Package ‘antaresEditObject’

November 8, 2021

Type Package
Title Edit an 'Antares' Simulation
Version 0.3.0
Description Edit an 'Antares' simulation before running it: create new areas, links, thermal clusters or binding constraints or edit existing ones. Update 'Antares' general & optimization settings. 'Antares' is an open source power system generator, more information available here: <https://antares-simulator.org/>.
License GPL (>= 2) | file LICENSE
BugReports https://github.com/rte-antares-rpackage/antaresEditObject/issues
Encoding UTF-8
RoxygenNote 7.1.2
Depends antaresRead (>= 2.2.7)
Imports assertthat, grDevices, data.table, whisker, doParallel, pbapply, parallel
Suggests testthat, covr, knitr, rmarkdown
VignetteBuilder knitr
NeedsCompilation no
Author Veronique Bachelier [aut, cre], Frederic Breant [aut], Victor Perrier [aut], Baptiste Seguinot [ctb], Benoit Thiermu [ctb], Titouan Robert [ctb], Jalal-Edine Zawam [ctb], Etienne Sanchez [ctb], Janus De Bondt [ctb], RTE [cph]
Maintainer Veronique Bachelier <veronique.bachelier@rte-france.com>
Repository CRAN
Date/Publication 2021-11-08 11:30:02 UTC
R topics documented:

activateRES .......................................................... 3
backupStudy .......................................................... 3
checkRemovedArea .................................................. 4
computeTimeStampFromHourly .................................. 5
copyOutput .......................................................... 6
create-cluster ....................................................... 6
createArea ........................................................... 10
createBindingConstraint ......................................... 11
createDistrict ....................................................... 12
createDSR ........................................................... 13
createLink .......................................................... 15
createPSP ............................................................ 17
createStudy .......................................................... 19
dicoGeneralSettings .............................................. 19
dicoOptimizationSettings ........................................ 20
edit-cluster ......................................................... 20
editArea ............................................................. 22
editBindingConstraint ............................................ 23
editLink ............................................................. 24
filteringOptions ................................................... 26
getPlaylist ........................................................... 27
is_antares_v7 ......................................................... 27
nodalOptimizationOptions ....................................... 28
propertiesLinkOptions .......................................... 29
readIniFile .......................................................... 30
remove-cluster ...................................................... 30
removeArea .......................................................... 31
removeBindingConstraint ....................................... 32
removeLink .......................................................... 33
runSimulation ....................................................... 33
runTsGenerator ..................................................... 34
scenario-builder ................................................... 35
setPlaylist .......................................................... 37
setSolverPath ....................................................... 38
updateGeneralSettings .......................................... 38
updateInputSettings .............................................. 41
updateOptimizationSettings .................................... 42
updateOutputSettings ............................................ 43
writeEconomicOptions ........................................... 44
writeIni ............................................................. 45
writeInputTS ........................................................ 46
writeOutputValues ............................................... 46
writeSeriesPrepro ................................................ 47
writeWaterValues .................................................. 48

Index 50
activateRES  

**Activate RES in an Antares study**

**Description**

Helper to activate Renewables Energy Sources. This will update `renewable.generation.modelling` parameter and create appropriate structure for RES clusters.

**Usage**

```r
activateRES(opts = antaresRead::simOptions(), quietly = !interactive())
```

**Arguments**

- `opts` List of simulation parameters returned by the function `antaresRead::setSimulationPath`
- `quietly` Display or not a message to the user if success.

**Value**

An updated list containing various information about the simulation.

**Examples**

```r
## Not run:
library(antaresEditObject)
tmp <- tempfile()
createStudy(path = tmp)
opts <- antaresRead::setSimulationPath(tmp)
activateRES()

# then you can use createClusterRES()...

## End(Not run)
```

backUpStudy  

**Create a backup with an Antares Study**

**Description**

Save an Antares Study or only inputs in a `.tar.gz` file
checkRemovedArea

Usage

backupStudy(
  backupfile,
  what = c("input", "study"),
  opts = antaresRead::simOptions()
)

Arguments

backupfile Name of the backup, without extension. If missing, either the name of the study or 'input' according argument what.
what Which folder to save, input for the input folder or study for the whole study.
opts List of simulation parameters returned by the function antaresRead::setSimulationPath

Value

The path of the backup

Examples

## Not run:
backupStudy()
## End(Not run)

checkRemovedArea Seek for a removed area

Description

Check if it remains trace of a deleted area in the input folder

Usage

checkRemovedArea(area, all_files = TRUE, opts = antaresRead::simOptions())

Arguments

area An area
all_files Check files in study directory.
opts List of simulation parameters returned by the function antaresRead::setSimulationPath

Value

a named list with two elements
computeTimeStampFromHourly

Description

Compute daily, weekly, monthly and annual mc-ind from hourly data.

Usage

computeTimeStampFromHourly(
  opts,
  mcYears = "all",
  nbcl = 8,
  verbose = 1,
  type = c("areas", "links", "clusters")
)

Arguments

  opts          opts simulation path.
  mcYears      mcYears to compute.
  nbcl          number of thread for parallel computing.
  verbose       verbose for execution.
  type          type of file to compute.

Examples

## Not run:
library(antaresEditObject)
opts <- setSimulationPath("my_study")
computeTimeStampFromHourly(opts)

## End(Not run)
create-cluster

Description

Copy of the output files of an Antares study

Usage

copyOutput(opts, extname, mcYears = "all")

Arguments

- **opts**: List of simulation parameters returned by the function `antaresRead::setSimulationPath`
- **extname**: Extension to be added to the name of the study, to be used as a name for the newly created folder.
- **mcYears**: mcYears to copy. Can be "all".

Examples

```r
## Not run:
library(antaresRead)

# Set simulation path
opts = setSimulationPath(path = "PATH/TO/SIMULATION", simulation = "input")

# Create a new area
copyOutput(opts, ".adq")
```

## End(Not run)

create-cluster

Create a cluster

Description

Create a new thermal or RES (renewable energy source) cluster.
create-cluster

Usage

createCluster(
  area,
  cluster_name,
  group = "Other",
  ...
  time_series = NULL,
  prepro_data = NULL,
  prepro_modulation = NULL,
  add_prefix = TRUE,
  overwrite = FALSE,
  opts = antaresRead::simOptions()
)

createClusterRES(
  area,
  cluster_name,
  group = "Other RES 1",
  ...
  time_series = NULL,
  add_prefix = TRUE,
  overwrite = FALSE,
  opts = antaresRead::simOptions()
)

Arguments

area The area where to create the cluster.
cluster_name Name for the cluster, it will prefixed by area name, unless you set add_prefix = FALSE.
group Group of the cluster, depends on cluster type:
  • thermal cluster, one of: Gas, Hard coal, Lignite, Mixed fuel, Nuclear, Oil, Other, Other 2, Other 3, Other 4.
  • renewable cluster, one of: Wind Onshore, Wind Offshore, Solar Thermal, Solar PV, Solar Rooftop, Other RES 1, Other RES 2, Other RES 3, Other RES 4.
...
Parameters to write in the Ini file. Careful! Some parameters must be set as integers to avoid warnings in Antares, for example, to set unitcount, you’ll have to use unitcount = 1L.
time_series the "ready-made" 8760-hour time-series available for simulation purposes.
prepro_data Pre-process data, a data.frame or matrix, default is a matrix with 365 rows and 6 columns.
prepro_modulation Pre-process modulation, a data.frame or matrix, if specified, must have 8760 rows and 1 or 4 columns.
add_prefix If TRUE (the default), cluster_name will be prefixed by area name.
create-cluster

overwrite Logical, overwrite the cluster or not.

opts List of simulation parameters returned by the function `antaresRead::setSimulationPath()`

Value
An updated list containing various information about the simulation.

