Package ‘D4TAlink.light’

October 22, 2022

Type Package
Title FAIR Data - Workflow Management
Version 2.1.11
Date 2022-10-19
Author Gregoire Thomas [aut, cre] (<https://orcid.org/0000-0002-6247-9438>)
Maintainer Gregoire Thomas <gregoire.thomas@SQU4RE.com>
Description Tools, methods and processes for the management of analysis workflows. These lightweight solutions facilitate structuring R&D activities. These solutions were developed to comply with FAIR principles as discussed by Jacobsen et al. (2017) <doi:10.1162/dint_r_00024>, and with ALCOA+ principles as proposed by the U.S. FDA.
Depends R (>= 4.0.0), jsonlite, rmarkdown, openssl, utils
License GPL-3
Suggests roxygen2 (>= 3.1.0), testthat, usethis, knitr, WriteXLS, XLConnect, openxlsx, xlsx, officedown, Biobase, getPass
Encoding UTF-8
RoxygenNote 7.2.1
VignetteBuilder knitr
ByteCompile true
BugReports https://bitbucket.org/SQ4/d4talink.light/issues
NeedsCompilation no
Repository CRAN
Date/Publication 2022-10-22 10:07:50 UTC

R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Number of Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>archiveTask</td>
<td>3</td>
</tr>
<tr>
<td>binaryDir</td>
<td>4</td>
</tr>
</tbody>
</table>
archiveTask

Create an archive containing the files of a given task.

Description

Create an archive containing the files of a given task.

Usage

archiveTask(task, file, overwrite = FALSE, ...)

Arguments

task Object of class D4TAlinkTask, as created by initTask.
file full name of the output zip file
overwrite overwrite the output zip file is it exists
... Arguments passed on to utils::zip
zipfile The pathname of the zip file: tilde expansion (see path.expand) will be performed.
files A character vector of recorded filepaths to be included.
flags A character string of flags to be passed to the command: see 'Details'.
extras An optional character vector: see 'Details'.
zip A character string specifying the external command to be used.

Value

the archive file name invisibly.
binaryDir  

*Get path of binary directory.*  

**Description**  
Get path of binary directory.

**Usage**  

```r  
binaryDir(task, subdir = NULL, dirCreate = TRUE)  
```  

**Arguments**  
- **task** Object of class `D4TAlinkTask`, as created by `initTask`.
- **subdir** (optional) Subdirectory.
- **dirCreate** Logical, if TRUE (by default) the directory is created.

**Value**  
File path.

binaryFn  

*Get path of binary file.*  

**Description**  
Get path of binary file.

**Usage**  

```r  
binaryFn(task, type, ext = "rds", subdir = NULL, dirCreate = TRUE)  
```  

**Arguments**  
- **task** Object of class `D4TAlinkTask`, as created by `initTask`.
- **type** Filename type. If the type is an array, the concatenation of the elements is used with separator"\". Filenames have the form [task name]_[type].[ext]
- **ext** Filename extension.
- **subdir** (optional) Subdirectory.
- **dirCreate** Logical, if TRUE (by default) the directory is created.

**Value**  
File path.
**Description**

Output R object using function `cat`.

**Usage**

```r
catReport(
  x,
  task,
  type,
  ext = "txt",
  subdir = NULL,
  dirCreate = TRUE,
  sep = "\n",
  eof = "\n",
  ...
)
```

**Arguments**

- `x` R object to output.
- `task` Object of class `D4AlinkTask`, as created by `initTask`.
- `type` Filename type. If the type is an array, the concatenation of the elements is used with separator "-". Filenames have the form `[task name]_[type].[ext]`.
- `ext` Filename extension.
- `subdir` (optional) Subdirectory.
- `dirCreate` Logical, if TRUE (by default) the directory is created.
- `sep` separator
- `eof` EOF
- `...` Arguments passed on to `base::cat`

**Value**

the file name invisibly.
---

### D4getenv

*Get env var.*

#### Description

Get env var.

#### Usage

```r
d4getenv(name, desc, quiet = FALSE)
```

#### Arguments

- **name**: name of variable
- **desc**: description of variable
- **quiet**: suppress error messages, default=FALSE.

#### Value

Environment variable.

---

### D4TAlink-common-args

*Arguments used across the functions of the D4TAlink package.*

#### Description

Arguments used across the functions of the D4TAlink package.

#### Arguments

- **project**: Project name.
- **package**: Package name.
- **taskname**: Task name.
- **author**: Author name, system username by default.
- **sponsor**: Sponsor name, default set by `setTaskSponsor`.
- **rootpath**: Path of the task repository, default set by `setTaskRoot`.
- **task**: Object of class `D4TAlinkTask`, as created by `initTask`.
- **type**: Filename type. If the type is an array, the concatenation of the elements is used with separator".". Filenames have the form [task name]_[type].[ext]
- **ext**: Filename extension.
- **dirType**: Directory type, e.g. 'bin' or 'data' or 'doc'.
- **subdir**: (optional) Subdirectory.
- **dirCreate**: Logical, if TRUE (by default) the directory is created.
- **pathgen**: optional function returning a list of paths, currently `pathsGLPG` or `pathsPMS`. 
DataLinkTask

Value

No return value, used for the documentation of the functions of the package.

Description

The DataLinkTask object is created by the initTask function. This object is a list containing the task properties:

- task: task name
- package: package name
- project: project name
- sponsor: sponsor name
- author: author name
- copyright: copyright, by default 'Copyright (c) [sponsor] [year]'
- 'date': date of the task initialization, formatted as 'year-month-day'
- 'footer': footer for the task, e.g. 'Copyright (c) [sponsor] [year] - CONFIDENTIAL'
- 'version': string with task version, '0.0' at the initialization
- dependencies: information on R versions and names of loaded/attached dependencies and corresponding versions

There are different functions dedicated for this DataLinkTask object:

- taskID: Get ID

Value

Not relevant

Examples

```r
## Not run:
# set D4TAlink's global parameters
setTaskAuthor("Doe Johns")
setTaskSponsor("mySponsor")
stop("STOPHERE")

# Create data repository
setTaskRoot(file.path(tempdir(),"D4TAlink_example001"),dirCreate=TRUE)

# Create a task
task <- initTask(project="myProject",
                 package="myPackage",
                 taskname=sprintf("%s_myTask",format(Sys.time(),"%Y%m%d")))
```
# Output a plot to a PDF file
file <- pdfReport(task,c("plots",1),dim=c(100,100))
opa <- par()$task
par(task=FALSE)
hist(rnorm(100))
par(task=opa)
dev.off()
# View the plot:
utils::browseURL(file)

# Output tables to an Excel file
d <- list(letters=data.frame(a=LETTERS,b=letters,c=1:length(letters)),
    other=data.frame(a=1:3,b=11:13))
file <- saveReportXls(d,task,"table")
utils::browseURL(file)

# Save an R object to a binary file
saveBinary(d,task,"data")
e <- readBinary(task,"data")
if(!all(names(e)%in%names(d))) stop("error [1]")

# Create and render R markdown file
initTaskRmd(task,overwrite=TRUE)
file <- renderTaskRmd(task) # requires having run 'tinytex::install_tinytex()
utils::browseURL(file)

# Delete new data repository
unlink(getTaskRoot(),recursive=TRUE)

## End(Not run)

datasourceDir

Get path of data source directory.

