Package 'rjd3workspace'

April 2, 2026

Type Package

Title Wrangling 'JDemetra+ 3.x' Workspace

Version 3.7.1

Description R Interface to 'JDemetra+ 3.x'(<https://github.com/jdemetra>). It offers several functions to manipulate 'JDemetra+' workspaces, which can be read by the software and can store several seasonal adjusted series along with user-defined calendars or regression variables.

License EUPL

URL https://github.com/rjdverse/rjd3workspace, https://rjdverse.github.io/rjd3workspace/

BugReports https://github.com/rjdverse/rjd3workspace/issues

Depends R (>= 4.1.0)

Imports rJava (>= 1.0-6), rjd3providers (>= 3.7.1), rjd3toolkit (>= 3.7.1), rjd3tramoseats (>= 3.7.1), rjd3x13 (>= 3.7.1), tools, utils, methods

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

SystemRequirements Java (>= 21)

Collate 'deprecated.R' 'rjd3workspace-package.R' 'utils.R' 'saitem.R' 'saprocessing.R' 'update_path.R' 'workspace.R' 'zzz.R'

NeedsCompilation no

Author Jean Palate [aut], Alain Quartier-la-Tente [aut] (ORCID: <https://orcid.org/0000-0001-7890-3857>), Tanguy Barthelemy [aut, cre, art], Anna Smyk [aut]

Maintainer Tanguy Barthelemy <tanguy.barthelemy@insee.fr>

Repository CRAN

Date/Publication 2026-04-02 20:30:02 UTC

Contents

.jd2r_spec *Converts a jspec to a spec*

Description

Converts a jspec to a spec

Usage

```
.jd2r_spec(jspec)
```

Arguments

jspec	Specification in java format

Value

Specification in R format

add_calendar *Add a Calendar to a Workspace*

Description

Add a Calendar to a Workspace

Usage

```
add_calendar(jws, name, calendar)
```

Arguments

jws	a java workspace object.
name	character name of the calendar to add.
calendar	JDemetra+ calendar to add.

Value

NULL returned invisibly

Examples

```
# French calendar
french_calendar <- rjd3toolkit::national_calendar(
    days = list(
        rjd3toolkit::fixed_day(7, 14), # Bastille Day
      rjd3toolkit::fixed_day(5, 8, validity = list(start = "1982-05-08")), # End of 2nd WW
        rjd3toolkit::special_day("NEWYEAR"),
        rjd3toolkit::special_day("CHRISTMAS"),
        rjd3toolkit::special_day("MAYDAY"),
        rjd3toolkit::special_day("EASTERMONDAY"),
        rjd3toolkit::special_day("ASCENSION"),
        rjd3toolkit::special_day("WHITMONDAY"),
        rjd3toolkit::special_day("ASSUMPTION"),
        rjd3toolkit::special_day("ALLSAINTSDAY"),
        rjd3toolkit::special_day("ARMISTICE")
    )
)
# Load a Workspace
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")

jws <- jws_open(file)

# Add calendar to the workspace
add_calendar(jws, "French Calendar", french_calendar)
get_context(jws) # The workspace already contained a Test Calendar
```

`add_sa_item` *Add a SA-item to a SAProcessing*

Description

Add a SA-item to a SAProcessing

Usage

```
add_sa_item(jsap, name, x, spec)

## S3 method for class 'ts'
add_sa_item(jsap, name, x, spec)

## Default S3 method:
add_sa_item(jsap, name, x, spec)

## S3 method for class 'jobjRef'
add_sa_item(jsap, name, x, spec)
```

Arguments

jsap	SAProcessing.
name	name of the SA-item to be added.
x	either a seasonal adjustment model (from rjd3x13::x13() or rjd3tramoseats::tramoseats()), a SA-item object, "ts" object.
spec	specification to use when x is a "ts" object.

Value

NULL returned invisibly

Examples

```
dir <- tempdir()

# Raw series
y <- rjd3toolkit::ABS$X0.2.09.10.M

# Creating an empty workspace and SAProcessing
jws <- jws_new()
jsap1 <- jws_sap_new(jws, "sap1")

# Adding SA-item as estimation result


# Estimation with rjd3x13
add_sa_item(jsap1, name = "series_1", x = rjd3x13::x13(y))

# Estimation with rjd3tramoseats
add_sa_item(jsap1, name = "series_2", x = rjd3tramoseats::tramoseats(y))

# Adding SA-item as raw series + specification
add_sa_item(jsap1, name = "series_3", x = y, rjd3x13::x13_spec("RSA3"))
add_sa_item(jsap1, name = "series_4", x = y, rjd3tramoseats::tramoseats_spec("RSAFull"))

jsai1 <- jsap_sai(jsap = jsap1, idx = 1L)
# Adding SA-item from a Workspace
add_sa_item(jsap = jsap1, name = "series_1_bis", x = jsai1)

rws <- read_workspace(jws)
rws$processing$sap1$series_4

# Writing the workspace
save_workspace(jws, file.path(dir, "workspace.xml"))
```

add_variables *Add a Variable to a JD+ Workspace*

Description

Adds a single time series variable to a specified group within a JD+ workspace..