See Also
`editCluster()` or `editClusterRES()` to edit existing clusters, `removeCluster()` or `removeClusterRES()` to remove clusters.

Examples

```r
## Not run:

library(antaresRead)
library(antaresEditObject)

# Create a cluster :
createCluster(
  area = "fr",
  cluster_name = "my_cluster",
  group = "other",
  unitcount = 1L, # or as.integer(1)
  marginal_cost = 50
)
# by default, cluster name is prefixed
# by the area name
levels(readClusterDesc()$cluster)
# > "fr_my_cluster"

# To prevent this, use `add_prefix`
createCluster(
  area = "fr",
  cluster_name = "my_cluster",
  add_prefix = FALSE,
  group = "other",
  marginal_cost = 50
)
levels(readClusterDesc()$cluster)
# > "my_cluster"

# Create a RES cluster :
createClusterRES(
  area = "fr",
  cluster_name = "my_cluster_res",
  group = "other",
  unitcount = 1L, # or as.integer(1)
  nominal_capacity = 50,
  ts_interpretation = "power-generation"
)
```

create-cluster

)

# You can also specify that the Time-Series of the RES cluster are
# production factors:
createClusterRES(
  area = "fr",
  cluster_name = "my_cluster_res",
  group = "other",
  unitcount = 1L, # or as.integer(1)
  nominalcapacity = 50,
  ts_interpretation = "production-factor"
)

# Pre-process data:

# this is the default data:
createCluster(
  area = "fr",
  cluster_name = "my_cluster",
  prepro_data = matrix(
    data = c(rep(1, times = 365 * 2),
             rep(0, times = 365 * 4)),
    ncol = 6
  )
)

# with a data.frame
createCluster(
  area = "fr",
  cluster_name = "my_cluster",
  prepro_data = data.frame(
    v1 = rep(7, 365), # column name doesn't matter
    v2 = rep(27, 365),
    v3 = rep(0.05, 365),
    v4 = rep(0.12, 365),
    v5 = rep(0, 365),
    v6 = rep(1, 365)
  )
)

# Pre-process modulation:

# this is the default data
createCluster(
  area = "fr",
  cluster_name = "my_cluster",
  prepro_modulation = matrix(
    data = c(rep(1, times = 365 * 24 * 3),
             rep(0, times = 365 * 24 * 1)),
    ncol = 4
  )
)
# with a data.frame
createCluster(
    area = "fr",
    cluster_name = "my_cluster",
    prepro_modulation = data.frame(
        var1 = rep(0, 8760), # column name doesn't matter
        var2 = rep(1, 8760),
        var3 = rep(0, 8760),
        var4 = rep(1, 8760)
    )
)

## End(Not run)

---

**createArea**  
*Create An Area In An Antares Study*

**Description**
Create An Area In An Antares Study

**Usage**

```r
createArea(
    name,
    color = grDevices::rgb(230, 108, 44, max = 255),
    localization = c(0, 0),
    nodalOptimization = nodalOptimizationOptions(),
    filtering = filteringOptions(),
    overwrite = FALSE,
    opts = antaresRead::simOptions()
)
```

**Arguments**

- **name**  
  Name of the area as a character, without punctuation except - and _.

- **color**  
  Color of the node

- **localization**  
  Localization on the map

- **nodalOptimization**  
  Nodal optimization parameters, see `nodalOptimizationOptions`.

- **filtering**  
  Filtering parameters, see `filteringOptions`.

- **overwrite**  
  Overwrite the area if already exist.

- **opts**  
  List of simulation parameters returned by the function `antaresRead::setSimulationPath`
createBindingConstraint

Value

An updated list containing various information about the simulation.

Examples

## Not run:

```r
library(antaresRead)

# Set simulation path
setSimulationPath(path = "PATH/TO/SIMULATION", simulation = "input")

# Create a new area
createArea("fictive_area")
```

## End(Not run)

---

createBindingConstraint

Create a Binding Constraint

Description

Create a Binding Constraint

Usage

```r
createBindingConstraint(
  name,
  id = tolower(name),
  values = NULL,
  enabled = TRUE,
  timeStep = c("hourly", "daily", "weekly"),
  operator = c("both", "equal", "greater", "less"),
  coefficients = NULL,
  overwrite = FALSE,
  opts = antaresRead::simOptions()
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name for the binding constraint</td>
</tr>
<tr>
<td>id</td>
<td>An id</td>
</tr>
<tr>
<td>values</td>
<td>Values used by the constraint. It contains one line per time step and three columns &quot;less&quot;, &quot;greater&quot; and &quot;equal&quot;.</td>
</tr>
<tr>
<td>enabled</td>
<td>Logical, is the constraint enabled?</td>
</tr>
</tbody>
</table>
createDistrict

timeStep Time step the constraint applies to: hourly, daily or weekly
operator Type of constraint: equality, inequality on one side or both sides.
coefficients A named vector containing the coefficients used by the constraint.
overwrite If the constraint already exist, overwrite the previous value.
opts List of simulation parameters returned by the function antaresRead::setSimulationPath

Value

An updated list containing various information about the simulation.

Examples

```r
## Not run:
createBindingConstraint(
  name = "myconstraint",
  values = matrix(data = rep(0, 8760 * 3), ncol = 3),
  enabled = FALSE,
  timeStep = "hourly",
  operator = "both",
  coefficients = c("fr%myarea" = 1)
)

## End(Not run)
```

createDistrict Create a district

Description

Allows selecting a set of areas so as to bundle them together in a "district".

Usage

```r
createDistrict(
  name,
  caption = NULL,
  comments = NULL,
  apply_filter = "none",
  add_area = NULL,
  remove_area = NULL,
  output = FALSE,
  overwrite = FALSE,
  opts = antaresRead::simOptions()
)
```
createDSR

Arguments

- **name**: District’s name.
- **caption**: Caption for the district.
- **comments**: Comments for the district.
- **apply_filter**: Possible values are `add-all` to add all areas to the district, `remove-all` to clear the district, or `none` (default) to don’t apply a filter.
- **add_area**: Character vector of area(s) to add to the district.
- **remove_area**: Character vector of area(s) to remove from the district.
- **output**: Logical, compute the results for the district or not?
- **overwrite**: Logical, should the district be overwritten if already exist?
- **opts**: List of simulation parameters returned by the function `antaresRead::setSimulationPath`

Value

An updated list containing various information about the simulation.

Examples