Description

Get path of data source directory.

Usage

datasourceDir(task, subdir = NULL, dirCreate = TRUE)

Arguments

task Object of class D4TAlinkTask, as created by initTask.
subdir (optional) Subdirectory.
dirCreate Logical, if TRUE (by default) the directory is created.
**datasourceFn**  
*Get path of data source file.*

**Description**  
Get path of data source file.

**Usage**  
```r
 datasourceFn(task, filename, subdir = ".", dirCreate = TRUE)
```

**Arguments**
- **task**: Object of class `D4TAlinkTask`, as created by `initTask`.
- **filename**: Name of the input file.
- **subdir**: (optional) Subdirectory.
- **dirCreate**: Logical, if TRUE (by default) the directory is created.

**Value**  
File path.

---

**docDir**  
*Get path of documentation directory.*

**Description**  
Get path of documentation directory.

**Usage**  
```r
 docDir(task, subdir = NULL, dirCreate = TRUE)
```

**Arguments**
- **task**: Object of class `D4TAlinkTask`, as created by `initTask`.
- **subdir**: (optional) Subdirectory.
- **dirCreate**: Logical, if TRUE (by default) the directory is created.

**Value**  
File path.
docFn

*Get path of documentation file.*

**Description**

Get path of documentation file.

**Usage**

```
docFn(task, type, ext, subdir = NULL, dirCreate = TRUE)
```

**Arguments**

- `task`: Object of class `D4TAlinkTask`, as created by `initTask`.
- `type`: Filename type. If the type is an array, the concatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
- `ext`: Filename extension.
- `subdir`: (optional) Subdirectory.
- `dirCreate`: Logical, if TRUE (by default) the directory is created.

**Value**

File path.

---

DTx

*Generic function.*

**Description**

Generic function.

**Usage**

```
DTx(sponsor = getTaskSponsor(), task = NULL)
```

**Arguments**

- `sponsor`: Sponsor name, default set by `setTaskSponsor`.
- `task`: Object of class `D4TAlinkTask`, as created by `initTask`.

**Value**

NULL.
formatTaskDocx  Replace default task fields in 'docx' file.

**Description**
Replace default task fields in 'docx' file.

**Usage**
formatTaskDocx(task, ifn)

**Arguments**
- **task** Object of class D4TAlinkTask, as created by initTask.
- **ifn** input file name.

**Value**
the file name invisibly.

getTaskAuthor  Get the name of the task author.

**Description**
Get the name of the task author.

**Usage**
getTaskAuthor(quiet = FALSE)

**Arguments**
- **quiet** suppress error messages, default=FALSE.

**Value**
The current name of the task author.

**Examples**
getTaskAuthor(quiet=TRUE)
getTaskEnckey  

Get the encryption key for binary data files.

**Description**

Get the encryption key for binary data files.

**Usage**

getTaskEnckey(ask = FALSE)

**Arguments**

- **ask**  
  query encryption key from user, default FALSE.

**Value**

The encryption key invisibly.

getTaskFilepath  

Get the path of a file.

**Description**

Get the path of a file.

**Usage**

getTaskFilepath(task, type, ext, dirtype, subdir = NULL, dirCreate = TRUE)

**Arguments**

- **task**  
  Object of class D4TAlinkTask, as created by initTask.
- **type**  
  Filename type. If the type is an array, the concatenation of the elements is used with separator ":". Filenames have the form [task name]_[type].[ext]
- **ext**  
  Filename extension.
- **dirtype**  
  task directory where file is stored, i.e., 'documentation', 'code', 'data', 'data source' or 'binary data'.
- **subdir**  
  (optional) Subdirectory.
- **dirCreate**  
  Logical, if TRUE (by default) the directory is created.

**Value**

Full path to file.
**getTaskPaths**

Get the paths of the task.

**Description**

Get the paths of the task.

**Usage**

```r
getTaskPaths(task)
```

**Arguments**

- `task` Object of class `D4TAlinkTask`, as created by `initTask`.

**Value**

List of task’s paths.

---

**getTaskRmdTemplate**

Get the path to the Rmd task template.

**Description**

Get the path to the Rmd task template.

**Usage**

```r
getTaskRmdTemplate(quiet = FALSE)
```

**Arguments**

- `quiet` suppress error messages, default=FALSE.

**Value**

The path to the Rmd task template.
getTaskRoot

Get the root of the task repository.

**Description**
Get the root of the task repository.

**Usage**
getTaskRoot(quiet = FALSE)

**Arguments**
- **quiet** suppress error messages, default=FALSE.

**Value**
Path to the current task root.

**Examples**
getTaskRoot(quiet=TRUE)

getTaskRscriptTemplate

Get the path to the R script task template.

**Description**
Get the path to the R script task template.

**Usage**
getTaskRscriptTemplate(quiet = FALSE)

**Arguments**
- **quiet** suppress error messages, default=FALSE.

**Value**
The path to the R script task template.
**getTaskSponsor**

Get the name of the task sponsor.

**Usage**

```r
getTaskSponsor(quiet = FALSE)
```

**Arguments**

- `quiet` suppress error messages, default=FALSE.

**Value**

The current name of the task sponsor.

**Examples**

```r
getTaskSponsor(quiet=TRUE)
```

---

**getTaskStructure**

Get repository directory structure.

**Usage**

```r
getTaskStructure(quiet = FALSE)
```

**Arguments**

- `quiet` suppress error messages, default=FALSE.

**Value**

The directory structure function.
initTask

Initialize a task

Description

During the initialization:

- The folder structure for the task is created in the data repository.
- The task properties are also saved in rds and json format.

Please note that it is recommended to load packages for your analysis before initializing the task.

Usage

initTask(
  project,
  package,
  taskname,
  sponsor = getTaskSponsor(),
  author = getTaskAuthor(),
  dirCreate = TRUE,
  templateCreate = FALSE,
  overwrite = FALSE
)

Arguments

project Project name.
package Package name.
taskname Task name.
sponsor Sponsor name, default set by setTaskSponsor.
author Author name, system username by default.
dirCreate logical, if TRUE (by default) the directory structure for the task is created in the repository.
templateCreate create the prefilled Rmd template for the task, default value: FALSE.
overwrite logical, if TRUE and the task already exists, overwrite its parameters.

Value

D4TAlinkTask object
initTaskRmd

Create task template in Rmd format.

Description

Create task template in Rmd format.

Usage

initTaskRmd(task, encoding = "unknown", overwrite = FALSE)

Arguments

task Object of class D4TAlinkTask, as created by initTask.
encoding encoding to be assumed for input strings. It is used to mark character strings as known to be in Latin-1 or UTF-8; it is not used to re-encode the input. To do the latter, specify the encoding as part of the connection con or via options(encoding=): see the examples. See also ‘Details’.
overwrite overwrite Rmd file if exists, default FALSE

Value

the file name invisibly.

initTaskRscript

Create task R script.

