must exist. (type: character(1))

Details

Logs all messages onto the rsuite logger. Use logging::setLevel to control logs verbosity.

See Also

Other in repository management: repo_mng_init, repo_mng_list, repo_mng_remove, repo_mng_start, repo_mng_stop, repo_upload_bioc_package, repo_upload_ext_packages, repo_upload_github_package, repo_upload_package_files, repo_upload_prj_packages

Examples

```r
# create exemplary project base folder
prj_base <- tempfile("example_B")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# set it to use in project repository and CRAN
prj_config_set_repo_adapters(c("Dir", "CRAN"), prj = prj)

# start managing in project repository
rmgr <- repo_mng_start("Dir", prj = prj, ix = 1)

# create PKGZIP containing logging package
pkgzip_fpath <- pkgzip_build_ext_packages("logging", prj = prj, pkg_type = "source",
    path = tempdir())

# upload PKGZIP into the repository
repo_upload_pkgzip(rmgr, pkgzip_fpath)

# list available packages
repo_mng_list(rmgr, pkg_type = "source")

# stop repository management
repo_mng_stop(rmgr)
```
repo_upload_prj_packages

Builds and uploads project package(s) into the repository.

Description

Builds and uploads project package(s) into the repository.

Usage

repo_upload_prj_packages(repo_manager, pkgs = NULL, prj = NULL,
skip_rc = FALSE, pkg_type = .Platform$pkgType, with_deps = FALSE,
skip_build_steps = NULL)

Arguments

repo_manager  repo manager to use for uploading. (type: rsuite_repo_manager)
pkgs  vector of project packages which should be uploaded into the repository or NULL to upload all project packages (type: character, default: NULL)
prj  project object to use. If not passed will init project from working directory. (type: rsuite_project, default: NULL)
skip_rc  if TRUE skip detection of package revision and package tagging. (type: logical, default: FALSE)
pkg_type  type of packages to upload (type: character, default: platform default)
with_deps  If TRUE will include pkgs dependencies while uploading into the repository. Packages in repository satisfying pkgs requirements will not be included. (type: logical, default: FALSE)
skip_build_steps  character vector with steps to skip while building project packages. Can contain following entries:
specs  Process packages specifics
docs  Try build documentation with roxygen
imps  Perform imports validation
tests  Run package tests
rcpp_atrribs  Run rppAttribs on the package
vignettes  Build package vignettes
(type: character(N), default: NULL).

Details

If not specified to skip RC it will detect revision version and tag packages before uploading. In that case, a check for changes in the project sources is performed for consistency and project packages will be rebuilt with version altered: revision will be added as the least number to package version.
Logs all messages onto the rsuite logger. Use logging::setLevel to control logs verbosity.
See Also

Other in repository management: repo_mng_init, repo_mng_list, repo_mng_remove, repo_mng_start, repo_mng_stop, repo_upload_bioc_package, repo_upload_ext_packages, repo_upload_github_package, repo_upload_package_files, repo_upload_pkgzip

Examples

```r
# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start src project
src_prj <- prj_start("my_project_src", skip_rc = TRUE, path = prj_base)

# create project package
prj_start_package("mypackage", prj = src_prj, skip_rc = TRUE)

# build project environment
prj_install_deps(prj = src_prj)

# start dest project
dst_prj <- prj_start("my_project_dst", skip_rc = TRUE, path = prj_base)

# set dest to use in project repository and CRAN
prj_config_set_repo_adapters(c("Dir", "CRAN"), prj = dst_prj)

# start managing in project repository
rmgr <- repo_mng_start("Dir", prj = dst_prj, ix = 1)

# upload mypackage from src into dest's in project repository
repo_upload_prj_packages(rmgr, prj = src_prj, skip_rc = TRUE)

# list available packages
repo_mng_list(rmgr)

# stop repository management
repo_mng_stop(rmgr)
```

RSuite

Supports Developing, Building and Deploying R Solutions.

Description

Supports safe and reproducible solutions development in R.

It will help you with environment separation per project, dependency management, local packages creation and preparing deployment packs for your solutions.
Package options

RSuite uses the following options to configure behavior:

- rsuite.usertempl_path: path to folder containing user customized templates. If not set (which is default) no user custom templates can be used.
- rsuite.cachepath: path to RSuite’s cache folder to store downloaded packages for later usage and content index of used repositories. If not set (which is default) no caching will be performed.