```r
## Not run:
createDistrict(name = "mydistrict",
                apply_filter = "add-all",
                remove_area = c("fr", "be"))

## End(Not run)
```

createDSR

*Create a Demand Side Response (DSR)*

Description

Create a Demand Side Response (DSR)

Usage

```r
createDSR(
    areasAndDSRParam = NULL,
    spinning = 2,
    overwrite = FALSE,
    opts = antaresRead::simOptions()
)
```

```r
capacityDSR(area = NULL, opts = antaresRead::simOptions())
```

```r
eDitSRSR(
    area = NULL,
```


```r
unit = NULL,
nominalCapacity = NULL,
marginalCost = NULL,
spinning = NULL,
opts = antaresRead::simOptions()
)
```

**Arguments**

- `areasAndDSRParam`  
  A data.frame with 4 columns `area`, `unit`, `nominalCapacity`, `marginalCost` and `hour`. Hour represent the number of activation hours for the DSR per day.

- `spinning`  
  DSR spinning

- `overwrite`  
  Overwrite the DSR plant if already exist. This will overwrite the previous area and links.

- `opts`  
  List of simulation parameters returned by the function `antaresRead::setSimulationPath`

- `area`  
  an area where to edit the DSR

- `unit`  
  DSR unit number

- `nominalCapacity`  
  DSR nominalCapacity

- `marginalCost`  
  DSR marginalCost

**Value**

`createDSR()` and `editDSR()` returns an updated list containing various information about the simulation.

`getCapacityDSR()` returns DSR capacity (unit * nominalCapacity of virtual cluster) of the area

**Examples**

```r
# Not run:
library(antaresEditObject)
path<-pathToYourStudy
opts<-setSimulationPath(path, simulation = "input")
area, unit, nominalCapacity and marginalCost
dsrdData<-data.frame(area = c("a", "b"), unit = c(10,20),
                     nominalCapacity = c(100, 120), marginalCost = c(52, 65), hour = c(3, 7))
createDSR(dsrData)
createDSR(dsrData, spinning = 3, overwrite = TRUE)
getAreas()

# End(Not run)
```

```r
# Not run:
getCapacityDSR("a")
```
createLink  

Create a link between two areas

Description

Create a link between two areas

Usage

createLink(  
  from,  
  to,  
  propertiesLink = propertiesLinkOptions(),  
  dataLink = NULL,  
  overwrite = FALSE,  
  opts = antaresRead::simOptions()  
)

Arguments

from The first area from which to create a link
to The second one
propertiesLink a named list containing the link properties, e.g. hurdles-cost or transmission-capacities for example. See propertiesLinkOptions.
dataLink For Antares v7, a matrix with eight column corresponding to: trans. capacity (direct) trans. capacity (indirect), hurdles cost (direct), hurdles cost (indirect), impedances, loop flow, PST min, PST max. If NULL (default), a matrix whose rows are equal to 1,1,0,0,0,0,0,0 is set. See Details
overwrite Logical, overwrite the previous between the two areas if exist
opts List of simulation parameters returned by the function antaresRead::setSimulationPath
**Details**

The eight times-series are:

- **NTC direct** : the upstream-to-downstream capacity, in MW
- **NTC indirect** : the downstream-to-upstream capacity, in MW
- **Hurdle cost direct** : an upstream-to-downstream transmission fee, in euro/MWh
- **Hurdle cost indirect** : a downstream-to-upstream transmission fee, in euro/MWh
- **Impedances** : virtual impedances that are used in economy simulations to give a physical meaning to raw outputs, when no binding constraints have been defined to enforce Kirchhoff’s laws.
- **Loop flow** : amount of power flowing circularly though the grid when all "nodes" are perfectly balanced (no import and no export).
- **PST min** : lower bound of phase-shifting that can be reached by a PST installed on the link, if any.
- **PST max** : upper bound of phase-shifting that can be reached by a PST installed on the link, if any.

NB: For Antares v7 the eight columns must conform to above order. For Antares v6, only five columns are expected, and they must follow this other order: NTC direct, NTC indirect, Impedances, Hurdle cost direct, Hurdle cost indirect.

**Value**

An updated list containing various information about the simulation.

**Note**

In Antares, areas are sorted in alphabetical order to establish links between. For example, link between "fr" and "be" will appear under "be". So the areas are sorted before creating the link between them, and dataLink is rearranged to match the new order.

**Examples**

```r
## Not run:
library(antaresRead)

# Set simulation path
setSimulationPath(path = "PATH/TO/SIMULATION", simulation = "input")

# Create a link between two areas
createLink(from = "first_area", to = "second_area")

## End(Not run)
```
createPSP

Create a Pumped Storage Power plant (PSP)

Description
Create a Pumped Storage Power plant (PSP)

Usage
createPSP(
  areasAndCapacities = NULL,
  namePumping = "Psp_In",
  nameTurbining = "Psp_Out",
  hurdleCost = 5e-04,
  timeStepBindConstraint = "weekly",
  efficiency = NULL,
  overwrite = FALSE,
  opts = antaresRead::simOptions()
)

getCapacityPSP(
  area = NULL,
  nameTurbining = "Psp_Out",
  timeStepBindConstraint = "weekly",
  opts = antaresRead::simOptions()
)

editPSP(
  area = NULL,
  capacity = NULL,
  namePumping = "Psp_In",
  nameTurbining = "Psp_Out",
  timeStepBindConstraint = "weekly",
  hurdleCost = 5e-04,
  opts = antaresRead::simOptions()
)

Arguments
areasAndCapacities
  A data.frame with 2 columns installedCapacity and area
namePumping
  The name of the pumping area
nameTurbining
  The name of the turbining area
hurdleCost
  The cost of the PSP
timeStepBindConstraint
  Time step for the binding constraint : daily or weekly
createPSP

efficiency  The efficiency of the PSP
overwrite  Overwrite the Pumped Storage Power plant if already exist. This will overwrite the previous area and links.
opts  List of simulation parameters returned by the function antaresRead::setSimulationPath
area  an area name
capacity  PSP capacity for the area

Value
createPSP() and editPSP() returns an updated list containing various information about the simulation.
getCapacityPSP() returns PSP capacity of the area

Examples

```r
## Not run:
library(antaresEditObject)
path<-pathToYourStudy
opts<-setSimulationPath(path, simulation = "input")
pspData<-data.frame(area=c("a", "b"), installedCapacity=c(800,900))

createPSP(pspData, efficiency = 0.8)
createPSP(pspData, efficiency = 0.66, overwrite = TRUE)
createPSP(pspData, efficiency = 0.98, timeStepBindConstraint = "daily")
getAreas()