Description

Create task R script.

Usage

initTaskRscript(task, overwrite = FALSE, encoding = "unknown")

Arguments

task Object of class D4TAlinkTask, as created by initTask.
overwrite overwrite R file if exists, default FALSE
encoding encoding to be assumed for input strings. It is used to mark character strings as known to be in Latin-1 or UTF-8: it is not used to re-encode the input. To do the latter, specify the encoding as part of the connection con or via options(encoding=): see the examples. See also ‘Details’.

Value

the file name invisibly.
**jpegReport**  
*Graphics devices for JPEG format bitmap files.*

**Description**  
Graphics devices for JPEG format bitmap files.

**Usage**

```r
graphReport(
    task,
    type,
    ext = "jpg",
    subdir = NULL,
    dirCreate = TRUE,
    dim = c(500, 500),
    width = NULL,
    height = NULL,
    ...
)
```

**Arguments**

- **task**  
  Object of class `D4TAlinkTask`, as created by `initTask`.

- **type**  
  Should be plotting be done using Windows GDI or cairographics?

- **ext**  
  Filename extension.

- **subdir**  
  (optional) Subdirectory.

- **dirCreate**  
  Logical, if TRUE (by default) the directory is created.

- **dim**  
  device height and width in px.

- **width**  
  device height in px.

- **height**  
  device height in px.

- **...**  
  Arguments passed on to `grDevices::jpeg`

  - **filename**  
    the path of the output file, up to 511 characters. The page number is substituted if a C integer format is included in the character string, as in the default, and tilde-expansion is performed (see `path.expand`). (The result must be less than 600 characters long. See `postscript` for further details.)

  - **units**  
    The units in which height and width are given. Can be px (pixels, the default), in (inches), cm or mm.

  - **pointsize**  
    the default pointsize of plotted text, interpreted as big points (1/72 inch) at res ppi.

  - **bg**  
    the initial background colour: can be overridden by setting par("bg").

  - **quality**  
    the ‘quality’ of the JPEG image, as a percentage. Smaller values will give more compression but also more degradation of the image.
res The nominal resolution in ppi which will be recorded in the bitmap file, if a positive integer. Also used for units other than the default. If not specified, taken as 72 ppi to set the size of text and line widths.

family A length-one character vector specifying the default font family. The default means to use the font numbers on the Windows GDI versions and "sans" on the cairographics versions.

restoreConsole See the 'Details' section of windows. For type == "windows" only.

antialias Length-one character vector. For allowed values and their effect on fonts with type = "windows" see windows: for that type if the argument is missing the default is taken from windows.options($)bitmap.aa.win. For allowed values and their effect (on fonts and lines, but not fills) with type = "cairo" see svg.

symbolfamily For cairographics only: a length-one character string that specifies the font family to be used as the "symbol" font (e.g., for plotmath output). The default value is "default", which means that R will choose a default "symbol" font based on the graphics device capabilities.

Value

the file name invisibly.

jpegReportFn Get path of jpeg output file.

Description

Get path of jpeg output file.

Usage

jpegReportFn(task, type, ext = "jpg", subdir = NULL)

Arguments

task Object of class D4TAlinkTask, as created by initTask.
type Filename type. If the type is an array, the concatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
ext Filename extension.
subdir (optional) Subdirectory.

Value

File path.
listTaskFiles

List the files associated to a task.

Description

List the files associated to a task.

Usage

listTaskFiles(task, full.names = FALSE, which = NULL)

Arguments

task Object of class D4TAlinkTask, as created by initTask.
full.names a logical value. If TRUE, the directory path is prepended to the file names to give a relative file path. If FALSE, the file names (rather than paths) are returned.
which list of file types to list.

Value

array of file names.

listTasks

List details of all tasks stored in the root directory.

Description

List details of all tasks stored in the root directory.

Usage

listTasks(
    project = NULL,
    package = NULL,
    sponsor = NULL,
    rootpath = getTaskRoot()
)

Arguments

project Project name.
package Package name.
sponsor Sponsor name, default set by setTaskSponsor.
rootpath Path of the task repository, default set by setTaskRoot.
loadTask

Value

data.frame with the following information for tasks "sponsor", "project", "package", "task".

loadTask  Load a task.

Description

Load a task.

Usage

loadTask(
  project,
  package,
  taskname,
  sponsor = getTaskSponsor(),
  author = getTaskAuthor(),
  quiet = FALSE
)

Arguments

project    Project name.
package    Package name.
taskname   Task name.
sponsor    Sponsor name, default set by setTaskSponsor.
author     Author name, system username by default.
quiet      issue warning if file does not exists.

Value

Object of class D4TaLinkTask or NULL if the task does not exists.
pathsDefault

Task paths generator.

Description

The paths are:
- datasrc: [ROOT]/[sponsor]/[project]/[package]/raw/datasource
- data: [ROOT]/[sponsor]/[project]/[package]/output/adhoc/[taskname]
- bin: [ROOT]/[sponsor]/[project]/[package]/output/adhoc/[taskname]/bin
- code: [ROOT]/[sponsor]/[project]/[package]/progs
- doc: [ROOT]/[sponsor]/[project]/[package]/docs
- log: [ROOT]/[sponsor]/[project]/[package]/output/log

Usage

pathsDefault(project, package, taskname, sponsor)

Arguments

- **project**: Project name.
- **package**: Package name.
- **taskname**: Task name.
- **sponsor**: Sponsor name, default set by `setTaskSponsor`.

Value

a list of file paths

pathsGLPG

Task paths generator.

Description

The paths are:
- datasrc: [ROOT]/[sponsor]/[project]/[package]/raw/datasource
- data: [ROOT]/[sponsor]/[project]/[package]/output/adhoc/[taskname]
- bin: [ROOT]/[sponsor]/[project]/[package]/output/adhoc/[taskname]/bin
- code: [ROOT]/[sponsor]/[project]/[package]/progs
- doc: [ROOT]/[sponsor]/[project]/[package]/docs
- log: [ROOT]/[sponsor]/[project]/[package]/output/log

Usage

pathsGLPG(project, package, taskname, sponsor)

Arguments

- **project**: Project name.
- **package**: Package name.
- **taskname**: Task name.
- **sponsor**: Sponsor name, default set by `setTaskSponsor`.

Value

a list of file paths
pathsPMS  

Task paths generator.

**Description**

The paths are:  
- datsrc: [ROOT]/[sponsor]/PMS_data/[project]/[package]/datasource
- data: [ROOT]/[sponsor]/PMS_data/[project]/[package]/[taskname]
- bin: [ROOT]/[sponsor]/PMS_data/[project]/[package]/[taskname]/bin
- code: [ROOT]/[sponsor]/PMS_code/[project]/[package]/[taskname]
- doc: [ROOT]/[sponsor]/PMS_documentation/[project]/[package]/[taskname]
- log: [ROOT]/[sponsor]/PMS_data/[project]/[package]/[taskname]

**Usage**

pathsPMS(project, package, taskname, sponsor)

**Arguments**

- **project**  
  Project name.
- **package**  
  Package name.
- **taskname**  
  Task name.
- **sponsor**  
  Sponsor name, default set by setTaskSponsor.