Usage

```
add_variables(jws, group, name, y, overwrite = FALSE)
```

Arguments

jws	A JD+ workspace object (Java pointer).
group	A character string indicating the name of the group in which to store the variable.
name	A character string naming the variable.
y	A ts object (R time series) to be added. Only a single time series can be added at a time.
overwrite	a Boolean to indicate whether a variable already present should be replaced

Details

For the time being, if the group does not already exist, a new group is created, but the group will be named after name, not group.

Value

No return value (NULL returned invisibly). This function is used for its side effect of modifying the workspace.

Limitations

- Cannot add multiple variables at once.
- Does not support dynamic ts objects with metadata.
- If group does not exist, a new group is created but named after the variable name, not the intended group.

See Also

rjd3toolkit::modelling_context() to create multiple variables and groups at once, and read_variables(), write_variables() to import/export variables.

Examples

```
# Load a Workspace
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")

jws <- jws_open(file)
add_variables(jws = jws, group = "reg1", y = AirPassengers, name = "x1")
```

---

deprecated-rjd3workspace

Deprecated functions

Description

Deprecated functions

Usage

```
.jmp_sa_count(jmp)

.jsap_sa_count(jmp)

.jsap_sai_count(jsap)

.jmp_name(jmp)

.jsap_name(jsap)

.jmp_sa(jmp, idx)

.jsap_sa(jsap, idx)

.jsap_sai(jsap, idx)

.jmp_sa_name(jmp)

.jsap_sa_name(jsap)

.jsap_sai_names(jsap)

.jmp_load(jmp)

.jsa_read(jsa)

.jsa_results(jsa, items = NULL)
```

```
.jsa_jresults(jsa)

.jsa_metadata(jsa, key)

.jsai_metadata(jsai, key)

.jsa_ts_metadata(jsa, key)

.jsai_ts_metadata(jsa, key)

.jws_sap_count(jws)

.jws_open(file)

.jread_workspace(jws, compute = TRUE)

.jread_sap(jsap)

.jws_new(modelling_context = NULL)

.jws_sap_new(jws, name)

.jws_make_copy(jws)

.jsap_make_copy(jsap)

.jws_compute(jws)

.jws_sap(jws, idx)

.jsai_name(jsai)

.jsap_refresh(
  jsap,
  policy = c("FreeParameters", "Complete", "Outliers_StochasticComponent", "Outliers",
    "FixedParameters", "FixedAutoRegressiveParameters", "Fixed"),
  period = 0,
  start = NULL,
  end = NULL,
  info = c("All", "Data", "None")
)

.jws_refresh(
  jws,
  policy = c("FreeParameters", "Complete", "Outliers_StochasticComponent", "Outliers",
    "FixedParameters", "FixedAutoRegressiveParameters", "Fixed"),
  period = 0,
```

```
  start = NULL,
  end = NULL,
  info = c("All", "Data", "None")
)

transfer_series(
  jsap_from,
  jsap_to,
  selected_sa_items,
  print_indications = TRUE
)

.jws_add(jws, jsap)
```

Arguments

`jmp`, `idx`, `jws`, `name`, `jsa`, `jsai`, `jsap`, `items`, `key`, `file`, `compute`, `policy`, `period`, `start`, `end`, `info`, `modelling_context`, `jsap_from`, `jsap_to`, `selected_sa_items`, `print_indications`
: Parameters.

Value

The same value as returned by the corresponding non-deprecated function. The returned object represents an encoded identifier for a spreadsheet series or collection.

`get-results` *Extract results from a SA-item*

Description

`get_results()` extracts the results of a SA-item. `.jsai_results()` extracts specific output of the model of the SA-item. `.jsai_jresults()` extracts the Java object of the results of a SA-item.

Usage

```
get_results(jsai)

.jsai_results(jsai, items = NULL)

.jsai_jresults(jsai)
```

Arguments

`jsai`	Java SA-item object.
`items`	vector of characters containing the variables to extract. See `rjd3x13::x13_dictionary()` or `rjd3tramoseats::tramoseats_dictionary()`. By default, extracts all the possible variables.

Value

List with all the results of the adjustment.

`get-specification` *Get Specification in a Sa-Item*

Description

`get_estimation_specification()` extract the estimation specification, `get_domain_specification()` the domain specification, , `get_active_specification()` the active specification `get_point_specification()` the point specification

Usage

```
get_domain_specification(jsai)

get_estimation_specification(jsai)

get_point_specification(jsai)

get_active_specification(jsai)
```

Arguments

`jsai`	Java SA-item object.

Value

the specification

Examples

```
# Load a Workspace to modify
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")

jws <- jws_open(file)

# Select SAProcessing with the target SA-item
jsap1 <- jws_sap(jws, 1)
jsai1 <- jsap_sai(jsap1, 1)

# Get the active specification in targeted SA-item
get_active_specification(jsai1)

# Get the domain specification in targeted SA-item
get_domain_specification(jsai1)
```

```
# Get the estimation specification in targeted SA-item
get_estimation_specification(jsai1)

# Get the point specification in targeted SA-item
get_point_specification(jsai1)
```

`get_context` *Get Context from Workspace*

Description

Get Context from Workspace

Usage

```
get_context(jws)
```

Arguments

`jws`	the Workspace.