## End(Not run)

## Not run:

## Not run:

getCapacityPSP("a")
editPSP("a", capacity = 8000, hurdleCost = 0.1)
getCapacityPSP("a")

areaName<="suisse"
createArea(areaName, overwrite = TRUE)
pspData<-data.frame(area=c(areaName), installedCapacity=c(9856))
createPSP(pspData, efficiency = 0.5, overwrite = TRUE, timeStepBindConstraint = "daily")

getCapacityPSP(areaName, timeStepBindConstraint = "daily")

## End(Not run)
```
createStudy

Create an empty Antares study

**Description**

Create an empty Antares study

**Usage**

`createStudy(path, study_name = "my_study", antares_version = "8.1.0")`

**Arguments**

- **path**
  Path where to create study, it should be an empty directory, if it doesn’t exist, it’ll be created.
- **study_name**
  Name of the study.
- **antares_version**
  Antares number version.

**Value**

Result of `antaresRead::setSimulationPath()`.

**Examples**

```r
## Not run:
createStudy("path/to/simulation")
## End(Not run)
```

dicoGeneralSettings

Correspondence between arguments of updateGeneralSettings and actual Antares parameters.

**Description**

Correspondence between arguments of updateGeneralSettings and actual Antares parameters.

**Usage**

`dicoGeneralSettings(arg)`

**Arguments**

- **arg**
  An argument from function updateGeneralSettings.
### dicoGeneralSettings

**Description**

The corresponding Antares general parameter.

**Examples**

```python
dicoGeneralSettings("year.by.year") # "year-by-year"
```

### dicoOptimizationSettings

**Description**

Correspondence between arguments of `updateOptimizationSettings` and actual Antares parameters.

**Usage**

```python
dicoOptimizationSettings(arg)
```

**Arguments**

- **arg**
  
  An argument from function `updateOptimizationSettings`.

**Value**

The corresponding Antares general parameter.

**Examples**

```python
dicoGeneralSettings("year.by.year") # "year-by-year"
```

### edit-cluster

**Description**

Edit parameters of an existing cluster, thermal or RES (renewable energy source).
edit-cluster

Usage

```r
editCluster(
  area,
  cluster_name,
  ...,  
  time_series = NULL,
  prepro_data = NULL,
  prepro_modulation = NULL,
  add_prefix = TRUE,
  opts = antaresRead::simOptions()
)
```

```r
editClusterRES(
  area,
  cluster_name,
  ...,  
  time_series = NULL,
  add_prefix = TRUE,
  opts = antaresRead::simOptions()
)
```

Arguments

- **area**: The area where the cluster is.
- **cluster_name**: cluster name.
- **...**: Parameters to write in the Ini file.
- **time_series**: the "ready-made" 8760-hour time-series available for simulation purposes.
- **prepro_data**: Pre-process data, a data.frame or matrix, default is a matrix with 365 rows and 6 columns.
- **prepro_modulation**: Pre-process modulation, a data.frame or matrix, if specified, must have 8760 rows and 1 or 4 columns.
- **add_prefix**: If TRUE, cluster_name will be prefixed by area name.
- **opts**: List of simulation parameters returned by the function `antaresRead::setSimulationPath()`

Value

An updated list containing various information about the simulation.

See Also

- `createCluster()` or `createClusterRES()` to create new clusters, `removeCluster()` or `removeClusterRES()` to remove clusters.
Examples

## Not run:

# Update only nominalCapacity for an existing cluster
editCluster(
  area = "myarea",
  cluster_name = "mycluster",
  nominalcapacity = 10600.000
)

## End(Not run)

editArea   Edit An Existing Area In An Antares Study

Description

Edit An Existing Area In An Antares Study

Usage

eeditArea(
  name,
  color = NULL,
  localization = NULL,
  nodalOptimization = NULL,
  filtering = NULL,
  opts = antaresRead::simOptions()
)

Arguments

name        Name of the area as a character, without punctuation except - and _.
color       Color of the node
localization Localization on the map
odalOptimization Nodal optimization parameters, see nodalOptimizationOptions.
filtering  Filtering parameters, see filteringOptions.
opts        List of simulation parameters returned by the function antaresRead::setSimulationPath

Value

An updated list containing various information about the simulation.
**editBindingConstraint**

*Description*

Update An Existing Binding Constraint

*Usage*

```r
editBindingConstraint(
  name,
  id = tolower(name),
  values = NULL,
  enabled = NULL,
  timeStep = NULL,
  operator = NULL,
  coefficients = NULL,
  opts = antaresRead::simOptions()
)
```

*Arguments*

- **name**: The name for the binding constraint
- **id**: An id
- **values**: Values used by the constraint. It contains one line per time step and three columns "less", "greater" and "equal".

---

**Examples**

```r
## Not run:

library(antaresRead)

# Set simulation path
setSimulationPath(path = "PATH/TO/SIMULATION", simulation = "input")

# Edit an existing area
editArea("area", color = grDevices::rgb(230, 108, 44, max = 255),
         localization = c(1, 1),
         opts = antaresRead::simOptions())

editArea("de", nodalOptimization = list("spilledenergycost" = list(fr = 30)),
         opts = antaresRead::simOptions())

editArea("de", nodalOptimization = nodalOptimizationOptions(),
         opts = antaresRead::simOptions())

## End(Not run)
```
enabled Logical, is the constraint enabled?

timeStep Time step the constraint applies to: hourly, daily or weekly

operator Type of constraint: equality, inequality on one side or both sides.

coefficients A named vector containing the coefficients used by the constraint.

opts List of simulation parameters returned by the function `antaresRead::setSimulationPath`

Value

An updated list containing various information about the simulation.

Examples

```r
## Not run:
editBindingConstraint(
  name = "toto",
  values = matrix(data = rep(0, 8760 * 3), ncol = 3),
  enabled = FALSE,
  timeStep = "hourly",
  operator = "both",
  coefficients = c("fr%de" = 1)
)

## End(Not run)
```

editLink

`editLink` *Edit a link between two areas*

Description

Edit a link between two areas

Usage