**Value**

a list of file paths

---

pdfReport  

Graphics devices for pdf format bitmap files.

**Description**

Graphics devices for pdf format bitmap files.

**Usage**

pdfReport(
  task,
  type,
  ext = "pdf",
  subdir = NULL,
  dirCreate = TRUE,
  title = NA,
  file = NA,
  dim = c(297, 210),
  height = NULL,
  width = NULL,
  landscape = NULL,
  ...  
)
Arguments

- **task**: Object of class `D4AlinkTask`, as created by `initTask`.
- **type**: Filename type. If the type is an array, the concatenation of the elements is used with separator"-". Filenames have the form `[task name]_[type].[ext]`
- **ext**: Filename extension.
- **subdir** (optional): Subdirectory.
- **dirCreate**: Logical, if TRUE (by default) the directory is created.
- **title**: Title string to embed as the '/Title' field in the file. Defaults to "R Graphics Output".
- **file**: a character string giving the file path. If it is of the form "|cmd", the output is piped to the command given by cmd. If it is NULL, then no external file is created (effectively, no drawing occurs), but the device may still be queried (e.g., for size of text).
  For use with `onefile = FALSE` give a C integer format such as "Rplot%03d.pdf" (the default in that case). (See `postscript` for further details.)
  Tilde expansion (see `path.expand`) is done. An input with a marked encoding is converted to the native encoding or an error is given.
- **dim**: device height and width in mm.
- **height**: device height in mm.
- **width**: device height in mm.
- **landscape**: If defined, orientation of the document.
- **...**: Arguments passed on to `grDevices::pdf`
  `width, height` the width and height of the graphics region in inches. The default values are 7.
- **onefile**: logical; if true (the default) allow multiple figures in one file. If false, generate a file with name containing the page number for each page. Defaults to TRUE, and forced to true if `file` is a pipe.
- **family**: the font family to be used, see `postscript`. Defaults to "Helvetica".
- **fonts**: a character vector specifying R graphics font family names for additional fonts which will be included in the PDF file. Defaults to NULL.
- **version**: a string describing the PDF version that will be required to view the output. This is a minimum, and will be increased (with a warning) if necessary. Defaults to "1.4", but see ‘Details’.
- **paper**: the target paper size. The choices are "a4", "letter", "legal" (or "us") and "executive" (and these can be capitalized), or "a4r" and "usr" for rotated (‘landscape’). The default is "special", which means that the width and height specify the paper size. A further choice is "default"; if this is selected, the paper size is taken from the option "papersize" if that is set and as "a4" if it is unset or empty. Defaults to "special".
- **encoding**: the name of an encoding file. See `postscript` for details. Defaults to "default".
- **bg**: the initial background color to be used. Defaults to "transparent".
- **fg**: the initial foreground color to be used. Defaults to "black".
The default point size to be used. Strictly speaking, in bp, that is 1/72 of an inch, but approximately in points. Defaults to 12.

`pagecentre` logical: should the device region be centred on the page? – is only relevant for `paper != "special"`. Defaults to TRUE.

`colormodel` a character string describing the color model: currently allowed values are "srgb", "gray" (or "grey") and "cmyk". Defaults to "srgb". See section 'Color models'.

`useDingbats` logical. Should small circles be rendered via the Dingbats font? Defaults to FALSE. If TRUE, this can produce smaller and better output, but there can font display problems in broken PDF viewers: although this font is one of the 14 guaranteed to be available in all PDF viewers, that guarantee is not always honoured. For Unix-alikes (including macOS) see the ‘Note’ for a possible fix for some viewers.

`useKerning` logical. Should kerning corrections be included in setting text and calculating string widths? Defaults to TRUE.

`fillOddEven` logical controlling the polygon fill mode: see `polygon` for details. Defaults to FALSE.

`compress` logical. Should PDF streams be generated with Flate compression? Defaults to TRUE.

**Value**

the file name invisibly.

---

**pdfReportFn**

Get path of pdf output file.

**Description**

Get path of pdf output file.

**Usage**

```r
pdfReportFn(task, type, ext = "pdf", subdir = NULL)
```

**Arguments**

- **task** Object of class `D4TAlinkTask`, as created by `initTask`.
- **type** Filename type. If the type is an array, the concatenation of the elements is used with separator".". Filenames have the form `[task name]_[type].[ext]`
- **ext** Filename extension.
- **subdir** (optional) Subdirectory.

**Value**

File path.
**Description**

Graphics devices for PNG format bitmap files.

**Usage**

```r
graphicsReport(
    task,
    type,
    ext = "png",
    subdir = NULL,
    dirCreate = TRUE,
    dim = c(500, 500),
    width = NULL,
    height = NULL,
    ...
)
```

**Arguments**

- **task**: Object of class `D4TAlinkTask`, as created by `initTask`.
- **type**: Should be plotting done using Windows GDI or cairographics?
- **ext**: Filename extension.
- **subdir**: (optional) Subdirectory.
- **dirCreate**: Logical, if TRUE (by default) the directory is created.
- **dim**: device height and width in px.
- **width**: device height in px.
- **height**: device height in px.
- **...**: Arguments passed on to `grDevices::png`
  - **filename**: the path of the output file, up to 511 characters. The page number is substituted if a C integer format is included in the character string, as in the default, and tilde-expansion is performed (see `path.expand`). (The result must be less than 600 characters long. See `postscript` for further details.)
  - **units**: The units in which height and width are given. Can be px (pixels, the default), in (inches), cm or mm.
  - **pointsize**: the default pointsize of plotted text, interpreted as big points (1/72 inch) at res ppi.
  - **bg**: the initial background colour: can be overridden by setting `par("bg")`.
  - **res**: The nominal resolution in ppi which will be recorded in the bitmap file, if a positive integer. Also used for units other than the default. If not specified, taken as 72 ppi to set the size of text and line widths.
family  A length-one character vector specifying the default font family. The
default means to use the font numbers on the Windows GDI versions and
"sans" on the cairographics versions.
restoreConsole  See the ‘Details’ section of windows. For type == "windows"
only.
antialias  Length-one character vector.
   For allowed values and their effect on fonts with type = "windows" see
windows: for that type if the argument is missing the default is taken from
windows.options()$bitmap.aa.win.
   For allowed values and their effect (on fonts and lines, but not fills) with
type = "cairo" see svg.
symbolfamily  For cairographics only: a length-one character string that spec-
ifies the font family to be used as the "symbol" font (e.g., for plotmath
output). The default value is "default", which means that R will choose a
default "symbol" font based on the graphics device capabilities.

Value
   the file name invisibly.

Description
   Get path of png output file.

Usage
   pngReportFn(task, type, ext = "png", subdir = NULL)

Arguments
   task  Object of class D4TAlinkTask, as created by initTask.
   type  Filename type. If the type is an array, the concatenation of the elements is used
   with separator". " . Filenames have the form [task name]_[type][ext]
   ext  Filename extension.
   subdir  (optional) Subdirectory.