Value

The modelling context (list object with Calendars and Variables).

Examples

```
# Load a Workspace
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")

jws <- jws_open(file)

# Get context
my_context <- get_context(jws)
```

get_metadata *Extract Metadata from a SA-Item*

Description

Extract specific metadata or time series metadata of a SA-item.

Usage

```
get_metadata(jsai, key)

get_ts_metadata(jsai, key)
```

Arguments

jsai	Java SA-item object.
key	key of the metadata.

Value

The corresponding metadata (character, numeric...)

Examples

```
# Load a Workspace
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")

jws <- jws_open(file)

# Select SAProcessing
jsap1 <- jws_sap(jws, 1)

# Select SA-item (as java object)
jsai1 <- jsap_sai(jsap1, 3)

# Extract the comment as metadata
get_metadata(jsai1, "comment")

# Extract the ts metadata
get_metadata(jsai1, "@id")
get_metadata(jsai1, "@source")
get_metadata(jsai1, "@timestamp")
```

jsap_make_copy *Copy a Workspace or SA-Processing*

Description

Copy a Workspace or SA-Processing

Usage

```
jsap_make_copy(jsap)

jws_make_copy(jws)
```

Arguments

jws, jsap	Java Workspace or SA-Processing

Details

The copy of a SA-processing will be made in the same workspace. The modelling context of the workspace is also copied.

Value

Returns a java object workspace or SA-Processing

References

More information on workspaces in JDemetra+ Graphical User Interface: https://jdemetra-new-documentation.netlify.app/t-gui-sa-modelling-features/

See Also

read_workspace(), read_sap()

Examples

```
# Create an empty 'JDemetra+' Workspace
jws <- jws_new()
# Add an empty SA-Processing
jsap <- jws_sap_new(jws, "sap1")
# Make a copy of the workspace
jws2 <- jws_make_copy(jws)
# Make a copy of sap1 in jws2
jsap2 <- jsap_make_copy(jsap)
```

jsap_refresh *Refresh a Workspace or SA-Processing*

Description

Refresh a Workspace or SA-Processing

Usage

```
jsap_refresh(
  jsap,
 policy = c("FreeParameters", "Complete", "Outliers_StochasticComponent", "Outliers",
    "FixedParameters", "FixedAutoRegressiveParameters", "Fixed"),
  period = 0,
  start = NULL,
  end = NULL,
  info = c("All", "Data", "None")
)

jws_refresh(
  jws,
 policy = c("FreeParameters", "Complete", "Outliers_StochasticComponent", "Outliers",
    "FixedParameters", "FixedAutoRegressiveParameters", "Fixed"),
  period = 0,
  start = NULL,
  end = NULL,
  info = c("All", "Data", "None")
)
```

Arguments

policy	refresh policy to apply (see details).
period, start, end	to specify the span on which outliers will not be re-identified (i.e.: re-detected) when policy = "Outliers" or policy = "Outliers_StochasticComponent". Span definition: period: numeric, number of observations in a year (12, 4...). start and end: first and last date from which outliers will not be re-identified, defined as arrays of two elements: year and first period (for example, if period = 12, c(1980, 1) for January 1980). If they are not specified, the outliers will be re-identified on the whole series.
info	information to refresh.
jws, jsap	Java Workspace or SA-Processing

Details

Available refresh policies are:

Current: applying the current pre-adjustment reg-arima model and adding the new raw data points as Additive Outliers (defined as new intervention variables)

Fixed: applying the current pre-adjustment reg-arima model and replacing forecasts by new raw data points.

FixedParameters: pre-adjustment reg-arima model is partially modified: regression coefficients will be re-estimated but regression variables, Arima orders and coefficients are unchanged.

FixedAutoRegressiveParameters: same as FixedParameters but Arima Moving Average coefficients (MA) are also re-estimated, Auto-regressive (AR) coefficients are kept fixed.

FreeParameters: all regression and Arima model coefficients are re-estimated, regression variables and Arima orders are kept fixed.

Outliers: regression variables and Arima orders are kept fixed, but outliers will be re-detected on the defined span, thus all regression and Arima model coefficients are re-estimated

Outliers_StochasticComponent: same as "Outliers" but Arima model orders (p,d,q)(P,D,Q) can also be re-identified.

Value

The refreshed element.

`jsap_sai` *Extract a SA-Processing or a SA-Item*

Description

Functions allowing to extract a SA-Processing from a Workspace using its order number (index) and a SA-Item from a SA-Processing its order number (index). The original object is unaltered.

Usage

```
jsap_sai(jsap, idx)

jws_sap(jws, idx)
```

Arguments

`idx`	index of the object to extract.
`jws`, `jsap`	Workspace or SA-Processing.

Value

Returns a java object SA-Processing or SA-Item.