```r
editLink(
  from,
  to,
  hurdles_cost = NULL,
  transmission_capacities = NULL,
  asset_type = NULL,
  display_comments = NULL,
  filter_synthesis = NULL,
  filter_year_by_year = NULL,
  dataLink = NULL,
  opts = antaresRead::simOptions()
)
```
Arguments

from The first area from which to create a link
to The second one
hurdles_cost Logical, which is used to state whether (linear) transmission fees should be taken into account or not in economy and adequacy simulations
transmission_capacities Character, one of enabled, ignore or infinite, which is used to state whether the capacities to consider are those indicated in 8760-hour arrays or if zero or infinite values should be used instead (actual values / set to zero / set to infinite)
asset_type Character, one of ac, dc, gas, virt or other. Used to state whether the link is either an AC component (subject to Kirchhoff’s laws), a DC component, or another type of asset.
display_comments Logical
filter_synthesis Output synthesis
filter_year_by_year Output year-by-year
dataLink For Antares v7, a matrix with eight column corresponding to : trans. capacity (direct) trans. capacity (indirect), hurdles cost (direct), hurdles cost (indirect), impedances, loop flow, PST min, PST max. If NULL (default), a matrix whose rows are equal to 1,1,0,0,0,0,0,0 is set. See Details
opts List of simulation parameters returned by the function antaresRead::setSimulationPath

Details

The eight times-series are:

- **NTC direct**: the upstream-to-downstream capacity, in MW
- **NTC indirect**: the downstream-to-upstream capacity, in MW
- **Hurdle cost direct**: an upstream-to-downstream transmission fee, in euro/MWh
- **Hurdle cost indirect**: a downstream-to-upstream transmission fee, in euro/MWh
- **Impedances**: virtual impedances that are used in economy simulations to give a physical meaning to raw outputs, when no binding constraints have been defined to enforce Kirchhoff’s laws.
- **Loop flow**: amount of power flowing circularly though the grid when all “nodes” are perfectly balanced (no import and no export).
- **PST min**: lower bound of phase-shifting that can be reached by a PST installed on the link, if any.
- **PST max**: upper bound of phase-shifting that can be reached by a PST installed on the link, if any.

NB: For Antares v7 the eight columns must conform to above order. For Antares v6, only five columns are expected, and they must follow this other order: NTC direct, NTC indirect, Impedances, Hurdle cost direct, Hurdle cost indirect.
filteringOptions

Value

An updated list containing various information about the simulation.

Note

In Antares, areas are sorted in alphabetical order to establish links between. For example, link between "fr" and "be" will appear under "be". So the areas are sorted before creating the link between them, and dataLink is rearranged to match the new order.

Examples

## Not run:
```r
editLink(
  from = "area1",
  to = "area2",
  transmission_capacities = "infinite"
)
## End(Not run)
```

filteringOptions  

Output profile options for creating an area

Description

Output profile options for creating an area

Usage

```r
filteringOptions(
  filter_synthesis = c("hourly", "daily", "weekly", "monthly", "annual"),
  filter_year_by_year = c("hourly", "daily", "weekly", "monthly", "annual")
)
```

Arguments

filter_synthesis
  Output synthesis
filter_year_by_year
  Output Year-by-year

Value

a named list

Examples

```r
filteringOptions()
```
getPlaylist

Get the playlist of an Antares study

Description

getPlaylist gives the identifier of the MC years which will be simulated in the Antares study, taking into account the potential use of a playlist which can skip some MC years.

Usage

getPlaylist(opts = antaresRead::simOptions())

Arguments

opts list of simulation parameters returned by the function antaresRead::setSimulationPath

Value

Returns a vector of the identifier of the simulated MC year

is_antares_v7

Is study an Antares v7 study?

Description

Is study an Antares v7 study?

Usage

is_antares_v7(opts = antaresRead::simOptions())

Arguments

opts List of simulation parameters returned by the function antaresRead::setSimulationPath

Value

a logical, TRUE if study is v7 or above, FALSE otherwise
Examples

```r
## Not run:
# setSimulationPath
is_antares_v7()

## End(Not run)
```

nodalOptimizationOptions

*Nodal optimization parameters for creating an area*

Description

Nodal optimization parameters for creating an area

Usage

```r
nodalOptimizationOptions(
  non_dispatchable_power = TRUE,
  dispatchable_hydro_power = TRUE,
  other_dispatchable_power = TRUE,
  spread_unsupplied_energy_cost = 0,
  spread_spilled_energy_cost = 0,
  average_unsupplied_energy_cost = 0,
  average_spilled_energy_cost = 0
)
```

Arguments

- `non_dispatchable_power`
  - logical, default to FALSE
- `dispatchable_hydro_power`
  - logical, default to FALSE
- `other_dispatchable_power`
  - logical, default to FALSE
- `spread_unsupplied_energy_cost`
  - numeric, default to 0
- `spread_spilled_energy_cost`
  - numeric, default to 0
- `average_unsupplied_energy_cost`
  - numeric, default to 0
- `average_spilled_energy_cost`
  - numeric, default to 0
Value

a named list

Examples

nodalOptimizationOptions()

---

**Properties for creating a link**

**Description**

Properties for creating a link

**Usage**

```r
propertiesLinkOptions(
  hurdles_cost = FALSE,
  transmission_capacities = "enabled",
  asset_type = "ac",
  display_comments = TRUE,
  filter_synthesis = c("hourly", "daily", "weekly", "monthly", "annual"),
  filter_year_by_year = c("hourly", "daily", "weekly", "monthly", "annual")
)
```

**Arguments**

- `hurdles_cost` Logical, which is used to state whether (linear) transmission fees should be taken into account or not in economy and adequacy simulations
- `transmission_capacities` Character, one of `enabled`, `ignore` or `infinite`, which is used to state whether the capacities to consider are those indicated in 8760-hour arrays or if zero or infinite values should be used instead (actual values / set to zero / set to infinite)
- `asset_type` Character, one of `ac`, `dc`, `gas`, `virt` or another. Used to state whether the link is either an AC component (subject to Kirchhoff’s laws), a DC component, or another type of asset.
- `display_comments` Logical
- `filter_synthesis` Output synthesis
- `filter_year_by_year` Output year-by-year

**Value**

A named list
Examples

```r
## Not run:
propertiesLinkOptions()

## End(Not run)
```

---

**readIniFile**  
*Read a INI file*

### Description

Read a INI file

### Usage

```r
readIniFile(file, stringsAsFactors = FALSE)
```

### Arguments

- **file**  
  file path.

- **stringsAsFactors**  
  logical: should character vectors be converted to factors?

### Value

A list with an element for each section of the .ini file.

---

**remove-cluster**  
*Remove a cluster*

### Description

Remove a cluster, thermal or RES (renewable energy source), and all its data.

### Usage

```r
removeCluster(
  area,
  cluster_name,
  add_prefix = TRUE,
  opts = antaresRead::simOptions()
)
```

```r
removeClusterRES(
  area,
  cluster_name,
)```
removeArea

    add_prefix = TRUE,
    opts = antaresRead::simOptions()
  )

Arguments

  area            Area from which to remove a cluster.
  cluster_name   Cluster to remove.
  add_prefix     If TRUE, cluster_name will be prefixed by area's name.
  opts           List of simulation parameters returned by the function antaresRead::setSimulationPath

Value

  An updated list containing various information about the simulation.

See Also

  createCluster() or createClusterRES() to create new clusters, editCluster() or editClusterRES() to edit existing clusters.

Examples

  ## Not run:
  createCluster(area = "fr", cluster_name = "fr_gas",
                 group = "other", `marginal-cost` = 50)
  removeCluster(area = "fr", cluster_name = "fr_gas")

  ## End(Not run)
removeBindingConstraint

Remove a Binding Constraint

Description
Remove a Binding Constraint

Usage
removeBindingConstraint(name, opts = antaresRead::simOptions())

Arguments
name       Name(s) of the binding constraint(s) to remove.
opts       List of simulation parameters returned by the function antaresRead::setSimulationPath

Value
An updated list containing various information about the simulation.

