

# Package: D4TAlink.light (via r-universe)

October 22, 2024

**Type** Package

**Title** FAIR Data - Workflow Management

**Version** 2.1.15

**Date** 2024-04-25

**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](https://doi.org/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** 7.3.1

**VignetteBuilder** knitr

**URL** <https://bitbucket.org/SQ4/d4talink.light>, <https://www.d4ta.link/>

**ByteCompile** true

**BugReports** <https://bitbucket.org/SQ4/d4talink.light/issues>

**Repository** <https://squ4reanalytics.r-universe.dev>

**RemoteUrl** <https://bitbucket.org/sq4/d4talink.light>

**RemoteRef** HEAD

**RemoteSha** 0a7220fb8dcbf34708ff3d060a8789b5e52b78bd

## Contents

archiveTask . . . . .	3
binaryDir . . . . .	4
binaryFn . . . . .	4
catReport . . . . .	5
D4getenv . . . . .	6
D4TAlink-common-args . . . . .	6
D4TAlinkTask . . . . .	7
datasourceDir . . . . .	8
datasourceFn . . . . .	9
docDir . . . . .	10
docFn . . . . .	10
DTx . . . . .	11
formatTaskDocx . . . . .	11
getTaskAuthor . . . . .	12
getTaskEnckey . . . . .	12
getTaskFilepath . . . . .	13
getTaskPaths . . . . .	13
getTaskRmdTemplate . . . . .	14
getTaskRoot . . . . .	14
getTaskRscriptTemplate . . . . .	15
getTaskSponsor . . . . .	15
getTaskStructure . . . . .	16
initTask . . . . .	16
initTaskRmd . . . . .	17
initTaskRscript . . . . .	18
jpegReport . . . . .	18
jpegReportFn . . . . .	20
listTaskFiles . . . . .	20
listTasks . . . . .	21
loadTask . . . . .	21
pathsDefault . . . . .	22
pathsGLPG . . . . .	23
pathsPMS . . . . .	23
pdfReport . . . . .	24
pdfReportFn . . . . .	26
pngReport . . . . .	26
pngReportFn . . . . .	28
progDir . . . . .	28
readBinary . . . . .	29
readReportJSON . . . . .	29
readReportTable . . . . .	30
renderTaskRmd . . . . .	31
reportDir . . . . .	33
reportFn . . . . .	34
reportXlsFn . . . . .	34
restoreTask . . . . .	35

rmdFn . . . . .	36
rscriptFn . . . . .	36
saveBinary . . . . .	37
saveBinaryE . . . . .	37
saveReportJSON . . . . .	38
saveReportTable . . . . .	39
saveReportXls . . . . .	40
scanReport . . . . .	41
setTaskAuthor . . . . .	44
setTaskEnckey . . . . .	45
setTaskRmdTemplate . . . . .	45
setTaskRoot . . . . .	46
setTaskRscriptTemplate . . . . .	46
setTaskSponsor . . . . .	47
setTaskStructure . . . . .	47
taskID . . . . .	48

## Index 49

---

archiveTask	<i>Create an archive containing the files of a given task.</i>
-------------	--

---

### Description

Create an archive containing the files of a given task.

### Usage

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

### Arguments

task	Object of class <a href="#">D4TALinkTask</a> , as created by <a href="#">initTask</a> .
file	full name of the output zip file
overwrite	overwrite the output zip file is it exists
...	Arguments passed on to <a href="#">utils::zip</a>
zipfile	The pathname of the zip file: tilde expansion (see <a href="#">path.expand</a> ) 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	<i>Get path of binary directory.</i>
-----------	--------------------------------------

---

**Description**

Get path of binary directory.

**Usage**

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

**Arguments**

task	Object of class <code>D4TAlinkTask</code> , as created by <code>initTask</code> .
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.

**Value**

File path.

---

binaryFn	<i>Get path of binary file.</i>
----------	---------------------------------

---

**Description**

Get path of binary file.