Examples

```
# Load a Workspace
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")

jws <- jws_open(file)

# Compute the workspace to enable accessing its components
jws_compute(jws)

# Extract 2nd SA-Processing
jsap2 <- jws_sap(jws, 2)

# Extract 3rd SA-item
jsai3 <- jsap_sai(jsap2, 3)
```

---

jws_add *Add a SA-Processing to a Workspace*

Description

Add a SA-Processing to a Workspace

Usage

```
jws_add(jws, jsap)
```

Arguments

jws, jsap	Java Workspace or SA-Processing

Value

Invisibly NULL

---

jws_compute *Compute a Workspace*

Description

jws_compute() allows to extract all the SA-Items as java object.

Usage

```
jws_compute(jws)
```

Arguments

jws	a workspace

Value

Invisibly NULL

Examples

```
# Load a Workspace
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")

jws <- jws_open(file)

# Compute the workspace to access its components
jws_compute(jws)
```

---

jws_new *Create a Workspace or SA-Processing*

Description

Functions creating a 'JDemetra+' Workspace (jws_new()) and adding a new SA-Processing (jws_sap_new()). A modelling context can be added to a workspace, it will be valid for all its SA-Processings.

Usage

```
jws_new(modelling_context = NULL)

jws_sap_new(jws, name)
```

Arguments

modelling_context	a list of variables and calendars
jws	a java workspace object.
name	name of the new SA-Processing to be added (character).

Details

A modelling context is a list of variables to be used as external regressors in modelling processes (Reg-Arima or Tramo) or calendars to be used to generate calendar regressors. It can be created with rjd3toolkit::modelling_context() function or retrieved from another workspace ((set_context))

Value

Returns a java object workspace or SA-Processing.

References

More information on workspaces in JDemetra+ Graphical User Interface: https://jdemetra-new-documentation.netlify.app/t-gui-sa-modelling-features/

See Also

read_workspace(), read_sap()

Examples

```
# Create an empty 'JDemetra+' Workspace
jws <- jws_new()
# Add an empty SA-Processing
jsap <- jws_sap_new(jws, "sap1")
```

---

jws_open *Open an existing 'JDemetra+' Workspace*

Description

jws_open() opens an existing Workspace (as a Java pointer) and jws_compute() computes it (allowing to extract all the SA-Items as java objects).

Usage

```
jws_open(file)
```

Arguments

file	path to Workspace xml master file By default a dialog box opens.

Value

a java workspace

See Also

read_workspace() to transform the workspace in a R list.

Examples

```
# Load a Workspace
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")

jws <- jws_open(file)

# Compute the workspace to enable access its components
jws_compute(jws)
```

`read_calendars` *Read a Calendar file*

Description

The calendar file is a xml file like the one JDemetra+ would write when defining a calendar in the Graphical User Interface.

Usage

```
read_calendars(file)
```

Arguments

`file`	path to a calendar file (in xml format)

Value

a list of `JD3_CALENDAR` objects

Examples

```
file <- system.file("workspaces", "workspace_test", "Calendars",
                    "Calendars.xml", package = "rjd3workspace")
my_calendar <- read_calendars(file)
my_calendar
```

read_sai *Read an SA-item*

Description

read_sai() extracts all the information of a SA-item (see details).

Usage

```
read_sai(jsai)
```

Arguments

jsai	Java SA-item object.

Details

A SA-item contains more information than just the results of an estimation. Full information is extracted with the read_sai() function that returns a list of 5 objects:

- ts: raw time series.
- domainSpec: initial specification. Reference when refreshing and relaxing constraints.
- estimationSpec: specification used for the current estimation.
- pointSpec: specification corresponding to the results of the current estimation (fully identified model).
- results: results of the estimation.

Value

a list

Examples

```
# Load a Workspace
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")

jws <- jws_open(file)

# Select SAProcessing
jsap1 <- jws_sap(jws, 1)

# Select SA-item (as java object)
jsai1 <- jsap_sai(jsap1, 3)
```

read_sap *Read all SA-Items from a Workspace or SA-Processing*

Description

Functions reading all SA-Items from a Workspace (`read_workspace()`) or a SA-Processing (`read_sap()`) and allowing to access them as R lists. Whereas functions `jread_sap()` and `jread_workspace()` only return corresponding Java objects

Usage

```
read_sap(jsap)

jread_sap(jsap)

read_workspace(jws, compute = TRUE)

jread_workspace(jws, compute = TRUE)
```

Arguments

jsap	java SA-Processing.
jws	java Workspace.
compute	compute or not the workspace (to get the estimation results).

Value

list or java object

Examples

```
# Load workspace
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")

jws <- jws_open(file)

# Read workspace
jread_workspace(jws, compute = FALSE)

rws <- read_workspace(jws)

# Read sap
sap <- jws_sap(jws,1)
jread_sap(sap)
read_sap(sap)
```

read_variables *Read auxiliary regressors file*

Description

The variables (regressors) file is a xml file like the one JDemetra+ would write when setting-up user defined regressors in the Graphical User Interface.

Usage

```
read_variables(file)
```

Arguments

file	xml format

Value

A named list of time series objects.

Examples

```
file <- system.file("workspaces", "workspace_test", "Variables",
                    "Vars-1.xml", package = "rjd3workspace")
my_regressors <- read_variables(file)
class(my_regressors)
str(my_regressors)
```

regarima_read_spec *Read a Reg-Arima specification file*

Description

The specification file is a xml file like the one JDemetra+ would write when defining a specification in the Graphical User Interface.