Project management

These functions will help you start a new RSuite project or package inside it, detect and install dependencies into the local environment, build your project packages and prepare deployment zip when you are done with the development.

- prj_start Creates project structure at the specified path.
- prj_start_package Creates package structure inside a project.
- prj_install_deps Installs project dependencies and needed supportive packages.
- prj_clean_deps Uninstalls unused packages from project local environment.
- prj_build Builds project internal packages and installs them.
- prj_zip Prepares deployment zip tagged with version.
- prj_pack Prepares project source pack tagged with version.
- prj_lock_env Locks the project environment.
- prj_unlock_env Unlocks the project environment.

Repository management

These functions make you able to manage package repositories. This RSuite built-in repository manager allows you to manage S3 based and local (in folder) repositories.

- repo_mng_start Starts management over the repository.
- repo_mng_init Initializes a repository (creates its structure).
- repo_mng_stop Stops management over the repository.
- repo_mng_list Retrieves the list of packages available in the repository.
- repo_mng_remove Removes packages from the repository.
- repo_upload_prj_packages Builds and uploads project package(s) into the repository.
- repo_upload_package_files Uploads package file(s) into a managed repository.
- repo_upload_ext_packages Uploads external packages into a managed repository.
- repo_upload_pkgzip Uploads PKGZIP into a managed repository.
- repo_upload_github_package Loads package from a GitHub repository.
- repo_upload_bioc_package Loads package from a BioConductor repository.
PKGZIP building

PKGZIPs are for management of repositories in an internet-less environment. There is often no internet access on corporate servers. In that case, you can prepare a PKGZIP with required packages somewhere with an internet connection and use it to update an internal CRAN-like repository which has no access to the internet.

pkgzip_build_prj_packages  Builds PKGZIP out of project packages.
pkgzip_build_package_files  Builds PKGZIP out of passed package files.
pkgzip_build_ext_packages  Builds PKGZIP out of passed external packages.
pkgzip_build_github_package  Builds PKGZIP out of a package on GitHub.
pkgzip_build_bioc_package  Builds PKGZIP out of a package on BioConductor.

Bash installer

You can create bash installer script to deploy in optimized (intelligent & parallel) way you project.

inst_wrap_zip  Wraps deployment zip into bash installer script.

RSuite miscellaneous

rsuite_check_version  Checks if a newer version of RSuite is available.
rsuite_update  Updates RSuite to the newest available version
rsuite_register_repo_adapter  Registers repository adapter to use for projects.
rsuite_get_repo_adapter_names  Gets all names of known repository adapters.
rsuite_register_rc_adapter  Registers RC (revision control) adapter to use for projects.
rsuite_unregister_rc_adapter  Unregisters RC (revision control) adapter.
rsuite_get_rc_adapter_names  Gets all names of known RC (revision control) adapters.
rsuite_get_logger  Retrieves RSuite logger.
rsuite_get_os_info  Retrieves information on current OS.

Template management

These functions will help you to manage RSuite templates. They allow you to create a project and package templates, register them in the local or global template directory and list all registered templates.

tmpl_start  Creates a new template.
tmpl_list_registered  Lists all registered templates
tmpl_register  Registers a template.
System requirements

Some packages have special system requirements declared. E.g. XML package on Linuxes requires the libxml2 system library to be installed. Such requirements are specified in free form in the SystemRequirements field in the package DESCRIPTION. The team from R Consortium (https://www.r-consortium.org/) performed a great job with collecting sysreqs database. As RSuite is supposed to work also in a connection-less environment the database they created is included in RSuite.

These functions extract system requirements for the whole project environment and make it possible to prepare installation scripts or update your system if you have privileged access.

**sysreqs_collect**  Prints out all system requirements from dependencies and project packages.
**sysreqs_check**  Checks for system requirements availability.
**sysreqs_install**  Updates the system to satisfy detected requirements.
**sysreqs_script**  Creates a script to update a system to satisfy project requirements.

Extending RSuite - RC adapter

This API allows you to implement your own RC (revision control) adapter for RSuite.
RSuite has SVN and Git adapters built-in for you.
After you developed your very own RC adapter you can register it in RSuite with the `rsuite_register_rc_adapter` function.