**Usage**

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

**Arguments**

task	Object of class <code>D4TAlinkTask</code> , as created by <code>initTask</code> .
type	Filename type. If the type is an array, the cocatenation 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.

---

catReport                      *Output R object using function cat.*

---

## Description

Output R object using function cat.

## Usage

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

## Arguments

x	R object to output.
task	Object of class <a href="#">D4TALinkTask</a> , as created by <a href="#">initTask</a> .
type	Filename type. If the type is an array, the cocatenation 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 <a href="#">base::cat</a>

## Value

the file name invisibly.

---

D4getenv	<i>Get env var.</i>
----------	---------------------

---

**Description**

Get env var.

**Usage**

```
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 <a href="#">setTaskSponsor</a> .
rootpath	Path of the task repository, default set by <a href="#">setTaskRoot</a> .
task	Object of class <a href="#">D4TALinkTask</a> , as created by <a href="#">initTask</a> .
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator"-". Filenames have the form [task name]_[type].[ext]
ext	Filename extension.
suffix	Filename suffix, used to develop scripts for sub-analyses for a given task, default NA.

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 <a href="#">pathsGLPG</a> or <a href="#">pathsPMS</a> .

### Value

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

---

D4TAlinkTask

D4TAlinkTask *Documentation of the D4TAlinkTask class*

---

### Description

The D4TAlinkTask 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 D4TAlinkTask object:

- [taskID](#): Get ID

### Value

Not relevant

**Examples**

```

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

# 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()$ask
par(ask=FALSE)
hist(rnorm(100))
par(ask=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 <code>D4TALinkTask</code> , as created by <code>initTask</code> .
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.

**Value**

File path.

---

datasourceFn	<i>Get path of data source file.</i>
--------------	--------------------------------------

---

**Description**

Get path of data source file.

**Usage**

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

**Arguments**

task	Object of class <code>D4TALinkTask</code> , as created by <code>initTask</code> .
filename	name of the input file.
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.

**Value**

File path.

---

docDir	<i>Get path of documentation directory.</i>
--------	---

---

**Description**

Get path of documentation directory.

**Usage**

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

**Arguments**

task	Object of class <code>D4TAlinkTask</code> , as created by <code>initTask</code> .
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.

**Value**

File path.

---

docFn	<i>Get path of documentation file.</i>
-------	--

---

**Description**

Get path of documentation file.

**Usage**

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

**Arguments**

task	Object of class <code>D4TAlinkTask</code> , as created by <code>initTask</code> .
type	Filename type. If the type is an array, the cocatenation 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	<i>Generic function.</i>
-----	--------------------------

---

**Description**

Generic function.

**Usage**

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

**Arguments**

sponsor	Sponsor name, default set by <a href="#">setTaskSponsor</a> .
task	Object of class <a href="#">D4TAlinkTask</a> , as created by <a href="#">initTask</a> .

**Value**

NULL.

---

formatTaskDocx	<i>Replace default task fields in 'docx' file.</i>
----------------	--

---

**Description**

Replace default task fields in 'docx' file.

**Usage**

```
formatTaskDocx(task, ifn)
```

**Arguments**

task	Object of class <a href="#">D4TAlinkTask</a> , as created by <a href="#">initTask</a> .
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 tasks 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	<i>Get the path of a file.</i>
-----------------	--------------------------------

---

**Description**

Get the path of a file.

**Usage**

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

**Arguments**

task	Object of class <a href="#">D4TALinkTask</a> , as created by <a href="#">initTask</a> .
type	Filename type. If the type is an array, the cocatenation 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	<i>Get the paths of the task.</i>
--------------	-----------------------------------

---

**Description**

Get the paths of the task.