Value
   File path.
**progDir**

*Get path of scripts directory.*

**Description**

Get path of scripts directory.

**Usage**

```
progDir(task, subdir = NULL, dirCreate = TRUE)
```

**Arguments**

- **task**: Object of class `D4TAlinkTask`, as created by `initTask`.
- **subdir**: (optional) Subdirectory.
- **dirCreate**: Logical, if TRUE (by default) the directory is created.

**Value**

File path.

---

**readBinary**

*Restore R object from binary file.*

**Description**

Restore R object from binary file.

**Usage**

```
readBinary(
    task,
    type,
    subdir = NULL,
    dirCreate = FALSE,
    ask = FALSE,
    quiet = FALSE
)
```
**Arguments**

- **task**: Object of class `D4TAlinkTask`, as created by `initTask`.
- **type**: Filename type. If the type is an array, the concatenation of the elements is used with separator ".". Filenames have the form `[task name]_[type].[ext]`.
- **subdir**: (optional) Subdirectory.
- **dirCreate**: Logical, if TRUE (by default) the directory is created.
- **ask**: query encryption key from user, default FALSE.
- **quiet**: issue warning if file does not exist.

**Value**

Object stored in binary file, or NULL if file does not exist.

---

**Description**

Read JSON data into R object.

**Usage**

```r
readReportJSON(task, type, ext = "json", subdir = NULL, dirCreate = FALSE)
```

**Arguments**

- **task**: Object of class `D4TAlinkTask`, as created by `initTask`.
- **type**: Filename type. If the type is an array, the concatenation of the elements is used with separator ".". Filenames have the form `[task name]_[type].[ext]`.
- **ext**: Filename extension.
- **subdir**: (optional) Subdirectory.
- **dirCreate**: Logical, if TRUE (by default) the directory is created.

**Value**

the data read, or NULL if the file does not exist.
readReportTable  

Read data into vector or list using function \texttt{scan}.

Description

Read data into vector or list using function \texttt{scan}.

Usage

\begin{verbatim}
readReportTable(task, type, ext = "csv", subdir = NULL, dirCreate = FALSE, ...)
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{task} Object of class \texttt{D4TAlinkTask}, as created by \texttt{initTask}.
  \item \texttt{type} Filename type. If the type is an array, the concatenation of the elements is used with separator ".". Filenames have the form \texttt{[task name]_[type].[ext]}
  \item \texttt{ext} Filename extension.
  \item \texttt{subdir} (optional) Subdirectory.
  \item \texttt{dirCreate} Logical, if \texttt{TRUE} (by default) the directory is created.
  \item ... Arguments passed on to \texttt{utils::read.csv}
  \item \texttt{file} the name of the file which the data are to be read from. Each row of the table appears as one line of the file. If it does not contain an absolute path, the file name is relative to the current working directory, \texttt{getwd()}. Tilde-expansion is performed where supported. This can be a compressed file (see \texttt{file}). Alternatively, file can be a readable text-mode \texttt{connection} (which will be opened for reading if necessary, and if so \texttt{closed} (and hence destroyed) at the end of the function call). (If \texttt{stdin()} is used, the prompts for lines may be somewhat confusing. Terminate input with a blank line or an EOF signal, \texttt{Ctrl-D} on Unix and \texttt{Ctrl-Z} on Windows. Any pushback on \texttt{stdin()} will be cleared before return.) file can also be a complete URL. (For the supported URL schemes, see the ‘URLs’ section of the help for \texttt{url}.)
  \item \texttt{header} a logical value indicating whether the file contains the names of the variables as its first line. If missing, the value is determined from the file format: \texttt{header} is set to \texttt{TRUE} if and only if the first row contains one fewer field than the number of columns.
  \item \texttt{sep} the field separator character. Values on each line of the file are separated by this character. If \texttt{sep = ""} (the default for \texttt{read.table}) the separator is ‘white space’, that is one or more spaces, tabs, newlines or carriage returns.
  \item \texttt{quote} the set of quoting characters. To disable quoting altogether, use \texttt{quote = ""}. See \texttt{scan} for the behaviour on quotes embedded in quotes. Quoting is only considered for columns read as character, which is all of them unless \texttt{colClasses} is specified.
  \item \texttt{dec} the character used in the file for decimal points.
\end{itemize}
fill logical. If TRUE then in case the rows have unequal length, blank fields are implicitly added. See ‘Details’.

comment.char character: a character vector of length one containing a single character or an empty string. Use "" to turn off the interpretation of comments altogether.

Value

the data read, or NULL if the file does not exist.

renderTaskRmd Render the task from the Rmd file

Description

The template of the task is rendered towards pdf or html in the documentation directory of the specified task.

Usage

renderTaskRmd(task, output_format = NULL, debug = FALSE, clean = TRUE, ...)

Arguments

task Object of class D4TAlinkTask, as created by initTask.

output_format The R Markdown output format to convert to. The option "all" will render all formats defined within the file. The option can be the name of a format (e.g. "html_document") and that will render the document to that single format. One can also use a vector of format names to render to multiple formats. Alternatively, you can pass an output format object (e.g. html_document()). If using NULL then the output format is the first one defined in the YAML frontmatter in the input file (this defaults to HTML if no format is specified there). If you pass an output format object to output_format, the options specified in the YAML header or _output.yml will be ignored and you must explicitly set all the options you want when you construct the object. If you pass a string, the output format will use the output parameters in the YAML header or _output.yml.

d debug if TRUE execute in the global environment.

clean Using TRUE will clean intermediate files that are created during rendering.

Arguments passed on to rmarkdown::render

input The input file to be rendered. This can be an R script (.R), an R Markdown document (.Rmd), or a plain markdown document.

output_file The name of the output file. If using NULL then the output filename will be based on filename for the input file. If a filename is provided, a path to the output file can also be provided. Note that the output_dir option allows for specifying the output file path as well, however, if also
specifying the path, the directory must exist. If output_file is specified but does not have a file extension, an extension will be automatically added according to the output format. To avoid the automatic file extension, put the output_file value in \texttt{I()}, e.g., \texttt{I(‘my-output’)}.

\texttt{output_dir}  The output directory for the rendered output_file. This allows for a choice of an alternate directory to which the output file should be written (the default output directory of that of the input file). If a path is provided with a filename in output_file the directory specified here will take precedence. Please note that any directory path provided will create any necessary directories if they do not exist.

\texttt{output_options}  List of output options that can override the options specified in metadata (e.g. could be used to force \texttt{self_contained} or \texttt{mathjax = "local"}). Note that this is only valid when the output format is read from metadata (i.e. not a custom format object passed to output_format).

\texttt{output_yaml}  Paths to YAML files specifying output formats and their configurations. The first existing one is used. If none are found, then the function searches YAML files specified to the output_yaml top-level parameter in the YAML front matter, \texttt{_output.yml} or \texttt{_output.yaml}, and then uses the first existing one.

\texttt{intermediates_dir}  Intermediate files directory. If a path is specified then intermediate files will be written to that path. If \texttt{NULL}, intermediate files are written to the same directory as the input file.