**rc_adapter_create_base**  Creates a base presentation for RC adapter to use by concrete implementations.
**rc_adapter_is_under_control**  Detects if directory is under adapter’s managed version control.
**rc_adapter_prj_struct_add**  Puts project structure under RC adapter’s managed version control.
**rc_adapter_pkg_struct_add**  Puts package structure under RC adapter’s managed version control.
**rc_adapter_get_version**  Retrieves current RC version number for working copy at the passed directory.
**rc_adapter_remove_admins**  Removes all RC related administrative entries from folder tree at the directory.

Extending RSuite - Repository adapter and manager

This API allows you to implement your own repository adapter for RSuite. If the repository can be managed (you can add/remove/update packages in it) you can provide a repo manager object creation ability to manage it with RSuite.
RSuite has CRAN, MRAN, S3(Amazon S3 bucket base repository), Url(repository under Url) and Dir(local CRAN-like folder) repo adapters and Dir and S3 repo managers built-in for you.
After you develop your very own repository adapter you can register it in RSuite with the `rsuite_register_repo_adapter` function.

**repo_adapter_create_base**  Creates the base presentation for the repo adapter to use by concrete implementations.
repo_adapter_get_info  Returns information about the repository the adapter is working on.
rc_adapter_prj_struct_add  Puts the project structure under RC adapter’s managed version control.
repo_adapter_get_path  Returns adapter path related to project to use for dependencies resolution.
repo_adapter_create_manager  Creates repo manager to manage its repository.
repo_manager_get_info  Returns information on repo manager.
repo_manager_init  Initializes managed repository structure.
repo_manager_upload  Adds packages to the managed repository.
repo_manager_remove  Removes specified packages from the repository.
repo_manager_destroy  Releases resources allocated to manage the repository.

**Project access/loading/unloading**

You normally will not need to use these functions unless you want to perform some scripting with use of RSuite.

prj_init  Loads project settings without loading it into the environment.
prj_load  Loads project into the environment so all master scripts can run.
prj_unload  Unloads last loaded project.

**Project configuration**

These functions are a convenient way to change the project global configuration.

You normally will not need to use these functions unless you want to perform some scripting with use of RSuite.

prj_config_set_repo_adapters  Updates the project configuration to use only specified repository adapters.
prj_config_set_rversion  Updates the project configuration to use specified R Version.

---

**rsuite_check_version**

Checks if a newer version of RSuite is available.

**Description**

Checks if a newer version of RSuite is available.

**Usage**

`rsuite_check_version()`

**Value**

NULL if a newer version is not available or newest available version number.
rsuite_getLogger

See Also

Other miscellaneous: rsuite_getLogger, rsuite_get_ci_adapter_names, rsuite_get_os_info, rsuite_get_rc_adapter_names, rsuite_get_repo_adapter_names, rsuite_register_ci_adapter, rsuite_register_rc_adapter, rsuite_register_repo_adapter, rsuite_unregister_ci_adapter, rsuite_unregister_rc_adapter, rsuite_unregister_repo_adapter, rsuite_update, tmpl_register

Examples

```c
# print latest version available or NULL if latest is currently installed
rsuite_check_version()
```

---

### rsuite_getLogger

Retrieves RSuite logger.

---

Description

Retrieves RSuite logger.

Usage

rsuite_getLogger()

Value

logger object

See Also

Other miscellaneous: rsuite_check_version, rsuite_get_ci_adapter_names, rsuite_get_os_info, rsuite_get_rc_adapter_names, rsuite_get_repo_adapter_names, rsuite_register_ci_adapter, rsuite_register_rc_adapter, rsuite_register_repo_adapter, rsuite_unregister_ci_adapter, rsuite_unregister_rc_adapter, rsuite_unregister_repo_adapter, rsuite_update, tmpl_register

Examples

```cpp
logging::loginfo("This is an INFO from RSuite", logger = rsuite_getLogger())
```
**rsuite_get_ci_adapter_names**

*Gets all names of known CI (continuous integration) adapters.*

---

**Description**

Gets all names of known CI (continuous integration) adapters.

**Usage**

```r
rsuite_get_ci_adapter_names()
```

**Value**

names of registered ci adapters as character vector.

**See Also**

Other miscellaneous: `rsuite_check_version`, `rsuite_get_logger`, `rsuite_get_os_info`, `rsuite_get_rc_adapter_names`, `rsuite_get_repo_adapter_names`, `rsuite_register_ci_adapter`, `rsuite_register_rc_adapter`, `rsuite_register_repo_adapter`, `rsuite_unregister_ci_adapter`, `rsuite_unregister_rc_adapter`, `rsuite_unregister_repo_adapter`, `rsuite_update`, `tmpl_register`

**Examples**

```r
rsuite_get_ci_adapter_names()
```

---

**rsuite_get_os_info**

*Retrieves information on current OS.*

---

**Description**

Retrieves information on current OS.