**Usage**

```
getTaskPaths(task)
```

**Arguments**

task	Object of class <a href="#">D4TALinkTask</a> , as created by <a href="#">initTask</a> .
------	---

**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**

```
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.*

---

**Description**

Get the name of the task sponsor.

**Usage**

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

**Arguments**

`quiet`                suppress error messages, default=FALSE.

**Value**

The current name of the tasks sponsor.

**Examples**

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

---

getTaskStructure	<i>Get repository directory structure.</i>
------------------	--

---

**Description**

Get repository directory structure.

**Usage**

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

**Arguments**

quiet                    suppress error messages, default=FALSE.

**Value**

The directory structure function.

---

initTask	<i>Initialize a task</i>
----------	--------------------------

---

**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 <a href="#">setTaskSponsor</a> .
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	<i>Create task template in Rmd format.</i>
-------------	--

---

**Description**

Create task template in Rmd format.

**Usage**

```
initTaskRmd(task, encoding = "unknown", overwrite = FALSE, suffix = NA)
```

**Arguments**

task	Object of class <a href="#">D4TAlinkTask</a> , as created by <a href="#">initTask</a> .
encoding	encoding to be assumed for input strings. It is used to mark character strings as known to be in Latin-1, UTF-8 or to be bytes: it is not used to re-encode the input. To do the latter, specify the encoding as part of the connection con or via <a href="#">options</a> (encoding=): see the examples and ‘Details’.
overwrite	overwrite Rmd file if exists, default FALSE
suffix	Filename suffix, used to develop scripts for sub-analyses for a given task, default NA.

**Value**

the file name invisibly.

---

initTaskRscript	<i>Create task R script.</i>
-----------------	------------------------------

---

**Description**

Create task R script.

**Usage**