Usage

```
regarima_read_spec(file)
```

Arguments

file	xml format,

Value

list

Examples

```
file <- system.file("workspaces", "workspace_test", "RegArimaSpec",
                    "RegArimaSpec-1.xml", package = "rjd3workspace")
my_spec<-regarima_read_spec(file)
class(my_spec)
str(my_spec)
```

---

regarima_write_spec *Write a Reg-Arima specification file*

Description

The specification file is a xml file like the one JDemetra+ would write when defining a specification in the Graphical User Interface.

Usage

```
regarima_write_spec(spec, file)
```

Arguments

spec	a specification created with `rjd3x13::regarima_spec`
file	xml format

Value

NULL returned invisibly

Examples

```
# Creating a spec from default
regarima_spec <- rjd3x13::regarima_spec("rg3")

# Forcing multiplicative model
regarima_spec_d <- rjd3toolkit::set_transform(
    regarima_spec ,
    fun = "Log",
    outliers = TRUE
)

# Writing the specification in a xml file
spec_path <- tempfile(fileext = ".xml")
regarima_write_spec(regarima_spec_d, file = spec_path)
```

replace_sa_item *Replace or Remove a SA-item*

Description

replace_sa_item() replaces a SA-item in a SAProcessing and remove_sa_item() removes a SA-item from a SAProcessing remove_all_sa_item() removes all SA-item from a SAProcessing

Usage

```
replace_sa_item(jsap, idx, jsai)

remove_sa_item(jsap, idx)

remove_all_sa_item(jsap)
```

Arguments

jsap	SAProcessing to be modified.
idx	index of the target SA-item.
jsai	new SA-item (for replacement).

Value

NULL returned invisibly

sai_name *Get the name of a SAProcessing or one (or all) Sa-item*

Description

Functions to retrieve the name of a SAProcessing (sap_name()) or Sa-item (sai_name()) or all SA-item (sap_sai_names()) .

Usage

```
sai_name(jsai)

sap_name(jsap)

sap_sai_names(jsap)
```

Arguments

jsap, jsai	the object to retrieve the name from.

Value

A vector character.

Examples

```
# Load a Workspace
file <- system.file("workspaces", "workspace_test.xml",
                    package = "rjd3workspace")

jws <- jws_open(file)

# Extract 2nd SA-Processing
jsap_2 <- jws_sap(jws, 2)

# Retrieve the name
sap_name(jsap_2)

# Retrieve all the SA-items names
sap_sai_names(jsap_2)
```

---

sap_sai_count *Count SA-Processings or SA-Items*

Description

Functions counting the SA-Processings in a Workspace (ws_sap_count) or the SA-Items in a SA-Processing (sap_sai_count).

Usage

```
sap_sai_count(jsap)

ws_sap_count(jws)
```

Arguments

jws, jsap	Workspace or SA-Processing.

Value

Returns an integer.

Examples

```
# Load a Workspace
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")

jws <- jws_open(file)

# Count the SA-Processings
ws_sap_count(jws)

# Count the SA-Items
# In SAP 1
sap1 <- jws_sap(jws,1)
sap_sai_count(sap1)
```

`save_workspace` *Save Workspace*

Description

Function allowing to write a workspace as a collection of xml files readable by JDemetra+ Graphical User Interface.

Usage

```
save_workspace(jws, file, replace = FALSE)
```

Arguments

`jws`	Workspace object to export.
`file`	path where to export the 'JDemetra+' Workspace (.xml file).
`replace`	boolean indicating if the Workspace should be replaced if it already exists.

Value

A boolean indicating if the saving was successful.

Examples

```
dir <- tempdir()
jws <- jws_new()
jsap1 <- jws_sap_new(jws, "sap1")
y <- rjd3toolkit::ABS$X0.2.09.10.M


add_sa_item(jsap1, name = "serie_1", x = y, rjd3x13::x13_spec())
save_workspace(jws, file.path(dir, "workspace.xml"))
```

set_comment *Get/Set Comment from a SA-item*

Description

Get/Set Comment from a SA-item

Usage

```
set_comment(jsap, idx, comment)

get_comment(jsai)
```

Arguments

jsap	SAProcessing to be modified.
idx	index of the target SA-item.
comment	character containing the comment.
jsai	a SA-item.