**Usage**

```r
rsuite_get_os_info()
```
rsuite_get_rc_adapter_names

Value

named list with following items

type One of windows, macos, unix. (type: character)
platform One of Windows, MacOS, SunOS, RedHat, Debian. (type: character(1))
release One of Solaris, MacOS, Ubuntu, Debian, Fedora, CentOS or RedHat or NA. (type: character(1))
distrib Distribution release e.g. for Debian: squeeze, wheezy, jessie. (type: character(1))
version Version number of the distribution. (type: character(1))

See Also

Other miscellaneous: rsuite_check_version, rsuite_get_logger, rsuite_get_ci_adapter_names, rsuite_get_rc_adapter_names, rsuite_get_repo_adapter_names, rsuite_register_ci_adapter, rsuite_register_rc_adapter, rsuite_register_repo_adapter, rsuite_unregister_ci_adapter, rsuite_unregister_rc_adapter, rsuite_unregister_repo_adapter, rsuite_update, tmpl_register

Examples

rsuite_get_os_info()

rsuite_get_rc_adapter_names

*Gets all names of known RC (revision control) adapters.*

Description

Gets all names of known RC (revision control) adapters.

Usage

rsuite_get_rc_adapter_names()

Value

names of registered rc adapters as character vector.

See Also

Other miscellaneous: rsuite_check_version, rsuite_get_logger, rsuite_get_ci_adapter_names, rsuite_get_os_info, rsuite_get_repo_adapter_names, rsuite_register_ci_adapter, rsuite_register_rc_adapter, rsuite_register_repo_adapter, rsuite_unregister_ci_adapter, rsuite_unregister_repo_adapter, rsuite_unregister_rc_adapter, rsuite_update, tmpl_register

Examples

rsuite_get_rc_adapter_names()
rsuite_get_repo_adapter_names

*Gets all names of known repository adapters.*

---

### Description

Gets all names of known repository adapters.

### Usage

```r
rsuite_get_repo_adapter_names()
```

### Value

names of registered repository management adapters as character vector.

### See Also

Other miscellaneous: `rsuite_check_version`, `rsuite_getLogger`, `rsuite_get_ci_adapter_names`, `rsuite_get_os_info`, `rsuite_get_rc_adapter_names`, `rsuite_register_ci_adapter`, `rsuite_register_rc_adapter`, `rsuite_register_repo_adapter`, `rsuite_unregister_ci_adapter`, `rsuite_unregister_rc_adapter`, `rsuite_unregister_repo_adapter`, `rsuite_update`, `tmpl_register`

### Examples

```r
rsuite_get_repo_adapter_names()
```

---

rsuite_register_ci_adapter

*Registers CI (continuous integration) adapter to use for projects.*

---

### Description

Registers CI (continuous integration) adapter to use for projects.

### Usage

```r
rsuite_register_ci_adapter(ci_adapter)
```

### Arguments

- `ci_adapter`: object complying rsuite_ci_adapter signature.
rsuite_register_rc_adapter

See Also

Other miscellaneous: rsuite_check_version, rsuite_getLogger, rsuite_get_ci_adapter_names, rsuite_get_os_info, rsuite_get_rc_adapter_names, rsuite_get_repo_adapter_names, rsuite_register_ci_adapter, rsuite_register_repo_adapter, rsuite_unregister_ci_adapter, rsuite_unregister_rc_adapter, rsuite_unregister_repo_adapter, rsuite_update, tmpl_register

Examples

ci_adapter <- ci_adapter_create_base("Own") # create your custom adapter
class(ci_adapter) <- c("ci_adapter_own", class(ci_adapter))

# register it
rsuite_register_ci_adapter(ci_adapter)

# unregister it
rsuite_unregister_ci_adapter("Own")

rsuite_register_rc_adapter

Registers RC (revision control) adapter to use for projects.

Description

Registers RC (revision control) adapter to use for projects.

Usage

rsuite_register_rc_adapter(rc_adapter)

Arguments

rc_adapter object complying rsuite_rc_adapter signature.