```
initTaskRscript(task, overwrite = FALSE, encoding = "unknown", suffix = NA)
```

**Arguments**

task	Object of class <a href="#">D4TALinkTask</a> , as created by <a href="#">initTask</a> .
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, UTF-8 or to be bytes: it is not used to re-encode the input. To do the latter, specify the encoding as part of the connection con or via <a href="#">options</a> (encoding=): see the examples and 'Details'.
suffix	Filename suffix, used to develop scripts for sub-analyses for a given task, default NA.

**Value**

the file name invisibly.

---

jpegReport	<i>Graphics devices for JPEG format bitmap files.</i>
------------	---

---

**Description**

Graphics devices for JPEG format bitmap files.

**Usage**

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

**Arguments**

task	Object of class <a href="#">D4TALinkTask</a> , as created by <a href="#">initTask</a> .
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 <a href="#">grDevices::jpeg</a>
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 <a href="#">path.expand</a> ). (The result must be less than 600 characters long. See <a href="#">postscript</a> 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 <code>par("bg")</code> .
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 <a href="#">windows</a> . For type == "windows" only.
antialias	Length-one character vector. For allowed values and their effect on fonts with type = "windows" see <a href="#">windows</a> : for that type if the argument is missing the default is taken from <code>windows.options()\$bitmap.aa.win</code> . For allowed values and their effect (on fonts and lines, but not fills) with type = "cairo" see <a href="#">svg</a> .
symbolfamily	For cairographics only: a length-one character string that specifies the font family to be used as the "symbol" font (e.g., for <a href="#">plotmath</a> 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	<i>Get path of jpeg output file.</i>
--------------	--------------------------------------

---

**Description**

Get path of jpeg output file.

**Usage**

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

**Arguments**

task	Object of class <a href="#">D4TALinkTask</a> , as created by <a href="#">initTask</a> .
type	Filename type. If the type is an array, the cocatenation 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	<i>List the files associated to a task.</i>
---------------	---

---

**Description**

List the files associated to a task.

**Usage**

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

**Arguments**

task	Object of class <a href="#">D4TALinkTask</a> , as created by <a href="#">initTask</a> .
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	<i>List details of all tasks stored in the root directory.</i>
-----------	--

---

**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 <a href="#">setTaskSponsor</a> .
rootpath	Path of the task repository, default set by <a href="#">setTaskRoot</a> .

**Value**

[data.frame](#) with the following information for tasks "sponsor", "project", "package", "task".

---

loadTask	<i>Load a task.</i>
----------	---------------------

---

**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 <a href="#">setTaskSponsor</a> .
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	<i>Task paths generator.</i>
--------------	------------------------------

---

**Description**

The paths are: datasrc: [ROOT]/[sponsor]/[project]/[package]/raw/datasource data: [ROOT]/[sponsor]/[project]/[package]/output/data  
 bin: [ROOT]/[sponsor]/[project]/[package]/output/[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 <a href="#">setTaskSponsor</a> .

**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/data bin: [ROOT]/[sponsor]/[project]/[package]/output/adhoc/[taskname]/bin code: [ROOT]/[sponsor]/[project]/[package]/progs code: [ROOT]/[sponsor]/[project]/[package]/progs 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 <a href="#">setTaskSponsor</a> .

### Value

a list of file paths

---

pathsPMS                      *Task paths generator.*

---

### Description

The paths are: datasrc: [ROOT]/[sponsor]/PMS\_data/[project]/[package]/datasource data: [ROOT]/[sponsor]/PMS\_data/[project]/[package]/output/data bin: [ROOT]/[sponsor]/PMS\_data/[project]/[package]/[taskname]/bin code: [ROOT]/[sponsor]/PMS\_code/[project]/[package]/progs code: [ROOT]/[sponsor]/PMS\_code/[project]/[package]/progs doc: [ROOT]/[sponsor]/PMS\_documentation/[project]/[package]/[taskname] log: [ROOT]/[sponsor]/PMS\_data/[project]/[package]/output/log

### Usage

pathsPMS(project, package, taskname, sponsor)

### Arguments

project	Project name.
package	Package name.
taskname	Task name.
sponsor	Sponsor name, default set by <a href="#">setTaskSponsor</a> .