Examples
## Not run:
removeArea("fictive_area")

## End(Not run)

## Not run:
removeBindingConstraint("mybindingconstraint")

## End(Not run)
removeLink

Remove a link between two areas

Description
Remove a link between two areas

Usage
removeLink(from, to, opts = antaresRead::simOptions())

Arguments
from The first area from which to create a link
to The second one
opts List of simulation parameters returned by the function antaresRead::setSimulationPath

Value
An updated list containing various information about the simulation.

Examples
```r
## Not run:
createLink(from = "myarea", to = "myarea2")
removeLink(from = "myarea", to = "myarea2")
## End(Not run)
```

runSimulation

Run an Antares Simulation

Description
run_simulation is a function which runs an ANTARES study in economic mode

Usage
```r
runSimulation(
  name,
  mode = "economy",
  path_solver = getOption("antares.solver"),
  wait = TRUE,
  show_output_on_console = FALSE,
  parallel = TRUE,
  opts = antaresRead::simOptions()
)
```
Arguments

- **name**: Name of the simulation.
- **mode**: Simulation mode, can take value "economy", "adequacy" or "draft".
- **path_solver**: Character containing the Antares Solver path.
- **wait**: Logical, indicating whether the R interpreter should wait for the simulation to finish, or run it asynchronously.
- **show_output_on_console**: Logical, indicating whether to capture the ANTARES log and show it on the R console.
- **parallel**: Logical. If TRUE the ANTARES simulation will be run in parallel mode (Work only with ANTARES v6.0.0 or more). In that case, the number of cores used by the simulation is the one set in advanced_settings/simulation_cores (see ANTARES interface).
- **opts**: List of simulation parameters returned by the function `antaresRead::setSimulationPath`.

Value

The function does not return anything. It is used to launch an ANTARES simulation.

---

**runTsGenerator**  
*Run Time-Series Generator*

Description

Run Time-Series Generator

Usage

```r
runTsGenerator(
  path_solver = getOption("antares.solver"),
  wait = TRUE,
  show_output_on_console = FALSE,
  opts = antaresRead::simOptions()
)
```

Arguments

- **path_solver**: Character containing the Antares Solver path.
- **wait**: Logical, indicating whether the R interpreter should wait for the simulation to finish, or run it asynchronously.
- **show_output_on_console**: Logical, indicating whether to capture the ANTARES log and show it on the R console.
- **opts**: List of simulation parameters returned by the function `antaresRead::setSimulationPath`. 
**scenario-builder**

Read, create & update scenario builder

## Description

Read, create & update scenario builder

## Usage

```r
scenarioBuilder(
  n_scenario,
  n_mc = NULL,
  areas = NULL,
  areas_rand = NULL,
  opts = antaresRead::simOptions()
)
```

```r
readScenarioBuilder(
  ruleset = "Default Ruleset",
  as_matrix = TRUE,
  opts = antaresRead::simOptions()
)
```

```r
updateScenarioBuilder(
  ldata,
  ruleset = "Default Ruleset",
  series = NULL,
  clusters_areas = NULL,
  opts = antaresRead::simOptions()
)
```

```r
clearScenarioBuilder(
  ruleset = "Default Ruleset",
  opts = antaresRead::simOptions()
)
```
## Arguments

- **n_scenario**: Number of scenario.
- **n_mc**: Number of Monte-Carlo years.
- **areas**: Areas to use in scenario builder, if NULL (default) all areas in Antares study are used.
- **areas_rand**: Areas for which to use "rand".
- **opts**: List of simulation parameters returned by the function `antaresRead::setSimulationPath()`.
- **ruleset**: Ruleset to read.
- **as_matrix**: If TRUE (default) return a matrix, else a list.
- **ldata**: A matrix obtained with `scenarioBuilder`, or a named list of matrices obtained with `scenarioBuilder`, names must be 'l', 'h', 'w', 's', 't' or 'r', depending on the series to update.
- **series**: Name(s) of the serie(s) to update if `ldata` is a single matrix.
- **clusters_areas**: A data.table with two columns area and cluster to identify area/cluster couple to update for thermal or renewable series. Default is to read clusters description and update all couples area/cluster.

## Value

- **scenarioBuilder**: a matrix
- **readScenarioBuilder**: a list of matrix or list according to `as_matrix` parameters.

## Examples

```r
## Not run:

library(antaresRead)
library(antaresEditObject)

# simulation path
setSimulationPath(
  path = "pat/to/simulation",
  simulation = "input"
)

# Create a scenario builder matrix
sbuilder <- scenarioBuilder(
  n_scenario = 51,
  n_mc = 2040,
  areas_rand = c("fr", "be")
)
sbuilder[, 1:6]
dim(sbuilder)

# Read previous scenario builder
# in a matrix format
```
prev_sb <- readScenarioBuilder()

# Update scenario builder
# for load serie
updateScenarioBuilder(ldata = sbuilder, series = "load")

# equivalent as
updateScenarioBuilder(ldata = list(l = sbuilder))

# update several series
# same input
sbuilder
updateScenarioBuilder(
  ldata = sbuilder,
  series = c("load", "hydro", "solar")
)

# different input
updateScenarioBuilder(ldata = list(
  l = load_sb,
  h = hydro_sb,
  s = solar_sb
))

## End(Not run)

---

### setPlaylist

*Set the playlist of an Antares Study*

#### Description

`set_playlist` is a function which modifies the input file of an ANTARES study and set the playlist in order to simulate only the MC years given in input.

#### Usage

```r
setPlaylist(playlist, weights = NULL, opts = antaresRead::simOptions())
```

#### Arguments

- **playlist** vector of MC years identifier to be simulated can be a list (V8 compatibility) but not recommended
- **weights** data.table, 2 columns: mcYears and weights. Only with after antares V8
- **opts** list of simulation parameters returned by the function antaresRead::setSimulationPath
**updateGeneralSettings**

**Value**

The function does not return anything. It is used to modify the input of an Antares study.

---

**setSolverPath**  
*Set path to Antares Solver*

**Description**

Set path to Antares Solver

**Usage**

```
setSolverPath(path)
```

**Arguments**

- `path` *(optional)* Path to the solver (e.g. `antares-6.0-solver.exe` in \bin directory where Antares is installed). If missing, a window opens and lets the user choose the directory of the simulation interactively.