\texttt{knit_root_dir}  The working directory in which to knit the document; uses \texttt{knitr’s root.dir} knit option. If \texttt{NULL} then the behavior will follow the \texttt{knitr} default, which is to use the parent directory of the document.

\texttt{runtime}  The runtime target for rendering. The static option produces output intended for static files; shiny produces output suitable for use in a Shiny document (see \texttt{run}). The default, \texttt{auto}, allows the runtime target specified in the YAML metadata to take precedence, and renders for a static runtime target otherwise.

\texttt{params}  A list of named parameters that override custom params specified within the YAML front-matter (e.g. specifying a dataset to read or a date range to confine output to). Pass “ask” to start an application that helps guide parameter configuration.

\texttt{knit_meta}  (This option is reserved for expert use.) Metadata generated by \texttt{knitr}.

\texttt{envir}  The environment in which the code chunks are to be evaluated during knitting (can use \texttt{new.env()} to guarantee an empty new environment).

\texttt{run_pandoc}  An option for whether to run pandoc to convert Markdown output.

\texttt{quiet}  An option to suppress printing during rendering from knitr, pandoc command line and others. To only suppress printing of the last “Output created: ” message, you can set \texttt{rmarkdown.render.message} to \texttt{FALSE}.

\texttt{encoding}  Ignored. The encoding is always assumed to be UTF-8.

\textbf{Value}

the file name invisibly.
**reportDir**

Get path of report directory.

**Description**

Get path of report directory.

**Usage**

`reportDir(task, subdir = NULL, dirCreate = TRUE)`

**Arguments**

- **task**: Object of class `D4TAlinkTask`, as created by `initTask`.
- **subdir**: (optional) Subdirectory.
- **dirCreate**: Logical, if TRUE (by default) the directory is created.

**Value**

File path.

---

**reportFn**

Get path of output file.

**Description**

Get path of output file.

**Usage**

`reportFn(task, type, ext, subdir = NULL, dirCreate = TRUE)`

**Arguments**

- **task**: Object of class `D4TAlinkTask`, as created by `initTask`.
- **type**: Filename type. If the type is an array, the concatenation of the elements is used with separator".". Filenames have the form `[task name]_[type].[ext]`.
- **ext**: Filename extension.
- **subdir**: (optional) Subdirectory.
- **dirCreate**: Logical, if TRUE (by default) the directory is created.

**Value**

File path.
**reportXlsFn**

*Get path of xlsx output file.*

**Description**

Get path of xlsx output file.

**Usage**

```
reportXlsFn(task, type, ext = "xlsx", subdir = NULL)
```

**Arguments**

- `task`: Object of class `D4TAlinkTask`, as created by `initTask`.
- `type`: Filename type. If the type is an array, the concatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
- `ext`: Filename extension.
- `subdir`: (optional) Subdirectory.

**Value**

File path.

---

**restoreTask**

*Restore an archive containing the files of a given task from a file created with archiveTask.*

**Description**

Restore an archive containing the files of a given task from a file created with archiveTask.

**Usage**

```
restoreTask(file, overwrite = FALSE, list = FALSE, ...)
```

**Arguments**

- `file`: full name of the input zip file
- `overwrite`: If TRUE, overwrite existing files (the equivalent of `unzip -o`), otherwise ignore such files (the equivalent of `unzip -n`).
- `list`: If TRUE, list the files and extract none. The equivalent of `unzip -l`.
- `...`: Arguments passed on to `utils::unzip`

zipfile The pathname of the zip file: tilde expansion (see `path.expand`) will be performed.
files A character vector of recorded filepaths to be extracted: the default is to extract all files.

junkpaths If TRUE, use only the basename of the stored filepath when extracting. The equivalent of unzip -j.

exdir The directory to extract files to (the equivalent of unzip -d). It will be created if necessary.

unzip The method to be used. An alternative is to use getOption("unzip"), which on a Unix-alike may be set to the path to a unzip program.

setTimes logical. For the internal method only, should the file times be set based on the times in the zip file? (NB: this applies to included files, not to directories.)

Value

if list FALSE, the task imported incisibly, otherwise the list of files in the archive.

---

**rmdFn**

*Get path of R script file name.*

Description

Get path of R script file name.

Usage

rmdFn(task)

Arguments

task Object of class D4TAlinkTask, as created by initTask.

Value

File path.
**rscriptFn**

*Get path of R script file name.*

**Description**

Get path of R script file name.

**Usage**

```
rscriptFn(task)
```

**Arguments**

- **task**: Object of class `D4TAlinkTask`, as created by `initTask`.

**Value**

File path.

---

**saveBinary**

*Save R object in binary file.*

**Description**

Save R object in binary file.

**Usage**

```
saveBinary(
    object,  
    task,    
    type,    
    subdir = NULL,  
    dirCreate = TRUE,  
    encrypt = FALSE,  
    ask = FALSE  
)
```

**Arguments**

- **object**: R object to serialize.
- **task**: Object of class `D4TAlinkTask`, as created by `initTask`.
- **type**: Filename type. If the type is an array, the concatenation of the elements is used with separator `"."`. Filenames have the form [task name]_[type].[ext]
- **subdir**: (optional) Subdirectory.
saveReportJSON

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dirCreate</td>
<td>Logical, if TRUE (by default) the directory is created.</td>
</tr>
<tr>
<td>encrypt</td>
<td>encrypt the output, default: FALSE.</td>
</tr>
<tr>
<td>ask</td>
<td>query encryption key from user, default FALSE.</td>
</tr>
</tbody>
</table>

**Value**

the file name invisibly.

---

**Description**

Output R object in JSON format.

**Usage**

```r
saveReportJSON(
  x, task, type, ext = "json", subdir = NULL, dirCreate = TRUE,
  ...
)
```

**Arguments**

- **x** R object to output.
- **task** Object of class `D4TAlinkTask`, as created by `initTask`.
- **type** Filename type. If the type is an array, the concatenation of the elements is used with separator ".". Filenames have the form [task name]_[type].[ext]
- **ext** Filename extension.
- **subdir** (optional) Subdirectory.
- **dirCreate** Logical, if TRUE (by default) the directory is created.
- **...** Arguments passed on to `base::cat`

**Value**

the file name invisibly.
Output R object using function `write.csv`.

### Usage

```r
saveReportTable(
  x, 
  task, 
  type, 
  ext = "csv", 
  subdir = NULL, 
  dirCreate = TRUE, 
  gzip = FALSE, 
  ...
)
```

### Arguments

- `x` R object to output.
- `task` Object of class `D4TAlinkTask`, as created by `initTask`.
- `type` Filename type. If the type is an array, the concatenation of the elements is used with separator `""`. Filenames have the form `[task name]_[type].[ext]`
- `ext` Filename extension.
- `subdir` (optional) Subdirectory.
- `dirCreate` Logical, if TRUE (by default) the directory is created.
- `gzip` unused.
- `...` Arguments passed on to `utils::write.csv`

### Value

the file name invisibly.
saveReportXls  

**Description**

Save R object in binary file.