### 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 <a href="#">D4TALinkTask</a> , as created by <a href="#">initTask</a> .
type	Filename type. If the type is an array, the cocatenation 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 <a href="#">postscript</a> for further details.) Tilde expansion (see <a href="#">path.expand</a> ) 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 <code>grDevices::pdf</code>
width,height	the width and height of the graphics region in inches. The default values are 7.
onfile	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 <code>postscript</code> . 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 papersize 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 <code>postscript</code> 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".
pointsize	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 <i>via</i> the Dingbats font? Defaults to FALSE. If TRUE, this can produce smaller and better output, but can cause 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 <code>polygon</code> for details. Defaults to FALSE.
compress	logical. Should PDF streams be generated with Flate compression? Defaults to TRUE.

**Value**

the file name invisibly.

---

pdfReportFn	<i>Get path of pdf output file.</i>
-------------	-------------------------------------

---

**Description**

Get path of pdf output file.

**Usage**

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

**Arguments**

task	Object of class <a href="#">D4TALinkTask</a> , as created by <a href="#">initTask</a> .
type	Filename type. If the type is an array, the cocatenation of the elements is used with separator "-". Filenames have the form [task name]_[type].[ext]
ext	Filename extension.
subdir	(optional) Subdirectory.

**Value**

File path.

---

pngReport	<i>Graphics devices for PNG format bitmap files.</i>
-----------	--

---

**Description**

Graphics devices for PNG format bitmap files.

**Usage**

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

**Arguments**

task	Object of class <a href="#">D4TALinkTask</a> , as created by <a href="#">initTask</a> .
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 <a href="#">grDevices::png</a>
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 <a href="#">path.expand</a> ). (The result must be less than 600 characters long. See <a href="#">postscript</a> 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 <code>par("bg")</code> .
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 <a href="#">windows</a> . For type == "windows" only.
antialias	Length-one character vector. For allowed values and their effect on fonts with type = "windows" see <a href="#">windows</a> : for that type if the argument is missing the default is taken from <code>windows.options()\$bitmap.aa.win</code> . For allowed values and their effect (on fonts and lines, but not fills) with type = "cairo" see <a href="#">svg</a> .
symbolfamily	For cairographics only: a length-one character string that specifies the font family to be used as the "symbol" font (e.g., for <a href="#">plotmath</a> 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.

---

pngReportFn	<i>Get path of png output file.</i>
-------------	-------------------------------------

---

**Description**

Get path of png output file.

**Usage**

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

**Arguments**

task	Object of class <a href="#">D4TAlinkTask</a> , as created by <a href="#">initTask</a> .
type	Filename type. If the type is an array, the cocatenation 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	<i>Get path of scripts directory.</i>
---------	---------------------------------------

---

**Description**

Get path of scripts directory.

**Usage**

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

**Arguments**

task	Object of class <a href="#">D4TAlinkTask</a> , as created by <a href="#">initTask</a> .
subdir	(optional) Subdirectory.
dirCreate	Logical, if TRUE (by default) the directory is created.

**Value**

File path.

---

readBinary	<i>Restore R object from binary file.</i>
------------	---

---

**Description**

Restore R object from binary file.

**Usage**

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

**Arguments**

task	Object of class <code>D4TAlinkTask</code> , as created by <code>initTask</code> .
type	Filename type. If the type is an array, the cocatenation 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 exists.
password	encryption password, default NULL

**Value**

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

---

readReportJSON	<i>Read JSON data into R object.</i>
----------------	--------------------------------------

---

**Description**

Read JSON data into R object.

**Usage**

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

**Arguments**

task	Object of class <a href="#">D4TALinkTask</a> , as created by <a href="#">initTask</a> .
type	Filename type. If the type is an array, the cocatenation 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	<i>Read data into vector or list using function <a href="#">scan</a>.</i>
-----------------	---

---

**Description**

Read data into vector or list using function [scan](#).

**Usage**