See Also

Other miscellaneous: rsuite_check_version, rsuite_getLogger, rsuite_get_ci_adapter_names, rsuite_get_os_info, rsuite_get_rc_adapter_names, rsuite_get_repo_adapter_names, rsuite_register_ci_adapter, rsuite_register_repo_adapter, rsuite_unregister_ci_adapter, rsuite_unregister_rc_adapter, rsuite_unregister_repo_adapter, rsuite_update, tmpl_register

Examples

rc_adapter <- rc_adapter_create_base("Own") # create your custom adapter
class(rc_adapter) <- c("rc_adapter_own", class(rc_adapter))

# register it
rsuite_register_rc_adapter(rc_adapter)
rsuite_register_repo_adapter

Registers repository adapter to use for projects.

Description

Registers repository adapter to use for projects.

Usage

rsuite_register_repo_adapter(repo_adapter)

Arguments

repo_adapter object complying rsuite_repo_adapter signature.

See Also

Other miscellaneous: rsuite_check_version, rsuite_getLogger, rsuite_get_ci_adapter_names, rsuite_get_os_info, rsuite_get_rc_adapter_names, rsuite_get_repo_adapter_names, rsuite_register_ci_adapter, rsuite_register_rc_adapter, rsuite_unregister_ci_adapter, rsuite_unregister_rc_adapter, rsuite_unregister_repo_adapter, rsuite_update, tmpl_register

Examples

repo_adapter <- repo_adapter_create_base("Own") # create your custom adapter
class(repo_adapter) <- c("repo_adapter_own", class(repo_adapter))

# register it
rsuite_register_repo_adapter(repo_adapter)

# unregister it
rsuite_unregister_repo_adapter("Own")
---

**rsuite_unregister_ci_adapter**

*Unregisters CI (continuous integration) adapter.*

---

**Description**

Unregisters CI (continuous integration) adapter.

**Usage**

```bash
rsuite_unregister_ci_adapter(name)
```

**Arguments**

- `name` CI adapter name to unregister.

**See Also**

Other miscellaneous: `rsuite_check_version`, `rsuite_getLogger`, `rsuite_get_ci_adapter_names`, `rsuite_get_os_info`, `rsuite_get_rc_adapter_names`, `rsuite_get_repo_adapter_names`, `rsuite_register_ci_adapter`, `rsuite_register_rc_adapter`, `rsuite_register_repo_adapter`, `rsuite_unregister_rc_adapter`, `rsuite_unregister_repo_adapter`, `rsuite_update`, `tmpl_register`

**Examples**

```bash
ci_adapter <- ci_adapter_create_base("Own")  # create your custom adapter
class(ci_adapter) <- c("ci_adapter_own", class(ci_adapter))

# register it
rsuite_register_ci_adapter(ci_adapter)

# unregister it
rsuite_unregister_ci_adapter("Own")
```

---

**rsuite_unregister_rc_adapter**

*Unregisters RC (revision control) adapter.*

---

**Description**

Unregisters RC (revision control) adapter.

**Usage**

```bash
rsuite_unregister_rc_adapter(name)
```
**Arguments**

*name*  
RC adapter name to unregister.

**See Also**

Other miscellaneous: `rsuite_check_version`, `rsuite_getLogger`, `rsuite_get_ci_adapter_names`, `rsuite_get_os_info`, `rsuite_get_rc_adapter_names`, `rsuite_get_repo_adapter_names`, `rsuite_register_ci_adapter`, `rsuite_register_rc_adapter`, `rsuite_register_repo_adapter`, `rsuite_unregister_ci_adapter`, `rsuite_unregister_repo_adapter`, `rsuite_update`, `tmpl_register`

**Examples**

```r
rc_adapter <- rc_adapter_create_base("Own")  # create your custom adapter
class(rc_adapter) <- c("rc_adapter_own", class(rc_adapter))

# register it
rsuite_register_rc_adapter(rc_adapter)

# unregister it
rsuite_unregister_rc_adapter("Own")
```

---

**rsuite_unregister_repo_adapter**

*Unregisters repository adapter.*

**Description**

Unregisters repository adapter.