Value

NULL returned invisibly

Examples

```
# Load a Workspace
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")

jws <- jws_open(file)

# Select SAProcessing
jsap1 <- jws_sap(jws, 1L)

# Add a comment
set_comment(jsap1, 2L, "data collection changed in 2012")

jsai2 <- jsap_sai(jsap1, 2L)
get_comment(jsai2)
```

set_context *Set Context of a Workspace*

Description

Set Context of a Workspace

Usage

```
set_context(jws, modelling_context = NULL)
```

Arguments

jws	a java workspace object.
modelling_context	a list of variables and calendars

Value

Invisibly NULL

Examples

```
library("rjd3toolkit")

# French calendar
french_calendar <- national_calendar(
    days = list(
        fixed_day(7, 14), # Bastille Day
        fixed_day(5, 8, validity = list(start = "1982-05-08")), # End of 2nd WW
        special_day("NEWYEAR"),
        special_day("CHRISTMAS"),
        special_day("MAYDAY"),
        special_day("EASTERMONDAY"),
        special_day("ASCENSION"),
        special_day("WHITMONDAY"),
        special_day("ASSUMPTION"),
        special_day("ALLSAINTSDAY"),
        special_day("ARMISTICE")
    )
)

# Load a Workspace
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")

jws <- jws_open(file)

# Creating a new context
```

```
new_context <- modelling_context(
    calendars = list(FR = french_calendar),
    variables = list(a = AirPassengers)
)

# Set the context
set_context(jws, new_context)
```

---

set_name *Set the name of a SA-item*

Description

Set the name of a SA-item

Usage

```
set_name(jsap, idx, name)
```

Arguments

jsap	SAProcessing to be modified.
idx	index of the target SA-item.
name	character corresponding to the new name

Value

NULL returned invisibly

See Also

sai_name()

Examples

```
# Load a Workspace
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")

jws <- jws_open(file)

# Select SAProcessing
sap1 <- jws_sap(jws,1)

# Select SA-item
sai1 <- jsap_sai(sap1,3) # java object sai

# set name
```

```
set_name(sap1,3,"RF1011_1")

# check
sai1 <- jsap_sai(sap1,3) # reload sai
sai_name(sai1) #get name
```

`set_priority` *Get/Set SA-item Priority*

Description

Get/Set SA-item Priority

Usage

```
set_priority(jsap, idx, priority = 0L)

get_priority(jsai)
```

Arguments

`jsap`	SAProcessing to be modified.
`idx`	index of the target SA-item.
`priority`	integer containing the priority.
`jsai`	a SA-item.

Value

`set_priority` returns `NULL` invisibly. `get_priority` returns the priority (an integer).

Examples

```
# Load a workspace
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")

my_jws <- jws_open(file)

# Select the first SA-Processing and SA-Item
jsap <- jws_sap(my_jws, 1)
jsai <- jsap_sai(jsap, 1L)

# Change priority
set_priority(jsap, idx = 1L, priority = 3L)

# Retrieve priority
```

```
get_priority(jsai)
```

---

set_raw_data *Get/Set Raw Data in a SA-item*

Description

Get/Set Raw Data in a SA-item

Usage

```
set_raw_data(jsap, idx, y)

get_raw_data(jsai)
```

Arguments

jsap	SAProcessing to be modified.
idx	index of the target SA-item.
y	new raw time series.
jsai	a SA-item.

Value

NULL returned invisibly (set) or TS object (get)

Examples

```
# Load a Workspace
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")

jws <- jws_open(file)

# Select SAProcessing
sap1 <- jws_sap(jws, 1)

# Select SA-item
sai1 <- jsap_sai(sap1, 3) # java object sai
tail(get_raw_data(sai1))

new_raw_data <- rjd3toolkit::ABS$X0.2.15.10.M
set_raw_data(sap1,3,new_raw_data)

sai1 <- jsap_sai(sap1,3) # reload SA-item
tail(get_raw_data(sai1)) # get raw data
```

set_specification *Set Specification in a Sa-Item*

Description

Set Specification in a Sa-Item

Usage

```
set_specification(jsap, idx, spec)

set_domain_specification(jsap, idx, spec)
```

Arguments

jsap	SAProcessing to be modified.
idx	index of the target SA-item.
spec	new specification generated with rjd3x13::x13_spec() or rjd3tramoseats::tramoseats_spec()

Value

NULL returned invisibly

Examples

```
# Create a (customized) spec) spec
library(rjd3x13)

spec <- rjd3x13::x13_spec("rsa3") |>
    rjd3toolkit::set_basic(type = "From", d0 = "2012-01-01")


# Load a Workspace to modify
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")
jws <- jws_open(file)

# Select SAProcessing with the target SA-item
sap1 <- jws_sap(jws, 1)

# Set specification in targeted SA-item
set_specification(sap1, 2, spec)

# Set domain specification in selected SA-item
set_domain_specification(sap1, 3, spec)
```

set_ts *Get/Set the (JDemetra+) time series of a SA-item*

Description

(JDemetra+) time series contains more information than raw data, which can be manipulated with `set_raw_data()` and `get_raw_data()`

Usage

```
set_ts(jsap, idx, y)

get_ts(jsai)
```

Arguments

jsap	SAProcessing to be modified.
idx	index of the target SA-item.
y	a "full" time series (jd3-like).
jsai	a SA-item.

Value

`NULL` returned invisibly

Examples

```
# Load a workspace
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")

my_jws <- jws_open(file)

library("rjd3providers")
data_path <- system.file("extdata", "IPI_nace4.csv", package = "rjd3workspace")

ts_object <- txt_series(
    file = data_path,
    series = 1L,
    delimiter = "SEMICOLON",
    fmt.date = "dd/MM/yyyy"
)

# Select the first SA-Processing
jsap <- jws_sap(my_jws, 1L)

# Change the ts object
set_ts(jsap = jsap, idx = 1L, ts_object)
```

```
jsai1 <- jsap_sai(jsap, 1L)
jsai2 <- jsap_sai(jsap, 2L)
jsai3 <- jsap_sai(jsap, 3L)

# Get the ts object
get_ts(jsai1)
get_ts(jsai2)
get_ts(jsai3)
```

`set_ts_metadata` *Set (JDemetra+) Metadata of a SA-item*

Description

Function to set the metadata of a SA-item.