```
readReportTable(task, type, ext = "csv", subdir = NULL, dirCreate = FALSE, ...)
```

**Arguments**

task	Object of class <a href="#">D4TALinkTask</a> , as created by <a href="#">initTask</a> .
type	Filename type. If the type is an array, the cocatenation 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 <a href="#">utils::read.csv</a>

**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, [getwd\(\)](#). Tilde-expansion is performed where supported. This can be a compressed file (see [file](#)).

Alternatively, file can be a readable text-mode [connection](#) (which will be opened for reading if necessary, and if so [closed](#) (and hence destroyed) at the end of the function call). (If [stdin\(\)](#) is used, the prompts for lines may be somewhat confusing. Terminate input with a blank line or an EOF signal, Ctrl-D on Unix and Ctrl-Z on Windows. Any pushback on [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 [url](#).)

- `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: `header` is set to TRUE if and only if the first row contains one fewer field than the number of columns.
- `sep` the field separator character. Values on each line of the file are separated by this character. If `sep = ""` (the default for `read.table`) the separator is 'white space', that is one or more spaces, tabs, newlines or carriage returns.
- `quote` the set of quoting characters. To disable quoting altogether, use `quote = ""`. See [scan](#) for the behaviour on quotes embedded in quotes. Quoting is only considered for columns read as character, which is all of them unless `colClasses` is specified.
- `dec` the character used in the file for decimal points.
- `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	<i>Render the task from the Rmd file</i>
---------------	--

---

**Description**

The template of the task is rendered towards pdf or html in the documentation directory of the specified task. Note that on windows, Gnu zip may be required. The path to the executable must be added to the environment variables.

**Usage**

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

**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. <code>html_document()</code> ). 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 <code>output_format</code> , the options specified in the YAML header or <code>_output.yml</code> 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 <code>_output.yml</code> .
debug	if TRUE execute in the global environment.
clean	Using TRUE will clean intermediate files that are created during rendering.
suffix	Filename suffix, used to develop scripts for sub-analyses for a given task, default NA.
...	Arguments passed on to <code>rmarkdown::render</code>
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 <code>output_dir</code> option allows for specifying the output file path as well, however, if also specifying the path, the directory must exist. If <code>output_file</code> 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 <code>output_file</code> value in <code>I()</code> , e.g., <code>I('my-output')</code> .
output_dir	The output directory for the rendered <code>output_file</code> . 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 <code>output_file</code> the directory specified here will take precedence. Please note that any directory path provided will create any necessary directories if they do not exist.
output_options	List of output options that can override the options specified in metadata (e.g. <code>could</code> be used to force <code>self_contained</code> or <code>mathjax = "local"</code> ). Note that this is only valid when the output format is read from metadata (i.e. not a custom format object passed to <code>output_format</code> ).
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 <code>output_yaml</code> top-level parameter in the YAML front matter, <code>_output.yml</code> or <code>_output.yaml</code> , and then uses the first existing one.
intermediates_dir	Intermediate files directory. If a path is specified then intermediate files will be written to that path. If NULL, intermediate files are written to the same directory as the input file.
knit_root_dir	The working directory in which to knit the document; uses knitr's <code>root.dir</code> knitr option. If NULL then the behavior will follow the knitr default, which is to use the parent directory of the document.



**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 [run](#)). The default, `auto`, allows the runtime target specified in the YAML metadata to take precedence, and renders for a static runtime target otherwise.

**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.

**knit\_meta** (This option is reserved for expert use.) Metadata generated by **knitr**.

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

**run\_pandoc** An option for whether to run pandoc to convert Markdown output.

**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 `rmarkdown.render.message` to `FALSE`

**encoding** Ignored. The encoding is always assumed to be UTF-8.

**Value**

the file name invisibly.

---

reportDir	<i>Get path of report directory.</i>
-----------	--------------------------------------

---

**Description**

Get path of report directory.

**Usage**

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

**Arguments**

task	Object of class <code>D4TALinkTask</code> , as created by <code>initTask</code> .
subdir	(optional) Subdirectory.
dirCreate	Logical, if <code>TRUE</code> (by default) the directory is created.

**Value**

File path.

---

reportFn	<i>Get path of output file.</i>
----------	---------------------------------

---

**Description**

Get path of output file.

**Usage**

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

**Arguments**

task	Object of class <a href="#">D4TALinkTask</a> , as created by <a href="#">initTask</a> .
type	Filename type. If the type is an array, the cocatenation 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	<i>Get path of xlsx output file.</i>
-------------	--------------------------------------

---

**Description**

Get path of xlsx output file.

**Usage**

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

**Arguments**

task	Object of class <a href="#">D4TALinkTask</a> , as created by <a href="#">initTask</a> .
type	Filename type. If the type is an array, the cocatenation 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	<i>Restore an archive containing the files of a given task from a file created with archiveTask.</i>
-------------	--

---

### Description

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

### Usage

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

### Arguments

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

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	<i>Get path of R script file name.</i>
-------	--

---

**Description**

Get path of R script file name.

**Usage**