**Usage**

```r
rsuite_unregister_repo_adapter(repos_adapter_name)
```

**Arguments**

*repo_adapter_name*  
name of the repo adapter to unregister. (type: character(1))

**See Also**

Other miscellaneous: `rsuite_check_version`, `rsuite_getLogger`, `rsuite_get_ci_adapter_names`, `rsuite_get_os_info`, `rsuite_get_rc_adapter_names`, `rsuite_get_repo_adapter_names`, `rsuite_register_ci_adapter`, `rsuite_register_rc_adapter`, `rsuite_register_repo_adapter`, `rsuite_unregister_ci_adapter`, `rsuite_unregister_repo_adapter`, `rsuite_update`, `tmpl_register`
Examples

repo_adapter <- repo_adapter_create_base("Own")  # create your custom adapter
class(repo_adapter) <- c("repo_adapter_own", class(repo_adapter))

# register it
rsuite_register_repo_adapter(repo_adapter)

# unregister it
rsuite_unregister_repo_adapter("Own")

---

rsuite_update  Updates RSuite to newest available version.

Description

Updates RSuite to newest available version.

Usage

rsuite_update(lib.dir = Sys.getenv("R_LIBS_USER"))

Arguments

lib.dir  folder path to install RSuite into. Folder must exist. (type: character(1); default: Sys.getenv("R_LIBS_USER"))

Value

TRUE if updated (invisible).

See Also

Other miscellaneous: rsuite_check_version, rsuite_getLogger, rsuite_get_ci_adapter_names, rsuite_get_os_info, rsuite_get_rc_adapter_names, rsuite_get_repo_adapter_names, rsuite_register_ci_adapter, rsuite_register_rc_adapter, rsuite_register_repo_adapter, rsuite_unregister_ci_adapter, rsuite_unregister_rc_adapter, rsuite_unregister_repo_adapter, tmpl_register

Examples

lib_dir <- tempfile("Rsuite")
dir.create(lib_dir, recursive = TRUE, showWarnings = FALSE)

rsuite_update(lib_dir)
sysreqs_check

Checks for system requirements availability.

Description
Collects system requirements with `sysreqs_collect` and performs checks for their existence. Will fail if some system requirements are not satisfied.

Usage
```haskell
sysreqs_check(prj = NULL)
```

Arguments
- `prj` project object to check sys requirements for. If not passed the loaded project will be used or the default whichever exists. Will init default project from the working directory if no default project exists. (type: rsuite_project, default: NULL)

See Also
Other in SYSREQS: `sysreqs_collect`, `sysreqs_install`

Examples
```haskell
# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# add dependency to XML
write("library(XML)",
      file = file.path(prj$path, "R", "master.R"),
      append = TRUE)

# check if requirements or XML are satisfied
sysreqs_check(prj)
```
sysreqs_collect

Description
Prints out all system requirements from dependencies and project packages.

Usage
sysreqs_collect(prj = NULL)

Arguments
prj project object to collect sys requirements for. If not passed the loaded project will be used or the default whichever exists. Will init the default project from working directory if no default project exists. (type: rsuite_project, default: NULL)

Value
named list with package names and containing system requirements as value.

See Also
Other in SYSREQS: sysreqs_check, sysreqs_install

Examples

# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# add package to the project
prj_start_package("mypackage", prj = prj)

# add system requirements specification
write("SystemRequirements: some requirement",
      file = file.path(prj$path, "packages", "mypackage", "DESCRIPTION"),
      append = TRUE)

# list content of pkgzip created
sysreqs_collect(prj)
sysreqs_install

Updates system to satisfy detected requirements.

Description

Collects system requirements with sysreqs_collect and builds/installs them.

Usage

sysreqs_install(prj = NULL)

Arguments

prj  
project object to handle sys requirements for. If not passed the loaded project will be used or the default whichever exists. Will init default project from the working directory if no default project exists. (type: rsuite_project, default: NULL)

See Also

Other in SYSREQS: sysreqs_check, sysreqs_collect

Examples

# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# add dependency to XML
write("library(XML)",
    file = file.path(prj$path, "R", "master.R"),
    append = TRUE)

# check if requirements of XML are satisfied
sysreqs_install(prj)
sysreqs_script  Creates a script to update the system to satisfy project requirements.

Description

Collects system requirements with sysreqs_collect and creates a script to build/install them. It creates a .cmd script for Windows and a bash script for Linuxes.

Usage

sysreqs_script(prj = NULL)

Arguments

prj  project object to process sys requirements for. If not passed the loaded project will be used or the default whichever exists. Will init default project from working directory if no default project exists. (type: rsuite_project, default: NULL)

Value

invisible path to script file created or NULL if no system requirements detected.