`XXX_ts_metadata()` set the time series metadata of a SA-item (provider, source of the data...). `XXX_metadata()` set any metadata to a SA-Item.

`set_XXX()` uses the metadata of another SA-item while `put_XXX()` allows to update a specific key with a new information.

Usage

```
set_ts_metadata(jsap, idx, ref_jsai)

put_ts_metadata(jsap, idx, key, value)

set_metadata(jsap, ref_jsai, idx)

put_metadata(jsap, idx, key, value)
```

Arguments

`jsap`	SAProcessing to be modified.
`idx`	index of the target SA-item.
`ref_jsai`	a reference SA-item containing the metadata.
`key`	key of the metadata.
`value`	value of the metadata.

Value

`NULL` returned invisibly.

Examples

```
# Change the file of a given item
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")

jws <- jws_open(file)
jsap <- jws_sap(jws, 1)
jsai <- jsap_sai(jsap, 1)
nid <- rjd3providers::txt_change_file(get_ts_metadata(jsai, "@id"), "test.csv")
put_ts_metadata(jsap, 1, "@id", nid)

jsai <- jsap_sai(jsap, 1)
get_ts_metadata(jsai, "@id")
```

---

spreadsheet_update_path

Update the path to raw data in a workspace (spreadsheet)

Description

Update the path to raw data in a workspace (spreadsheet)

Usage

```
spreadsheet_update_path(jws, new_path, idx_sap = NULL, idx_sai = NULL)
```

Arguments

jws	workspace object
new_path	new path to the spreadsheet containing raw data
idx_sap	index (or indices) of the SAProcessing(s)
idx_sai	index (or indices) of the SA-item(s).

Details

The spreadsheet file must be a .xlsx file. .xls files are not accepted in JDemetra+ v3.x.

Value

This function returns either NULL if the update was successful, or an error.

Examples

```
# Load a workspace
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")

my_ws <- jws_open(file)

# Update the entire second SA-Processing of the `my_ws` workspace with a new path to raw data
spreadsheet_update_path(
    jws = my_ws,
    new_path = system.file("extdata", "IPI_nace4.xlsx", package = "rjd3workspace"),
    idx_sap = 2
)

# Select one (the 2nd) SA-item from second SA-Processing
sap2 <- jws_sap(my_ws, 2)
sai2 <- jsap_sai(sap2, 2)

# Check path
get_ts_metadata(sai2, "@id")
```

---

tramoseats_read_spec *Read a Tramo-Seats specification file*

Description

The specification file is a xml file like the one JDemetra+ would write when defining a specification in the Graphical User Interface.

Usage

```
tramoseats_read_spec(file)
```

Arguments

file	xml format,

Value

list

Examples

```
file <- system.file("workspaces", "workspace_test", "TramoSeatsSpec",
                    "TramoSeatsSpec-1.xml", package = "rjd3workspace")
my_spec<- tramoseats_read_spec(file)
class(my_spec)
str(my_spec)
```

tramoseats_write_spec *Write a Tramo-Seats specification file*

Description

The specification file is a xml file like the one JDemetra+ would write when defining a specification in the Graphical User Interface.

Usage

```
tramoseats_write_spec(spec, file)
```

Arguments

spec	a specification created with rjd3tramoseats::tramoseats_spec
file	xml format

Value

NULL returned invisibly

Examples

```
# Creating a spec from default
tramoseats_spec <- rjd3tramoseats::tramoseats_spec("tr3")

# Forcing multiplicative model
tramoseats_spec_d <- rjd3toolkit::set_transform(
    tramoseats_spec ,
    fun = "Log",
    outliers = TRUE
)

# Writing the specification in a xml file
spec_path <- tempfile(fileext = ".xml")
tramoseats_write_spec(tramoseats_spec_d, file = spec_path)
```

tramo_read_spec *Read a Tramo specification file*

Description

The specification file is a xml file like the one JDemetra+ would write when defining a specification in the Graphical User Interface.

Usage

```
tramo_read_spec(file)
```

Arguments

file	xml format,

Value

list

Examples

```
file <- system.file("workspaces", "workspace_test", "TramoSpec",
                    "TramoSpec-1.xml", package = "rjd3workspace")
my_spec<- tramo_read_spec(file)
class(my_spec)
str(my_spec)
```

---

tramo_write_spec *Write a Tramo specification file*

Description

The specification file is a xml file like the one JDemetra+ would write when defining a specification in the Graphical User Interface.