**Examples**

```
## Not run:
setSolverPath(path = "C:/antares/bin/antares-6.0-solver.exe")

## End(Not run)
```

---

**updateGeneralSettings**  
*Update general parameters of an Antares study*

**Description**

Update general parameters of an Antares study.
Usage

updateGeneralSettings(
  mode = NULL,
  horizon = NULL,
  nbyears = NULL,
  simulation.start = NULL,
  simulation.end = NULL,
  january.1st = NULL,
  first.month.in.year = NULL,
  first.weekday = NULL,
  leapyear = NULL,
  year.by.year = NULL,
  derated = NULL,
  custom.ts.numbers = NULL,
  user.playlist = NULL,
  filtering = NULL,
  active.rules.scenario = NULL,
  generate = NULL,
  nbtimeseriesload = NULL,
  nbtimeserieshydro = NULL,
  nbtimeserieswind = NULL,
  nbtimeseriesthermal = NULL,
  nbtimeseriessolar = NULL,
  refreshtimeseries = NULL,
  intra.modal = NULL,
  inter.modal = NULL,
  refreshintervalload = NULL,
  refreshintervalhydro = NULL,
  refreshintervalwind = NULL,
  refreshintervalthermal = NULL,
  refreshintervalsolar = NULL,
  readonly = NULL,
  opts = antaresRead::simOptions()
)

Arguments

mode Economy, Adequacy, Draft.

horizon Reference year (static tag, not used in the calculations)

nbyears Number of Monte-Carlo years that should be prepared for the simulation (not always the same as the Number of MC years actually simulated, see 'selection mode' below).

simulation.start First day of the simulation (e.g. 8 for a simulation beginning on the second week of the first month of the year)

simulation.end Last day of the simulation (e.g. 28 for a simulation ending on the fourth week of the first month of the year)
updateGeneralSettings

january.1st  First day of the year (Mon, Tue, etc.).
first.month.in.year  Actual month by which the Time-series begin (Jan to Dec, Oct to Sep, etc.)
first.weekday  In economy or adequacy simulations, indicates the frame (Mon- Sun, Sat-Fri, etc.) to use for the edition of weekly results.
leapyear  (TRUE/FALSE) indicates whether February has 28 or 29 days.
year.by.year  (False) No individual results will be printed out, (True) For each simulated year, detailed results will be printed out in an individual directory: Study_name/OUTPUT/simu_tag/Economy/mc-i-number
derated  See Antares General Reference Guide.
custom.ts.numbers  See Antares General Reference Guide.
user.playlist  See Antares General Reference Guide.
filtering  See Antares General Reference Guide.
active.rules.scenario  See Antares General Reference Guide.
generate  See Antares General Reference Guide.
nbtimeseriesload  See Antares General Reference Guide.
nbtimeserieshydro  See Antares General Reference Guide.
nbtimeserieswind  See Antares General Reference Guide.
nbtimeseriesthermal  See Antares General Reference Guide.
nbtimeseriessolar  See Antares General Reference Guide.
refreshtimeseries  See Antares General Reference Guide.
intra.modal  See Antares General Reference Guide.
inter.modal  See Antares General Reference Guide.
refreshintervalload  See Antares General Reference Guide.
refreshintervalhydro  See Antares General Reference Guide.
refreshintervalwind  See Antares General Reference Guide.
refreshintervalthermal  See Antares General Reference Guide.
refreshintervalthermal  See Antares General Reference Guide.
refreshintervalthermal  See Antares General Reference Guide.
readonly  See Antares General Reference Guide.
opts  List of simulation parameters returned by the function antaresRead::setSimulationPath()
updateInputSettings

Value

An updated list containing various information about the simulation.

Examples

```r
## Not run:
# Update number of Monte-Carlo years
updateGeneralSettings(nbyears = 42)

# Use a vector to update a parameter that
# can take multiple values
updateGeneralSettings(generate = c("thermal", "hydro"))

## End(Not run)
```

updateInputSettings

Update input parameters of an Antares study

Description

Update input parameters of an Antares study

Usage

```r
updateInputSettings(import, opts = antaresRead::simOptions())
```

Arguments

- **import**: Series to import.
- **opts**: List of simulation parameters returned by the function `antaresRead::setSimulationPath()`

Value

An updated list containing various information about the simulation.

Examples

```r
## Not run:
updateInputSettings(import = c("thermal"))
updateInputSettings(import = c("hydro", "thermal"))

## End(Not run)
```
updateOptimizationSettings

Update optimization parameters of an Antares study

Description

Update optimization parameters of an Antares study

Usage

updateOptimizationSettings(
    simplex.range = NULL,
    transmission.capacities = NULL,
    include.constraints = NULL,
    include.hurdlecosts = NULL,
    include.tc.min.stable.power = NULL,
    include.tc.min.up.down.time = NULL,
    include.dayahead = NULL,
    include.strategicreserve = NULL,
    include.spinningreserve = NULL,
    include.primaryreserve = NULL,
    include.exportmps = NULL,
    power.fluctuations = NULL,
    shedding.strategy = NULL,
    shedding.policy = NULL,
    unit.commitment.mode = NULL,
    number.of.cores.mode = NULL,
    renewable.generation.modelling = NULL,
    day.ahead.reserve.management = NULL,
    opts = antaresRead::simOptions()
)

Arguments

simplex.range       week or day
transmission.capacities
                      true, false or infinite
include.constraints
                      true or false
include.hurdlecosts
                      true or false
include.tc.min.stable.power
                      true or false
include.tc.min.up.down.time
                      true or false
include.dayahead
                      true or false
**updateOutputSettings**

```
include.strategicreserve  
true or false
include.spinningreserve  
true or false
include.primaryreserve   
true or false
include.exportmps        
true or false
power.fluctuations       
free modulations, minimize excursions or minimize ramping
shedding.strategy        
share margins
shedding.policy          
shave peaks or minimize duration
unit.commitment.mode     
fast or accurate
number.of.cores.mode     
minimum, low, medium, high or maximum
renewable.generation.modelling  
aggregated or clusters
day.ahead.reserve.management   
global
opts                     
List of simulation parameters returned by the function antaresRead::setSimulationPath
```

**Value**

An updated list containing various information about the simulation options.

---

**updateOutputSettings**  *Update output parameters of an Antares study*

**Description**

Update output parameters of an Antares study.

**Usage**

```
updateOutputSettings(  
synthesis = NULL,  
storenewset = NULL,  
archives = NULL,  
opts = antaresRead::simOptions()  
)
```
Arguments

- **synthesis** Logical. If TRUE, synthetic results will be stored in a directory Study_name/OUTPUT/simu_tag/Economy/sim-all. If FALSE, No general synthesis will be printed out.
- **storenewset** Logical. See Antares General Reference Guide.
- **archives** Character vector. Series to archive.
- **opts** List of simulation parameters returned by the function antaresRead::setSimulationPath

Value

An updated list containing various information about the simulation.

Examples

```r
## Not run:

updateOutputSettings(synthesis = TRUE, storenewset = FALSE,
                      archives = c("load", "wind"))

## End(Not run)
```

writeEconomicOptions Write Economic Options

Description

This function allows to create or edit economic options. Areas/options present in the input dataframe are edited, while all other values are left unchanged.