```
rmdFn(task, suffix = NA)
```

**Arguments**

task	Object of class <a href="#">D4TALinkTask</a> , as created by <a href="#">initTask</a> .
suffix	Filename suffix, used to develop scripts for sub-analyses for a given task, default NA.

**Value**

File path.

---

rscriptFn	<i>Get path of R script file name.</i>
-----------	--

---

**Description**

Get path of R script file name.

**Usage**

```
rscriptFn(task, suffix = NA)
```

**Arguments**

task	Object of class <a href="#">D4TALinkTask</a> , as created by <a href="#">initTask</a> .
suffix	Filename suffix, used to develop scripts for sub-analyses for a given task, default NA.

**Value**

File path.

---

saveBinary	<i>Save R object in binary file.</i>
------------	--------------------------------------

---

**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 <code>D4TAlinkTask</code> , as created by <code>initTask</code> .
type	Filename type. If the type is an array, the cocatenation 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.
encrypt	encrypt the output, default: FALSE. If character string, then use the string as password.
ask	query encryption key from user, default FALSE.

**Value**

the file name invisibly.

---

saveBinaryE	<i>Save R object in encrypted binary file.</i>
-------------	--

---

**Description**

Save R object in encrypted binary file.

**Usage**

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

**Arguments**

object	R object to serialize.
task	Object of class <code>D4TALinkTask</code> , as created by <code>initTask</code> .
type	Filename type. If the type is an array, the cocatenation 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.

**Value**

the file name invisibly.

---

saveReportJSON	<i>Output R object in JSON format.</i>
----------------	--

---

**Description**

Output R object in JSON format.

**Usage**

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

**Arguments**

x	R object to output.
task	Object of class <code>D4TALinkTask</code> , as created by <code>initTask</code> .
type	Filename type. If the type is an array, the cocatenation 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 <code>base::cat</code>

**Value**

the file name invisibly.

---

saveReportTable	<i>Output R object using function <a href="#">write.csv</a>.</i>
-----------------	--

---

**Description**

Output R object using function [write.csv](#).

**Usage**

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

**Arguments**

x	R object to output.
task	Object of class <a href="#">D4TALinkTask</a> , as created by <a href="#">initTask</a> .
type	Filename type. If the type is an array, the cocatenation 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 <a href="#">utils::write.csv</a>

**Value**

the file name invisibly.

---

 saveReportXls

*Save R object in binary file.*


---

### Description

Save R object in binary file.

### Usage

```
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 <code>AnnotatedDataFrame</code> , or a list thereof.
task	Object of class <code>D4TALinkTask</code> , as created by <code>initTask</code> .
type	Filename type. If the type is an array, the cocatenation 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` array of metadata field names to be appended in header of tables.  
`...` 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 <a href="#">D4TALinkTask</a> , as created by <a href="#">initTask</a> .
type	Filename type. If the type is an array, the cocatenation 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 <a href="#">type</a> of what gives the type of data to be read. (Here ‘type’ is used in the sense of <a href="#">typeof</a> .) The supported types are logical, integer, numeric, complex, character, raw and <a href="#">list</a> . 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 ( <a href="#">atomic</a> ) types listed or NULL, see section ‘Details’ below.
...	Arguments passed on to <a href="#">base::scan</a>
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 <a href="#">stdin()</a> reads if input is redirected or <a href="#">R</a> 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 <i>relative</i> to the current working directory (given by <a href="#">getwd()</a> ), unless it specifies an <i>absolute</i> path. Tilde-expansion is performed where supported. When running <a href="#">R</a> from a script, file = "stdin" can be used to refer to the process’s stdin file stream. This can be a compressed file (see <a href="#">file</a> ). Alternatively, file can be a <a href="#">connection</a> , which will be opened if necessary, 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 <a href="#">url</a> .) 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 <a href="#">file</a> 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 than 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 backslash. 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.

skipNul logical: should nuls be skipped when reading character fields?

## Value

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

---

setTaskAuthor	<i>Set the name of the tasks author.</i>
---------------	--

---

## 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	<i>Set the encryption key for binary data files.</i>
---------------	--

---

**Description**

Set the encryption key for binary data files.

**Usage**

```
setTaskEnckey(key)
```

**Arguments**

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

**Value**

NULL invisibly.

---

setTaskRmdTemplate	<i>Set the path to the Rmd task template.</i>
--------------------	---

---

**Description**

Set the path to the Rmd task template.