Usage

```
tramo_write_spec(spec, file)
```

Arguments

spec	a specification created with `rjd3tramoseats::tramo_spec`
file	xml format

Value

`NULL` returned invisibly

Examples

```
# Creating a spec from default
tramo_spec <- rjd3tramoseats::tramo_spec("tr3")

# Forcing multiplicative model
tramo_spec_d <- rjd3toolkit::set_transform(
    tramo_spec ,
    fun = "Log",
    outliers = TRUE
)

# Writing the specification in a xml file
spec_path <- tempfile(fileext = ".xml")
tramo_write_spec(tramo_spec_d, file = spec_path)
```

---

transfer_sa_item *Copy & paste SA-items from one* SA-Processing *to another*

Description

Copy & paste SA-items from one SA-Processing to another

Usage

```
transfer_sa_item(
  jsap_from,
  jsap_to,
  selected_sa_items,
  print_indications = TRUE
)
```

Arguments

jsap_from	SA-Processing from which to take the SA-items
jsap_to	SA-Processing to which paste the SA-items
selected_sa_items	vector containing the SA-items names to be updated.
print_indications	A boolean to print indications on the processing status (optional)

Details

If selected_sa_items is missing, all SA-items from jsap_from will be copied.

Value

NULL returned invisibly

`txt_update_path` *Update the path to raw data in a workspace (txt/csv file)*

Description

Update the path to raw data in a workspace (txt/csv file)

Usage

```
txt_update_path(jws, new_path, idx_sap = NULL, idx_sai = NULL)
```

Arguments

`jws`	workspace object
`new_path`	new path to the csv/txt file containing raw data.
`idx_sap`	index (or indices) of the SAProcessing(s)
`idx_sai`	index (or indices) of the SA-item(s).

Value

This function returns either NULL if the update was successful, or an error

Examples

```
# Load a workspace
file <- system.file("workspaces", "workspace_test.xml", package = "rjd3workspace")

my_ws <- jws_open(file)

# Update the entire second SA-Processing of the `my_ws` workspace with a new path to raw data
txt_update_path(
    jws = my_ws,
    new_path = system.file("extdata", "IPI_nace4.csv", package = "rjd3workspace"),
    idx_sap = 1
)

# Select one (the 2nd) SA-item from first SA-Processing
sap1 <- jws_sap(my_ws, 1)
sai2 <- jsap_sai(sap1, 2)

# Check path
get_ts_metadata(sai2, "@id")
```

write_calendars *Write a Calendar file*

Description

The calendar file is a xml file like the one JDemetra+ would write when defining a calendar in the Graphical User Interface. Calendars can be defined with `rjd3toolkit::national_calendar`

Usage

```
write_calendars(calendars, file)
```

Arguments

`calendars`	list of calendars or a `JD3_CALENDAR` object
`file`	xml format

Value

`NULL` returned invisibly

Examples

```
library("rjd3toolkit")
BE <- national_calendar(list(
    fixed_day(7, 21),
    special_day("NEWYEAR"),
    special_day("CHRISTMAS"),
    special_day("MAYDAY"),
    special_day("EASTERMONDAY"),
    special_day("ASCENSION"),
    special_day("WHITMONDAY"),
    special_day("ASSUMPTION"),
    special_day("ALLSAINTSDAY"),
    special_day("ARMISTICE")
))

calendar_path <- tempfile(pattern = "calendar", fileext = ".xml")

write_calendars(BE, file = calendar_path)
write_calendars(list(BEL_cal = BE), file = calendar_path)
```

write_variables *Write regressors file*

Description

Write regressors file

Usage

```
write_variables(vars, file)
```

Arguments

vars	A named list of ts objects.
file	Path to the output XML file.

Value

No return value (NULL returned invisibly). This function writes variables to file for use in JD+.

Examples

```
# Load a Workspace
file <- system.file("workspaces", "workspace_test.xml",
                    package = "rjd3workspace")

jws <- jws_open(file)

# Get context
my_context <- get_context(jws)
vars <- my_context$variables[[1L]]

# Writing the regressors in a xml file
variable_path <- tempfile(fileext = ".xml")
write_variables(vars, file = variable_path)
```

x13_read_spec *Read a X13 specification file*

Description

The specification file is a xml file like the one JDemetra+ would write when defining a specification in the Graphical User Interface.

Usage

```
x13_read_spec(file)
```

Arguments

file	xml format,

Value

list

Examples

```
file <- system.file("workspaces", "workspace_test", "X13Spec",
                    "X13Spec-1.xml", package = "rjd3workspace")
my_spec<-x13_read_spec(file)
class(my_spec)
str(my_spec)
```

---

x13_write_spec *Write a X13 specification file*

Description

The specification file is a xml file like the one JDemetra+ would write when defining a specification in the Graphical User Interface.

Usage

```
x13_write_spec(spec, file)
```

Arguments

spec	a specification created with rjd3x13::x13_spec
file	xml format

Value

NULL returned invisibly

Examples

```
# Creating a spec from default
x13_spec <- rjd3x13::x13_spec("rsa3")

# Forcing multiplicative model
x13_spec_d <- rjd3toolkit::set_transform(
    x13_spec ,
    fun = "Log",
    outliers = TRUE
)

# Writing the specification in a xml file
spec_path <- tempfile(fileext = ".xml")
x13_write_spec(x13_spec_d, file = spec_path)
```

Index