Examples

```r
# create exemplary project base folder
prj_base <- tempfile("example_")
dir.create(prj_base, recursive = TRUE, showWarnings = FALSE)

# start project
prj <- prj_start("my_project", skip_rc = TRUE, path = prj_base)

# add dependency to XML
write("library(XML)",
      file = file.path(prj$path, "R", "master.R"),
      append = TRUE)

# generate script
sysreqs_fpath <- sysreqs_script(prj)

# present script contents
cat(readLines(sysreqs_fpath), sep = "\n")
```
tmpl_list_registered  Returns all available project/package templates

Description

Returns all available project/package templates

Usage

tmpl_list_registered()

Details

Project templates have to include a PARAMETERS file Package templates have to include the following files: DESCRIPTION

All templates can be found in folder pointed by rsuite.user_templ_path option.

Value

names of the registered project and package templates together with their file path

See Also

Other in templates management: tmpl_start

Examples

tmpl_list_registered()

tmpl_register  Registers the template specified with the path argument.

Description

Registers the template specified with the path argument.

Usage

tmpl_register(path = NULL, global = FALSE)
Arguments

path path to the directory where the template should be created (type: character, de-
default: NA)

global flag specifying if the template will be registered in the user’s local template
directory (taken from rsuite.user_templ_path) or in the global template directory
(/etc/.rsuite/templates on Linux platforms)

Details

All templates have specific requirements: Project templates have to contain a PARAMETERS file. Package templates have to contain a DESCRIPTION file.

The user’s local template directory is taken from the rsuite.user_templ_path option. The global template is specified as '/etc/.rsuite/templates' and only concerns Linux platforms

See Also

Other miscellaneous: rsuite_check_version, rsuite_getLogger, rsuite_get_ci_adapter_names, rsuite_get_os_info, rsuite_get_rc_adapter_names, rsuite_get_repo_adapter_names, rsuite_register_ci_adapter, rsuite_register_rc_adapter, rsuite_register_repo_adapter, rsuite_unregister_ci_adapter, rsuite_unregister_rc_adapter, rsuite_unregister_repo_adapter, rsuite_update

Examples

# setup
old_option_value <- getOption("rsuite.user_templ_path")
tmpl_dir <- tempfile("user_templates_")
dir.create(tmpl_dir, recursive = TRUE, showWarnings = FALSE)

options(rsuite.user_templ_path = tmpl_dir)
user_templ <- tempfile("usr_templ_")

# initialize template from builtin
tmpl_start(basename(user_templ), path = tempdir())
# register it
tmpl_register(user_templ)

# clean up
options(rsuite.user_templ_path = old_option_value)
unlink(tmpl_dir, recursive = TRUE, force = TRUE)
unlink(user_templ, recursive = TRUE, force = TRUE)
tmpl_start

Creates a new template with the specified name, in the specified path.

Description

Creates a new template with the specified name, in the specified path.

Usage

```r
tmpl_start(name, path = getwd(), add_prj = TRUE, add_pkg = TRUE,
            base_tmpl = "builtin")
```

Arguments

- `name`: name of the template being created. (type: character(1))
- `path`: path to the directory where the template should be created. If NULL will use the folder with user template. (type: character(1), default: getwd())
- `add_prj`: if TRUE include project template to the template directory
- `add_pkg`: if TRUE include package template in the template directory
- `base_tmpl`: name of the package and/or project template (or path to it) to use for template creation. (type: character(1); default: builtin).

Details

Project templates are required to include a PARAMETERS file whereas package templates are required to include a DESCRIPTION file

If there is no path argument provided. The function will create the template in working directory.

See Also

Other in templates management: `tmpl_list_registered`

Examples

```r
tmpl_dir <- tempfile("templ_")
tmpl_start(basename(tmpl_dir), path = tempdir())
```
Index

build_bash_script, 3
build_win_script, 4

ci_adapter_create_base, 5, 6, 7
ci_adapter_get_version, 5, 7
ci_adapter_is_building, 5, 6, 6