**Usage**

```r
saveReportXls(  
  x,  
  task,  
  type,  
  ext = "xlsx",  
  subdir = NULL,  
  dirCreate = TRUE,  
  AdjWidth = TRUE,  
  FreezeRow = 1,  
  FreezeCol = 3,  
  metadata = "metadata",  
  metadata.append = NULL,  
  ...  
)
```

**Arguments**

- `x` object to save. It can be either a data frame, an object of type `AnnotatedDataFrame`, or a list thereof.
- `task` Object of class `D4TAlinkTask`, as created by `initTask`.
- `type` Filename type. If the type is an array, the concatenation of the elements is used with separator ".". Filenames have the form `[task name]_[type].[ext]`
- `ext` Filename extension.
- `subdir` (optional) Subdirectory.
- `dirCreate` Logical, if TRUE (by default) the directory is created.
- `AdjWidth` If TRUE, will adjust the worksheet column widths based upon the longest entry in each column. This is approximate.
- `FreezeRow` Rows including this row and above this row will be frozen and not scroll. The default value of 0 will scroll the entire sheet. Note that not all spreadsheet applications support this feature.
- `FreezeCol` Columns including this column and to the left of this column will be frozen and not scroll. The default value of 0 will scroll the entire sheet. Note that not all spreadsheet applications support this feature.
- `metadata` prefix for names of worksheets holding metadata.
metadata.append

... Arguments passed on to WriteXLS::WriteXLS

ExcelFileName The name of the Excel file to be created. If the file extension is .XLS, an Excel 2003 file will be created. If the file extension is .XLSX, an Excel 2007 file will be created. Must be a valid Excel filename. May include an existing path. normalizePath is used to support tilde expansion, etc.

SheetNames A character vector containing the names of each worksheet to be created. If NULL (the default), the names of the dataframes will be used instead. Worksheet names may be up to 31 characters in length and must be unique. If specified, length(SheetNames) must be the same as length(x). NOTE: The order of the names here must match the order of the data frames as listed in x.

perl Name of the perl executable to be called.

verbose Output step-by-step status messages during the creation of the Excel file. Default is FALSE.

Encoding Define the character encoding to be used for the exported data frames. Defaults to UTF-8.

AllText If TRUE, all cell contents of the Excel file will be written as text. Default is FALSE. See Details.

row.names If TRUE, the row names of the data frames are included in the Excel file worksheets.

col.names If TRUE, the column names of the data frames are included in the Excel file worksheets.

AutoFilter If TRUE, will add autofiltering to each column in each worksheet. Note that not all spreadsheet applications support this feature.

BoldHeaderRow If TRUE, will apply a bold font to the header row for each worksheet.

na The string to use for missing values in the data. Defaults to "".

envir The environment in which to look for the data frames named in x. This defaults to the environment in which WriteXLS was called.

Value

the file name invisibly.

---

**scanReport**

*Read data into vector or list using function `scan`.*

---

**Description**

Read data into vector or list using function `scan`. 
Usage

scanReport(
  task,
  type,
  ext = "txt",
  subdir = NULL,
  dirCreate = TRUE,
  what = "",
  ...
)

Arguments

task
  Object of class D4TAlinkTask, as created by initTask.

type
  Filename type. If the type is an array, the concatenation of the elements is used
  with separator ".". Filenames have the form [task name]_[type].[ext]

ext
  Filename extension.

subdir
  (optional) Subdirectory.

dirCreate
  Logical, if TRUE (by default) the directory is created.

what
  the type of what gives the type of data to be read. (Here 'type' is used in
  the sense of typeof.) The supported types are logical, integer, numeric,
  complex, character, raw and list. If what is a list, it is assumed that the lines
  of the data file are records each containing length(what) items ('fields') and
  the list components should have elements which are one of the first six (atomic)
  types listed or NULL, see section ‘Details’ below.

file
  the name of a file to read data values from. If the specified file is "",
  then input is taken from the keyboard (or whatever stdin() reads if input
  is redirected or R is embedded). (In this case input can be terminated by a
  blank line or an EOF signal, ‘Ctrl-D’ on Unix and ‘Ctrl-Z’ on Windows.)
  Otherwise, the file name is interpreted relative to the current working di-
  rectory (given by getwd()), unless it specifies an absolute path. Tilde-
  expansion is performed where supported. When running R from a script,
  file = "stdin" can be used to refer to the process’s stdin file stream.
  This can be a compressed file (see file).
  Alternatively, file can be a connection, which will be opened if neces-
  sary, and if so closed at the end of the function call. Whatever mode the
  connection is opened in, any of LF, CRLF or CR will be accepted as the
  EOL marker for a line and so will match sep = "\n".
  File can also be a complete URL. (For the supported URL schemes, see
  the ‘URLs’ section of the help for url.)
  To read a data file not in the current encoding (for example a Latin-1 file in
  a UTF-8 locale or conversely) use a file connection setting its encoding
  argument (or scan’s fileEncoding argument).

nmax
  the maximum number of data values to be read, or if what is a list, the
  maximum number of records to be read. If omitted or not positive or an
invalid value for an integer (and nlines is not set to a positive value), scan will read to the end of file.

n integer: the maximum number of data values to be read, defaulting to no limit. Invalid values will be ignored.

sep by default, scan expects to read ‘white-space’ delimited input fields. Alternatively, sep can be used to specify a character which delimits fields. A field is always delimited by an end-of-line marker unless it is quoted. If specified this should be the empty character string (the default) or NULL or a character string containing just one single-byte character.

quote the set of quoting characters as a single character string or NULL. In a multibyte locale the quoting characters must be ASCII (single-byte).

dec decimal point character. This should be a character string containing just one single-byte character. (NULL and a zero-length character vector are also accepted, and taken as the default.)

skip the number of lines of the input file to skip before beginning to read data values.

nlines if positive, the maximum number of lines of data to be read.

na.strings character vector. Elements of this vector are to be interpreted as missing (NA) values. Blank fields are also considered to be missing values in logical, integer, numeric and complex fields. Note that the test happens after white space is stripped from the input, so na.strings values may need their own white space stripped in advance.

flush logical: if TRUE, scan will flush to the end of the line after reading the last of the fields requested. This allows putting comments after the last field, but precludes putting more that one record on a line.

fill logical: if TRUE, scan will implicitly add empty fields to any lines with fewer fields than implied by what.

strip.white vector of logical value(s) corresponding to items in the what argument. It is used only when sep has been specified, and allows the stripping of leading and trailing ‘white space’ from character fields (numeric fields are always stripped). Note: white space inside quoted strings is not stripped.