**Usage**

```
setTaskRmdTemplate(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, UTF-8 or to be bytes: it is not used to re-encode the input. To do the latter, specify the encoding as part of the connection con or via <code>options(encoding=)</code> : see the examples and 'Details'.

**Value**

The path to the Rmd task template invisibly.

---

setTaskRoot	<i>Set the root of the task repository.</i>
-------------	---

---

**Description**

Set the root of the task repository.

**Usage**

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

**Arguments**

rootpath	Path of the task repository, default set by <a href="#">setTaskRoot</a> .
dirCreate	Logical, if TRUE (by default) the directory is created.

**Value**

Path to the current task root.

---

setTaskRscriptTemplate	<i>Set the path to the R script task template.</i>
------------------------	--

---

**Description**

Set the path to the R script task template.

**Usage**

```
setTaskRscriptTemplate(file)
```

**Arguments**

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

**Value**

The path to the Rmd task template invisibly.

---

setTaskSponsor	<i>Set the name of the tasks sponsor.</i>
----------------	---

---

**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**

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

---

setTaskStructure	<i>Set task repository directory structure.</i>
------------------	---

---

**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**

```

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	<i>Get task identifier string.</i>
--------	------------------------------------

---

**Description**

Get task identifier string.

**Usage**

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

**Arguments**

task	Object of class <a href="#">D4TAlinkTask</a> , as created by <a href="#">initTask</a> .
sep	the field separator character, default: "/".

**Value**

String with task ID as:[sponsor][sep][project][sep][package][sep][task]



# Index

archiveTask, 3  
atomic, 42

base::cat, 5, 38  
base::scan, 42  
binaryDir, 4  
binaryFn, 4

catReport, 5  
close, 30  
connection, 30, 42

D4getenv, 6  
D4TAlink-common-args, 6  
D4TAlinkTask, 3–6, 7, 9–11, 13, 17–20, 22,  
24, 26–31, 33, 34, 36–40, 42, 48

data.frame, 21  
datasourceDir, 8  
datasourceFn, 9  
docDir, 10  
docFn, 10  
DTx, 11

file, 30, 42, 44  
formatTaskDocx, 11

getTaskAuthor, 12  
getTaskEnckey, 12  
getTaskFilepath, 13  
getTaskPaths, 13  
getTaskRmdTemplate, 14  
getTaskRoot, 14  
getTaskRscriptTemplate, 15  
getTaskSponsor, 15  
getTaskStructure, 16  
getwd, 30, 42  
grDevices::jpeg, 19  
grDevices::pdf, 25  
grDevices::png, 27

I, 32

initTask, 3–7, 9–11, 13, 16, 17–20, 24,  
26–31, 33, 34, 36–40, 42, 48  
initTaskRmd, 17  
initTaskRscript, 18

jpegReport, 18  
jpegReportFn, 20

list, 42  
listTaskFiles, 20  
listTasks, 21  
loadTask, 21

NA, 43  
new.env, 33

options, 17, 18, 45

path.expand, 3, 19, 24, 27, 35  
pathsDefault, 22  
pathsGLPG, 7, 23, 47  
pathsPMS, 7, 23, 47  
pdfReport, 24  
pdfReportFn, 26  
plotmath, 19, 27  
pngReport, 26  
pngReportFn, 28  
polygon, 25  
postscript, 19, 24, 25, 27  
progDir, 28

Quotes, 44

readBinary, 29  
readReportJSON, 29  
readReportTable, 30  
renderTaskRmd, 31  
reportDir, 33  
reportFn, 34  
reportXlsFn, 34  
restoreTask, 35

rmarkdown::render, 32  
rmdFn, 36  
rscriptFn, 36  
run, 33

saveBinary, 37  
saveBinaryE, 37  
saveReportJSON, 38  
saveReportTable, 39  
saveReportXls, 40  
scan, 30, 31, 41  
scanReport, 41  
setTaskAuthor, 44  
setTaskEnckey, 45  
setTaskRmdTemplate, 45  
setTaskRoot, 6, 21, 46, 46  
setTaskRscriptTemplate, 46  
setTaskSponsor, 6, 11, 17, 21–23, 47, 47  
setTaskStructure, 47  
stdin, 30, 42  
svg, 19, 27

taskID, 7, 48  
type, 42  
typeof, 42

url, 30, 42  
utils::read.csv, 30  
utils::unzip, 35  
utils::write.csv, 39  
utils::zip, 3

windows, 19, 27  
write.csv, 39  
WriteXLS::WriteXLS, 41