get_version_numbers, 7
inst_wrap_zip, 8

options, 60

perform, 8
pkgzip_build_bioc_package, 9, 11, 13–15, 61
pkgzip_build_ext_packages, 10, 10, 13–15, 61
pkgzip_build_github_package, 10, 11, 12, 14, 15, 61
pkgzip_build_package_files, 10, 11, 13, 13, 15, 61
pkgzip_build_prj_packages, 10, 11, 13, 14, 14, 61
prj_build, 16, 17, 21–25, 27, 28, 30, 60
prj_clean_deps, 16, 17, 21–25, 27, 28, 30, 60
prj_config_set_repo_adapters, 18, 20, 63
prj_config_set_rversion, 19, 19, 63
prj_init, 16, 17, 20, 22–25, 27, 28, 30, 63
prj_install_deps, 16, 17, 21, 21, 23–25, 27, 28, 30, 60
prj_load, 16, 17, 21, 22, 22, 24, 25, 27, 28, 30, 63
prj_lock_env, 16, 17, 21–23, 23, 25, 27–30, 60
prj_pack, 16, 17, 21–24, 24, 27, 28, 30, 60
prj_start, 16, 17, 21–25, 26, 28, 30, 60
prj_start_package, 16, 17, 21–25, 27, 27, 28, 30, 60
prj_unload, 16, 17, 21–25, 27, 28, 30, 63
prj_unlock_env, 29, 60

prj_zip, 16, 17, 21–25, 27, 28, 30, 60
rc_adapter_create_base, 31, 32–36, 62
rc_adapter_get_version, 31, 32, 33–36, 62
rc_adapter_is_under_control, 31, 32, 33, 34–36, 62
rc_adapter_pkg_struct_add, 31–33, 34, 35, 36, 62
rc_adapter_prj_struct_add, 31–34, 35, 36, 62, 63
rc_adapter_remove_admins, 31–35, 35, 62
repo_adapter_create_base, 36, 37–41, 43–45, 62
repo_adapter_create_manager, 37, 37, 38–41, 43–45, 63
repo_adapter_get_info, 37, 38, 39–41, 43–45, 63
repo_adapter_get_path, 37, 38, 39, 40, 41, 43–45, 63
repo_manager_destroy, 37–39, 40, 41, 43–45, 63
repo_manager_get_info, 37–40, 41, 43–45, 63
repo_mng_init, 45, 47–50, 52, 53, 55–57, 59, 60
repo_mng_list, 46, 46, 48–50, 52, 53, 55–57, 59, 60
repo_mng_remove, 46, 47, 49, 50, 52, 53, 55–57, 59, 60
repo_mng_start, 46–48, 49, 50, 52, 53, 55–57, 59, 60
repo_mng_stop, 46–49, 50, 52, 53, 55–57, 59, 60
repo_upload_bioc_package, 46–50, 51, 53, 55–57, 59, 60
repo_upload_ext_packages, 46–50, 52, 52, 55–57, 59, 60
repo_upload_github_package, 46–50, 52, 53, 54, 56, 57, 59, 60
repo_upload_package_files, 46–50, 52, 53, 55, 57, 59, 60
repo_upload_pkgzip, 46–50, 52, 53, 55, 56, 56, 59, 60
repo_upload_prj_packages, 46–50, 52, 53, 55–57, 58, 60
RSuite, 59
RSuite-package (RSuite), 59
rsuite_check_version, 61, 63, 64–72, 78
rsuite_get_ci_adapter_names, 64, 65, 66–72, 78
rsuite_get_os_info, 61, 64, 65, 66–72, 78
rsuite_get_rc_adapter_names, 61, 64–66, 66, 67–72, 78
rsuite_get_repo_adapter_names, 61, 64–66, 67, 68–72, 78
rsuite_getLogger, 61, 64, 64–57, 78
rsuite_register_ci_adapter, 64–67, 67, 68–72, 78
rsuite_register_rc_adapter, 61, 62, 64–68, 68, 69–72, 78
rsuite_register_repo_adapter, 61, 62, 64–68, 69, 70–72, 78
rsuite_unregister_ci_adapter, 64–69, 70, 71, 72, 78
rsuite_unregister_rc_adapter, 61, 64–70, 70, 71, 72, 78
rsuite_unregister_repo_adapter, 64–71, 71, 72, 78
rsuite_update, 61, 64–71, 72, 78
sysreq_check, 62, 73, 74, 75
sysreqs_collect, 62, 73, 74, 75, 76
sysreqs_install, 62, 73, 74, 75
sysreqs_script, 62, 76
tmpl_list_registered, 61, 77, 79
tmpl_register, 61, 64–72, 77
tmpl_start, 61, 77, 79