If strip.white is of length 1, it applies to all fields; otherwise, if strip.white[i] is TRUE and the i-th field is of mode character (because what[i] is) then the leading and trailing unquoted white space from field i is stripped.

quiet logical: if FALSE (default), scan() will print a line, saying how many items have been read.

blank.lines.skip logical: if TRUE blank lines in the input are ignored, except when counting skip and nlines.

multi.line logical. Only used if what is a list. If FALSE, all of a record must appear on one line (but more than one record can appear on a single line). Note that using fill = TRUE implies that a record will be terminated at the end of a line.

comment.char character: a character vector of length one containing a single character or an empty string. Use "" to turn off the interpretation of comments altogether (the default).
allowEscapes logical. Should C-style escapes such as \n be processed (the
default) or read verbatim? Note that if not within quotes these could be
interpreted as a delimiter (but not as a comment character).
The escapes which are interpreted are the control characters \a, \b, \f,
\n, \r, \t, \v and octal and hexadecimal representations like \040 and
\0x2A. Any other escaped character is treated as itself, including back-
slash. Note that Unicode escapes (starting \u or \U: see Quotes) are
never processed.

fileEncoding character string: if non-empty declares the encoding used on a
file (not a connection nor the keyboard) so the character data can be re-
encoded. See the ‘Encoding’ section of the help for file, and the ‘R Data
Import/Export Manual’.
encoding encoding to be assumed for input strings. If the value is "latin1"
or "UTF-8" it is used to mark character strings as known to be in Latin-1 or
UTF-8: it is not used to re-encode the input (see fileEncoding). See also
‘Details’.
text character string: if file is not supplied and this is, then data are read from
the value of text via a text connection.
skipNu1 logical: should nuls be skipped when reading character fields?

Value
the data read, or NULL if the file does not exist.

Description
Set the name of the tasks author.

Usage
setTaskAuthor(author)

Arguments
author Author name, system username by default.

Value
The current name of the tasks author.

Examples
setTaskAuthor("Doe Johns")
setTaskEnckey

*Set the encryption key for binary data files.*

**Description**
Set the encryption key for binary data files.

**Usage**

```r
setTaskEnckey(key)
```

**Arguments**

- **key**
  - encryption key, if NULL then query key from user.

**Value**

NULL invisibly.

---

setTaskRmdTemplate

*Set the path to the Rmd task template.*

**Description**

Set the path to the Rmd task template.

**Usage**

```r
csetTaskRmdTemplate(file, encoding = "unknown")
```

**Arguments**

- **file**
  - path to the Rmd task template.
- **encoding**
  - encoding to be assumed for input strings. It is used to mark character strings as known to be in Latin-1 or UTF-8: it is not used to re-encode the input. To do the latter, specify the encoding as part of the connection `con` or via `options(encoding=)`; see the examples. See also ‘Details’.

**Value**

The path to the Rmd task template invisibly.
**setTaskRoot**

Set the root of the task repository.

**Description**

Set the root of the task repository.

**Usage**

```r
setTaskRoot(rootpath, dirCreate = FALSE)
```

**Arguments**

- `rootpath` : Path of the task repository, default set by `setTaskRoot`.
- `dirCreate` : Logical, if TRUE (by default) the directory is created.

**Value**

Path to the current task root.

---

**setTaskRscriptTemplate**

Set the path to the R script task template.

**Description**

Set the path to the R script task template.

**Usage**

```r
setTaskRscriptTemplate(file)
```

**Arguments**

- `file` : path to the Rmd task template.

**Value**

The path to the Rmd task template invisibly.
### setTaskSponsor

*Set the name of the tasks sponsor.*

**Description**

Set the name of the tasks sponsor.

**Usage**

`setTaskSponsor(sponsor)`

**Arguments**

- `sponsor`  
  Sponsor name, default set by `setTaskSponsor`.

**Value**

The current name of the tasks sponsor.

**Examples**

```python
setTaskSponsor("SQU4RE")
```

### setTaskStructure

*Set task repository directory structure.*

**Description**

Set task repository directory structure.

**Usage**

`setTaskStructure(pathgen)`

**Arguments**

- `pathgen`  
  optional function returning a list of paths, currently `pathsGLPG` or `pathsPMS`.

**Value**

The task directory structure function invisibly.
**Examples**

```r
fun <- function(project, package, taskname, sponsor) {
  basePath <- file.path("%ROOT%", sponsor, project, package)
  list(
    root = "%ROOT%",
    datasrc = file.path(basePath, "raw", "data_source"),
    data = file.path(basePath, "output", "adhoc", taskname),
    bin = file.path(basePath, "output", "adhoc", taskname, "bin"),
    code = file.path(basePath, "progs"),
    doc = file.path(basePath, "docs"),
    log = file.path(basePath, "output", "log")
  )
}
setTaskStructure(fun)
```

---

**taskID**

*Get task identifier string.*

---

**Description**

Get task identifier string.

**Usage**

```r
taskID(task, sep = "/")
```

**Arguments**

- **task**
  - Object of class `D4TAlinkTask`, as created by `initTask`.
- **sep**
  - the field separator character, default: "/".

**Value**

String with task ID as:

```
[sponsor][sep][project][sep][package][sep][task]
```
Index

archiveTask, 3
atomic, 41

base::cat, 5, 37
base::scan, 41
binaryDir, 4
binaryFn, 4

catReport, 5
close, 30
collection, 30, 41

D4getenv, 6
D4TAlink-common-args, 6
D4TAlinkTask, 3–6, 7, 8–13, 16–21, 24–31, 33–39, 41, 47
data.frame, 21
datasourceDir, 8
datasourceFn, 9
docDir, 9
docFn, 10
dTx, 10

file, 30, 41, 43
formatTaskDocx, 11

getTaskAuthor, 11
getTaskEnckey, 12
getTaskFilepath, 12
getTaskPaths, 13
getTaskRmdTemplate, 13
getTaskRoot, 14
getTaskRscriptTemplate, 14
getTaskSponsor, 15
getTaskStructure, 15
getwd, 30, 41
grDevices::jpeg, 18
grDevices::pdf, 24
grDevices::png, 26

I, 32

initTaskRmd, 17
initTaskRscript, 17
jpegReport, 18
jpegReportFn, 19
list, 41
listTaskFiles, 20
listTasks, 20
loadTask, 21

NA, 42
new.env, 32

options, 17, 44

path.expand, 3, 18, 24, 26, 34
pathsDefault, 22
pathsGLPG, 6, 22, 46
pathsPMS, 6, 23, 46
pdfReport, 23
pdfReportFn, 25
plotmath, 19, 27
pngReport, 26
pngReportFn, 27
polygon, 25
postscript, 18, 24, 26
progDir, 28

Quotes, 43

readBinary, 28
readReportJSON, 29
readReportTable, 30
renderTaskRmd, 31
reportDir, 33
reportFn, 33
reportXlsFn, 34
restoreTask, 34

48
rmarkdown::render, 31
rmdFn, 35
rscriptFn, 36
run, 32
saveBinary, 36
saveReportJSON, 37
saveReportTable, 38
saveReportXls, 39
scan, 30, 40
scanReport, 40
setTaskAuthor, 43
setTaskEnckey, 44
setTaskRmdTemplate, 44
setTaskRoot, 6, 20, 45, 45
setTaskRscriptTemplate, 45
setTaskSponsor, 6, 10, 16, 20–23, 46, 46
setTaskStructure, 46
stdin, 30, 41
svg, 19, 27
taskID, 7, 47
type, 41
typeof, 41
url, 30, 41
utils::read.csv, 30
utils::unzip, 34
utils::write.csv, 38
utils::zip, 3
windows, 19, 27
write.csv, 38
WriteXLS::WriteXLS, 40