Usage

```r
writeEconomicOptions(x, opts = antaresRead::simOptions())
```

Arguments

- **x** A dataframe. Must contain an area column listing some (but not necessarily all) areas of the study. Can contain up to 7 other columns among: average_unsupplied_energy_cost, spread_unsupplied_energy_cost, average_spilled_energy_cost, spread_spilled_energy_cost, non_dispatchable_power, dispatchable_hydro_power and other_dispatchable_power (logical columns).
- **opts** List of simulation parameters returned by the function antaresRead::setSimulationPath
Examples

```r
## Not run:

library(antaresRead)

# Set simulation path
setSimulationPath(path = "PATH/TO/SIMULATION", simulation = "input")

# Write some economic options for areas a, b and c
writeEconomicOptions(data.frame(
  area = c("a", "b", "c"),
  dispatchable_hydro_power = c(TRUE, FALSE, FALSE),
  spread_unsupplied_energy_cost = c(0.03, 0.024, 0.01),
  average_spilled_energy_cost = c(10, 8, 8),
  stringsAsFactors = FALSE
))

## End(Not run)
```

writeIni

Write ini file from list obtain by antaresRead:::readIniFile and modify by user

Usage

```r
writeIni(listData, pathIni, overwrite = FALSE)
```

Arguments

- `listData` : list, modified list obtained by antaresRead:::readIniFile.
- `pathIni` : Character, Path to ini file.
- `overwrite` : logical, should file be overwritten if already exist?

Examples

```r
## Not run:

pathIni <- "D:/exemple_test/settings/generaldata.ini"

generalSetting <- antaresRead:::readIniFile(pathIni)
generalSetting$output$synthesis <- FALSE

writeIni(generalSetting, pathIni)

## End(Not run)
```
**writeOutputValues**

### Description

This function writes output values for an Antares study.

### Usage

```r
writeOutputValues(data, opts = NULL)
```

### Arguments

- `data`: A matrix of output values.
- `opts`: A list of simulation parameters.

### Examples

```r
## Not run:
writeOutputValues(cbind(1:3, 4:6), opts = antaresRead::simOptions())
## End(Not run)
```
Arguments

- `data`: obtain with `readAntares`.
- `opts`: List of simulation parameters returned by the function `antaresRead::setSimulationPath`.

Examples

```r
## Not run:

library(antaresRead)
library(data.table)
opts <- setSimulationPath("my_study")
data <- readAntares(links = "all", areas = "all", clusters = "all")
writeOutputValues(data)

## End(Not run)
```

writeSeriesPrepro  Write prepro data

Description

This function allows to write load, wind and solar prepro data. Using `character(0)` allows to erase data (cf Examples).

Usage

```r
writeSeriesPrepro(
  area,
  type = c("load", "wind", "solar"),
  coefficients = NULL,
  daily_profile = NULL,
  translation = NULL,
  conversion = NULL,
  overwrite = TRUE,
  opts = antaresRead::simOptions()
)
```

Arguments

- `area`: The area where to write prepro data.
- `type`: Type of data to write: "load", "wind" or "solar".
- `coefficients`: A 12*6 matrix of monthly values for the primary parameters alpha, beta, gamma, delta, theta and mu.
**writeWaterValues**

A 24*12 matrix of hourly / monthly coefficients K(hm) that are used to modulate the values of the stationary stochastic process by which the actual process is approximated.

A vector of length 8760 (or 8760*1 matrix) to add to the time-series generated, prior or after scaling.

A 2*N matrix (with 1<=N<=50) that is used to turn the initial time-series produced by the generators into final data. See Antares General Reference Guide.

Logical. Overwrite the values if a file already exists.

List of simulation parameters returned by the function `antaresRead::setSimulationPath`.

### Examples

```r
## Not run:
writeSeriesPrepro("fictive_area", type = "solar", daily_profile = matrix(rep(1, 24*12), nrow = 24))

# Erase daily profile data:
writeSeriesPrepro("fictive_area", type = "solar", daily_profile = character(0))

## End(Not run)
```

---

### Description

Write water values

### Usage

```r
writeWaterValues(
  area,
  data = NULL,
  overwrite = TRUE,
  opts = antaresRead::simOptions()
)
```

### Arguments

- **area**
  - The area where to add the water values.
- **data**
  - A 365x101 numeric matrix: table of marginal values for the stored energy, which depends on the date (365 days) and on the reservoir level (101 round percentage values ranging from 0% to 100%). OR a 3-column matrix with 365x101 rows. In this latter case the 3 columns must be 'date', 'level' and 'value' (in this order), and the rows must be sorted by: ascending day, ascending level.
- **overwrite**
  - Logical. Overwrite the values if a file already exists.
- **opts**
  - List of simulation parameters returned by the function `antaresRead::setSimulationPath`.
Examples

## Not run:

```r
writeWaterValues("fictive_area", data = matrix(rep(0, 365*101), nrow = 365))
```

## End(Not run)
Index

activateRES, 3
antaresRead::setSimulationPath, 31
antaresRead::setSimulationPath(), 8, 19, 21, 36, 40, 41, 48
backupStudy, 3
checkRemovedArea, 4
clearScenarioBuilder,(scenario-builder), 35
computeTimeStampFromHourly, 5
copyOutput, 6
create-cluster, 6
createArea, 10
createBindingConstraint, 11
createCluster (create-cluster), 6
createCluster(), 21, 31
createClusterRES (create-cluster), 6
createClusterRES(), 21, 31
createDistrict, 12
createDSR, 13
createLink, 15
createPSP, 17
createStudy, 19
dicoGeneralSettings, 19
dicoOptimizationSettings, 20
edit-cluster, 20
editArea, 22
editBindingConstraint, 23
editCluster (edit-cluster), 20
editCluster(), 8, 31
editClusterRES (edit-cluster), 20
editClusterRES(), 8, 31
editDSR (createDSR), 13
editLink, 24
editPSP (createPSP), 17
filteringOptions, 10, 22, 26
getCapacityDSR (createDSR), 13
getCapacityPSP (createPSP), 17
getPlaylist, 27
is_antares_v7, 27
nodalOptimizationOptions, 10, 22, 28
propertiesLinkOptions, 15, 29
readIniFile, 30
readScenarioBuilder (scenario-builder), 35
remove-cluster, 30
removeArea, 31
removeBindingConstraint, 32
removeCluster (remove-cluster), 30
removeCluster(), 8, 21
removeClusterRES (remove-cluster), 30
removeClusterRES(), 8, 21
removeLink, 33
runSimulation, 33
runTsGenerator, 34
scenario-builder, 35
scenarioBuilder (scenario-builder), 35
setPlaylist, 37
setSolverPath, 38
updateGeneralSettings, 38
updateInputSettings, 41
updateOptimizationSettings, 42
updateOutputSettings, 43
updateScenarioBuilder (scenario-builder), 35
writeEconomicOptions, 44
writeIni, 45
writeInputTS, 46
writeOutputValues, 46
writeSeriesPrepro, 47
writeWaterValues, 48